Rule-Based System Architecting of Earth Observation Satellite Systems

by

Daniel Selva Valero

Diplôme d'Ingénieur, ENSAE Supaero, Toulouse, France (2004)
Enginyer Superior de Telecomunicacions, Universitat Politècnica de Catalunya, Barcelona, Spain (2004)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUNE 2012

Signature of Author……..……..
Daniel Selva Valero
Department of Aeronautics and Astronautics
June 2012

Certified by……..
Prof. Edward F. Crawley
Ford Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Supervisor

Certified by……..……..
Prof. David W. Miller
Full Professor of Aeronautics and Astronautics
Committee Member

Certified by……..……..
Prof. Ronald G. Prinn
TEPCO Professor of Atmospheric Science; Director, Center for Global Change Science
Committee Member

Certified by……..……..
Dr. Christopher J. Scolese
Director, NASA Goddard Space Flight Center
Committee Member

Accepted by……..……..
Prof. Eytan H. Modiano
Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Rule-Based System Architecting of Earth Observation Satellite Systems

by

Daniel Selva Valero

Submitted to the Department of Aeronautics and Astronautics
on May 21st, 2012 in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

Abstract

System architecting is concerned with exploring the tradespace of early, high-level, system design decisions with a holistic, value-centric view. In the last few years, several tools and methods have been developed to support the system architecting process, focusing on the representation of an architecture as a set of interrelated decisions. These tools are best suited for applications that focus on breadth – i.e., enumerating a large and representative part of the architectural tradespace –as opposed to depth – modeling fidelity.

However, some problems in system architecting require good modeling depth in order to provide useful results. In some cases, a very large body of expert knowledge is required. Current tools are not designed to handle such large bodies of knowledge because they lack scalability and traceability. As the size of the knowledge base increases, it becomes harder: a) to modify existing knowledge or add new knowledge; b) to trace the results of the tool to the model assumptions or knowledge base.

This thesis proposes a holistic framework for architecture tradespace exploration of large complex systems that require a large body of expert knowledge. It physically separates the different bodies of knowledge required to solve a system architecting problem (i.e., knowledge about the domain, knowledge about the class of optimization or search problem, knowledge about the particular instance of problem) by using a rule-based expert system.

It provides a generic population-based heuristic algorithm for search, which can be augmented with rules that encode knowledge about the domain, or about the optimization problem or class of problems. It identifies five major classes of system architecting problems from the perspective of optimization and search, and provides rules to enumerate architectures and search through the architectural tradespace of each class. A methodology is also defined to assess the value of an architecture using a rule-based approach. This methodology is based on a decomposition of stakeholder needs into requirements and a systematic comparison between system requirements and system capabilities using the rules engine.

The framework is applied to the domain of Earth observing satellite systems (EOSS). Three EOSS are studied in depth: the NASA Earth Observing System, the NRC Earth Science Decadal Survey, and the Iridium GEOscan program. The ability of the framework to produce useful results is shown, and specific insights and recommendations are drawn.

Thesis Supervisor: Edward F. Crawley

Title: Ford Professor of Engineering

Acknowledgements

First and foremost, I would like to thank my advisor Prof. Ed Crawley for introducing me to the field of systems architecture, and for teaching me almost everything I know about how to architect a system. I am a much better engineer as a result of my exposure to his way of thinking about systems. I am also grateful for all the weekends of ski, flying, soaring, golf, and fun with the dudes in Vermont.

The other members of my doctoral committee also deserve a special mention. Prof. Dave Miller taught me how to design and build a satellite starting from a blank sheet, and always had insightful comments about my theoretical systems framework. Prof. Ron Prinn taught me the ABC of atmospheric chemistry and climate change science, and provided me with very useful first-hand information about several Earth observing missions. I was privileged to have the former NASA associate administrator Chris Scolese, now director of the Goddard Space Flight Center, as a member of the committee. This thesis is so much better thanks to Chris' insightful remarks all along the way. His encyclopedic knowledge of Earth observing missions combined with his outstanding engineering skills was an extremely useful resource.

Many other professors and researchers have been very helpful during my four years at MIT. Prof. Kerri Cahoy and Dr. Bill Blackwell provided very useful feedback about my dissertation, in addition to valuable expert knowledge in their own fields of remote sensing. Prof. Sara Seager taught me the physics of atmospheric radiation, my favorite subject at MIT, and her support and encouragement meant so much to me. Prof. Oli De Weck taught me the fundamentals of systems engineering, and introduced me to the field of multidisciplinary design optimization, that has inspired so much of my research. Prof. Jay Sussman taught me how to make beautiful computer programs that reason like humans. Finally, Prof. Dave Staelin, who sadly passed away before I could complete my thesis, was my minor advisor and taught me so much about remote sensing. Prof. Staelin was a remarkable teacher and mentor, and it was truly an honor to be one of his last students.

Thanks also to Kathi Brazil, Amy Shea, and Beth Marois for their help and eternal kindness.

This research was done thanks to the financial support of la Caixa foundation, NASA Goddard Space Flight Center, Draper Laboratory, and NASA Headquarters. Thanks in particular to Emilia Jordi, Bob Connerton, Bernie Seery, and Shawn Murphy for believing and investing in me.

My colleagues have been a continuous source of inspiration, motivation, guidance, and support. Alessandro Golkar read most of this document and provided insightful feedback about most of my research ideas. Matt Smith, Wilfried Hofstetter, Theo Seher, Maokai Lin, Tim Sutherland, Richard Rhodes, Paul Grogan, Wen Feng, Matt Silver, Anthony Wicht, Brandon Suarez, Jonathan Battat, Emily Calandrelli, Alexander Rudat, and Morgan Dwyer, and the rest of 33-409ers made these 10,000+ hours of research in my cubicle an extraordinary personal experience. I would totally do another PhD with you guys. I might do it at some point.

When I look back at the last four years of my life, I just cannot believe how blessed I was to meet my Bostonian family: Carlos Pardo, Jorge Cañizales, Maite Balda, Alessandro Golkar, Maria Jose Nieves, Ada Yeste, Maria De Soria, and Fernando de Sisternes. What an amazing group of individuals. I was humbled by all of you almost every day of these four years. I simply cannot conceive what my PhD would have been without you.

Last, I would like to thank my family for their unconditional and eternal support. My mom gave me everything I needed between the ages of 0 and 22 without even asking for anything in return. I took from my dad the penchant for aerospace engineering, and the ambition to go further and further. My sister has helped me in so many ways, and she means everything to me. “La Tata” simply was my second mom, and I am so grateful to her. And most of all, I would like to thank Ana, my future wife, for her unconditional support in this adventure, and for bearing with me during these four years. It is only because of your love and support that we made it.

Contents

Table of Figures

1 Introduction

1.1 Overview

Recent research in the field of system architecture studied the use of meta-languages (H.-yung B. Koo, 2005) and architecture decision graphs (Simmons, 2008) for decision support to system architects. This thesis explores a new class of decision support tools for system architecting based on the incorporation of a **rule-based engine** with two main purposes: to improve the **scalability** of the tool, i.e. the ease to add new knowledge or modify existing knowledge; and to increase the **transparency** of the tool, i.e. the traceability of the results to the modeling assumptions and the knowledge base. This new framework is called **rule-based system architecting**, in order to differentiate it from decision-based system architecting, and meta-language-based system architecting.

Scalability and transparency make rule-based system architecting particularly suitable for **knowledge-intensive problems** in system architecting. A knowledge-intensive problem requires a model containing a large body of expert knowledge in order to produce useful results. As the size of the body of knowledge included in the model grows, it becomes harder to add new knowledge or modify existing knowledge. Furthermore, as the model grows, it also becomes harder to understand the reasoning behind the results. Typically, a system architecting problem is knowledge-intensive when there is complexity in the formal decomposition of the system (i.e., the system consists of a large number of interconnected elements), and/or in the functional decomposition of the system (i.e., there are many stakeholders with different sets of needs that result in long lists of competing system requirements). Furthermore, the knowledge-intensive nature of these problems is intensified when the link between requirements and capabilities is non-trivial and exhibits high degrees of emergent behavior and coupling.

Better scalability is obtained in rule-based system architecting through the physical **separation of the different bodies of knowledge** required to solve a complex system architecting problem. A system architecting problem or SAP can be formulated as a constrained, multi-objective combinatorial optimization problem. For example, many SAPs can be formulated as generalized assignment problems, or set partitioning problems. Thus, solving an SAP requires knowledge in optimization, search, and decision analysis on one side (domain-independent knowledge), and knowledge about the domain of application on the other side (domain-specific knowledge). Moreover, part of the domain-specific knowledge will be common to all systems in the domain (e.g. L-band maximizes sensitivity to soil moisture retrievals), whereas part of the domain-specific knowledge will also depend on the instance of problem being considered (e.g. specific requirements for the NASA Earth Observing System). A similar argument applies to domain-independent knowledge: part of this knowledge is common to all SAPs (e.g., how to search a generic tradespace, or find a Pareto frontier); part of it is common to all instances of the same class of SAP (e.g., how to efficiently enumerate set partitioning architectures); and finally part of this knowledge is really dependent on the specific instance of SAP at hand (e.g., what is the relative importance of the different metrics to the stakeholders).

Separating these bodies of knowledge involves physically putting them on separate files in the decision support tool. This naturally results in increased scalability, and **enables the system architect to focus on the domain knowledge** only, leaving domain-independent knowledge to software and modeling engineers.

Enhanced transparency is obtained in rule-based system architecting through the construction of an **explanation facility**. Every result produced by the tool is accompanied by a description of the important highlights of the value delivery loop. This is an important feature, as it increases the confidence of the user on the tool and improves the man-machine interaction, which can potentially lead to better results. The structure of rule-based expert systems is a convenient framework around which to build an explanation facility, as **the trace of the rules fired during execution is a very good approximation of the value delivery loop**. Furthermore, explanation facilities tend to be knowledge-intensive applications by themselves, and thus a knowledge-based approach such as a rule-based system is appropriate.

One of the findings of this thesis is that **rule-based system architecting is also naturally suited to model emergent system behavior.** This is very important because the value of a system architecture usually arises from interactions between elements of the architecture, rather than by simple superposition of independent behaviors. As in agent-based modeling, the behavior of several functional and formal elements of the system architecture as well as their interactions can be modeled through simple rules that, once they are simultaneously executed, lead to emergent behavior.

The methodology and tools developed in this thesis are applied to the domain of **Earth Observing Satellite Systems** (EOSS). An EOSS can be defined as a collection of Earth observing satellite missions that together satisfy a set of measurement requirements. EOSS architecting involves three classes of architectural decisions that affect value delivery the most: selecting the instruments for taking these mesurements, allocating these instruments into satellites, and scheduling the launch of the resulting missions.

EOSS turn out to be a very good domain of application for rule-based system architecting for several reasons. First, they are undoubtedly **complex systems-of-systems** with a large number of relevant system components (namely missions and instruments). Second, they have complexity in the functional domain as well, because requirements come from all the disciplines of the Earth sciences and all possible remote sensing technologies is required in order to produce useful results. Third, their value delivery loop is often driven by **emergent behavior**. Indeed, in many cases, several aspects of the architecture of EOSS are driven by factors such as synergies between measurements and data products, or interferences between instruments on a certain spacecraft.

The architecture of three different EOSS is analyzed in this thesis. The **NASA Earth Observing System**, the NRC **Earth science decadal survey**, and the **GEOscan** program for the Iridium NEXT hosted payload opportunity. These EOSS are very different and stretch the tool and methodologies in different ways. The NASA EOS retrospective case study is used as a benchmark in order to assess the ability of the tool to produce useful results. Replicating the decisions made in the architecting process of such a complex system requires high fidelity modeling capabilities. The Decadal Survey is similar in scope and characteristics to the NASA EOS, but the uncertainty on the instrument characteristics is much higher, as some of these instruments are still in the early stages of development. The main challenge of the Decadal case study resides in demonstrating the ability of the tool to capture the main trade-offs between different mission concepts and bring them up to the program level. Finally, the GEOscan case study concerns an extremely different architecture.

Comparing traditional architectures based on a few medium or large satellites need to this hosted-payload based architecture is challenging for many reasons. In particular, the model needs to have enough modeling fidelity to capture subtle differences between the capabilities of large instruments and Cubesat-class instruments.

While this thesis is only the first exploratory step towards this new class of architecting tools, it demonstrates the feasibility of the separation of bodies of knowledge, and the ability of the framework to handle complex, knowledge-intensive system architecting problems and produce useful results.

1.2 System Architecture and System Architecting

In the late 80's, researchers started to realize that some of the concepts used in building architecting and civil engineering were being used by engineers in charge of designing and building unprecedented, large, complex systems (Maier & Rechtin, 2000). These concepts included the creation of a separate position for a lead systems engineer – or system architect – at the interface between the client and the design team, a more direct engagement of the client in the high-level design of the system, and a holistic, value-centered, lifecycle view of the system. Rechtin was arguably the first to formalize this concept, and he coined the term "systems architecting" (Rechtin, 1991). His book with Maier is still considered by many experts the best introduction to the field (Maier & Rechtin, 2000).

While there are many definitions for system architecture, **the architecture of a system essentially is its highest level design**. However, it takes a holistic view that goes beyond traditional design. More precisely: 1) it takes into account technical and non-technical factors; 2) it is centered in delivering value to stakeholders as opposed to optimizing performance or cost; 3) it takes into account all the phases of the lifecycle including manufacturing, testing, operations and disposal.

System architecture is concerned with the earliest decisions for the design of a complex system, and these decisions are particular, for several reasons: 1) they have to be done in a highly ambiguous context; 2) they commit the largest part of the lifecycle cost of the system; 3) they have the largest impact on subsequent design decisions; 4) they have the largest impact on performance, risk, flexibility and other figures of merit. Once the main architectural decisions are made, the concept or essence of the system is fixed.

More formally, Crawley et al define architecture as "*an abstract description of the entities of a system and the relationships between those entities*" (E. Crawley et al., 2004). Two broad categories of entities are distinguished: elements of **function** or processes, i.e., "what the system does" in order to provide value to the stakeholders; and elements of **form**, or objects, i.e., "what the system is", or the set of tangible elements that the system is composed of. The mapping between the main elements of function and the main elements of form constitutes the **concept**. The definition of function, form, and concept, allows a more precise definition of system architecture, also by Crawley: "*system architecture is the embodiment of concept: the allocation of physical/informational function to elements of form, and the definition of interfaces among the elements and with the surrounding context*" (E. F. Crawley, n.d.).

System architecting is the process of creating a system architecture. The typical process is similar to that of a trade study, and consists of three steps: 1) the system architect(s) select a handful of candidate architectures; 2) each architecture is assessed, typically in terms of cost and performance; 3) one or two architectures are selected for further studies. This process is far from being ideal for several reasons, mostly related to the bias that the human system architect(s) and their organization bring from their previous experience and expertise. This motivates the use of decision support tools in system architecting, in order to bring rigor and consistency into the process.

1.3 System architecting, decision analysis, and combinatorial optimization

1.3.1 System Architecting Problems

Simmons showed in his dissertation that **the system architecting process can essentially be seen as a decision making process** where the decisions to make concern the description of the entities of the system, as well as the relationships between them (Simmons, 2008). Therefore, system architecting can benefit from the literature on **decision analysis**.

Since the goal of the system architects is to find the best possible system architecture, the process can also be seen as an optimization problem, where the variables are the architectural decisions to be made, and the objective function capture value delivery to stakeholders. Constraints can also be added that capture for example logical interrelationships between the decisions. In particular, **optimization problems that appear in system architecting are typically very hard to solve** because: 1) they are non-convex, i.e., they have multiple optima; 2) they are integer, mixed-integer, or most often combinatorial, because variables are integers or Booleans and the set of allowed values for each variable is usually discrete; 3) they are large-scale, because the domain is usually a combinatorial space; 4) they are non-linear, both in the objective functions and in the constraints. Consequently, system architecting can also benefit from the literature on **large-scale combinatorial optimization**.

Finally, any optimization problem can be reformulated as a search or constraint satisfaction problem by adequately defining a goal state. Thus, the literature on **search and constraint satisfaction** from artificial intelligence is also relevant.

For the rest of this thesis, the term System Architecting Problem (SAP) will be used to designate the formulation of a real system architecting problem as either: 1) a decision making problem; 2) a combinatorial optimization problem; 3) or a search problem.

Note that this definition does not imply a one-to-one mapping between real system architecting problems and their formulations as SAPs. Indeed, the same real problem can be formulated in as many different ways as there are mathematical models to represent the same reality. In fact, multiple SAPs can often be used to gain insight into different aspects of a real life system architecting process.

1.3.2 System Architecting Tools

Today, it is widely accepted that computational tools can help improve the system design and architecting processes by providing: 1) a rigorous framework for objective and consistent evaluation of a large number of architectures under a variety of scenarios; 2) several figures of merit derived directly from stakeholder needs; 3) guidance for the search and selection processes (Shim et al., 2002).

In the context of this thesis, we introduce **the term "System Architecting Tool" (SAT) to designate any computational tool that is used with the purpose of solving an SAP**. SATs include in particular decision support tools (DSTs), combinatorial optimization algorithms (COAs), and search and constraint satisfaction algorithms (SAs).

Simmons used the term "architectural decision support system" to refer to SATs. In his dissertation, he defines four desirable aspects of an architectural decision support system: the **representational aspect**, which focuses on the process of formulating the SAP with its decision variables, constraints, and metrics; the **structural reasoning aspect**, which analyzes the interconnectivity between the decision variables; the **simulation aspect**, which concerns the enumeration and evaluation of feasible architectures; and the **viewing aspect,** which is related to the graphical representation of architectures, metrics, or more generally any data coming from structural reasoning and simulation. While Simmons focused on DSTs, these four aspects remain applicable to all SATs, and they will be used in the rest of this chapter. The state-of-the-art of DSTs, COAs, and SAs is briefly reviewed in the following sections, using Simmon's four desirable aspects of an SAT as a descriptive framework.

1.3.2.1 Decision Support Tools

Decision support tools (DST) can be used to solve SAPs formulated as decision making problems. Simmons classified DSTs in four groups: table and matrix-based DSTs, tree and directed graph-based DSTs, constraint graph-based DSTs, and meta-language-based DSTs.

Table and matrix-based DSTs, such as Design Structure Matrices (DSM) and morphological matrices, provide a means to represent and analyze the structure of the information needed to make a decision. **Morphological matrices** were invented by Zwicky (Zwicky, 1969) as a general strategy for problem solving by enumerating the possible solutions. In morphological matrices, each row represents one decision, and the options for each decision appear in successive columns. Thus, an architecture is obtained by choosing one option (column) for each decision (row). **Design Structure Matrices** are adjacency matrices that are used to study bilateral dependences between decisions, or more generally between the elements in a system.

Tree and directed-graph-based DSTs are used to represent sequential decisions, often under uncertainty. The simplest tool is perhaps the decision tree. In a **decision tree**, there are three types of nodes: decision nodes, where a decision is made amongst a discrete set of alternatives; chance nodes, where a state of nature is randomly chosen between a discrete set of possible states; and an outcome node, a terminal node where the outcome of a path within the decision tree is made explicit. The "atomic element" of a decision tree is a decision node connected to as many chance nodes as there are options for the decision. Hence for example, if a decision has three possible alternatives, the atomic element would have a chance node for each of the three alternatives. A decision tree typically consists of a number of these "atomic elements", terminated by as many outcome nodes as needed. Note that the number of branches in a decision tree grows exponentially with the number of decisions. Other tools such as influence diagrams and sequential decision diagrams are used to tackle this so called "curse of dimensionality by exploiting dependences between the decisions. All these methods are used in combination with the criterion of maximizing the expected outcome over all possible paths in the tree. This is a limitation, since it implies linearity of the objective functions.

Simmons included **Markov decision processes** in this category of directed-graph based DSTs. Let $X = \{x_i\}$ be a set of states, $A_i = \{a_{ij}\}$ be the set of alternatives for each state x_i, $R_{ij} = \{r_{ij}\}$ be the rewards for taking alternative j at state i, and $P_{ij}(x_i, x_j)$ the probabilities of transitioning between states i and j.

Then, the tuple $\langle X, A, R, P \rangle$ is called a Markov decision process if and only if the probability of transitioning to a certain state only depends in the current state, and not on the history of states: $P_{ij}\left(X_{t+1} = x_j \middle| X_t = x_i, \{X_k\}_{k=0}^{t-1}\right) = P_{ij}\left(X_{t+1} = x_j \middle| X_t = x_i\right)$. This property is called the Markov property. MDPs typically use dynamic programming approaches that exploit the Markov property by solving the problem recursively, which results in computationally efficient algorithms (Puterman, 1994). However, the use of MDPs and dynamic programming requires formulating the problem with a structure that allows this divide-and-conquer approach. Finding such structure can be non-trivial in some cases, unless the problem is simplified to neglect dependences between system entities.

Any attempt to overcome the limitations of the Markov property by including the history of states in the current states, as it is done in time-expanded decision networks (Silver & de Weck, 2007), will result in exponential computational complexity, thus eliminating much of the original interest in MDP formulations.

Constraint graph-based DSTs include the tools used to solve **constraint satisfaction problems** (CSPs) and **constraint optimization problems** (COPs). CSPs consist of a set of decisions, each with a discrete set of alternatives, plus a set of hard constraints that need to be satisfied in order for a solution to be feasible. The emphasis in original CSPs is to find a feasible architecture as opposed to finding the best architecture. This is accomplished through backtracking, which essentially is a way of keeping track of the last successful decision nodes visited in a tree, so that one can go back to a "safe node" in case of reaching a region of the tree that is infeasible. Backtracking is sometimes augmented with several variants of constraint propagation, in order to improve efficiency.

Constraint optimization problems (COPs) are an augmented version of CSPs in which a set of metrics is added in the form of soft constraints that are combined in a weighted average. The output of the COP is a solution that satisfies all the hard constraints and minimizes the weighted average of the soft constraints.

Variants exist for COPs and CSPs that relax the hard constraints by including some kind of penalty scheme, as it is done in Lagrangian relaxation (Fisher, 2004).

CSPs in which all constraints are Boolean in nature are called Boolean satisfiability or simply SAT problems. Although SAT problems are NP-complete, heuristics can be incorporated into the classical backtracking approach to improve efficiency. Examples of these heuristics include conflict-driven strategies and look ahead strategies (Williams & Ragno, 2007).

Efficient algorithms also exist that transform any COP into a set of SAT problems. Recently, Rayside et al combined these algorithms with an efficient SAT solver to create a general purpose multi-objective optimization tool especially tailored for system architecting called the Guided Improvement Algorithm (Rayside, Estler, & Jackson, 2009).

Meta-language-based DSTs is the term Simmons and Koo coined to describe the latter's Object-Process Network (OPN) language. While Simmon's assessment of the needs of the system architect included four aspects (representation, structural reasoning, simulation, and viewing), Koo identified three main tasks of the system architect: **encoding**, **enumeration** and **evaluation**. Encoding fits well in Simmon's representation aspect, while enumeration and simulation belong to Simmon's simulation aspect. The representational part of Koo's OPN allows the formulation of any SAP using a simplified version of Dori's Object Process Methodology with objects, processes, and relationships. The simulation part utilizes Petri Nets to enumerate and evaluate all feasible paths through the OPM graph that represents the architecture. While OPN is theoretically Turing complete, Simmons and Koo found that in practice it can be quite hard to formulate a generic system architecting problems in the OPN language (see (Simmons, 2008), page 45).

1.3.2.2 Combinatorial optimization algorithms

Optimization algorithms can be used to solve system architecting problems formulated as combinatorial optimization problems. The resulting optimization problem is almost always extremely hard to solve, more precisely NP-complete[1]. Indeed, SAPs are usually non-linear, non-convex, combinatorial, multi-objective optimization problems. A thorough introduction to the topic of combinatorial optimization can be found in (Ehrgott & Gandibleux, 2000). There are several approaches to deal with these problems: the integer optimization approach, the stochastic optimization approach, the dynamic programming approach, and the metaheuristics approach.

[1] P is the class of problems for which we can find a solution in polynomial time. NP is the class of problems for which we can check a solution in polynomial time. NP-hard problems are at least as hard as the hardest problems in NP, i.e., any NP problem can be reduced to any NP-hard problem in polynomial time, but they do not necessarily belong to NP. NP-complete problems are simultaneously NP and NP-hard. This assumes that P≠NP.

In operations research, the most common approach by far is the use of integer optimization techniques. These techniques require a linearization of the objective functions and constraints in order to exploit the nice properties of linear optimization, using simplex-based methods such as branch and bound (Lawler & Wood, 1966), cutting planes (Kelley, 1960), lagrangian relaxation (Fisher, 2004), and approximation algorithms (Hochba, 1997). When linearization is not possible, but the objective functions are still convex, semidefinite programming (Vandenberghe & Boyd, 1996), and sequential quadratic programming (Boggs & Tolle, 1995), are amongst the most common approaches currently used.

More advanced techniques include the use of matroids and submodular functions amongst others (Schrijver, 2003).

Dynamic programming (DP) approaches are also very popular as they provide polynomial time solutions to problems that have the required structure [31]. However, as explained earlier in the section describing Markov Decision Processes, DP approaches often suffer from exponential space complexity. Approximate dynamic programming (ADP) techniques such as Q-learning are used for large-scale problems in which the number of states is too large. ADP techniques sample both the state space and the control space, and also across disturbances in the case of stochastic DP (Bertsekas, 1996).

For general, non-convex optimization, i.e., problems that may have multiple local optima, there currently exists no general efficient algorithm, especially when dealing with large-scale problems, which is very often the case due to combinatorial explosion. The common trade-off made to tackle this problem is to sacrifice exactness of the solution to gain in computational time. In other words, the architectural tradespace is not exhaustively explored anymore, losing the ability to guarantee that the solution found when the algorithm stops actually is the global optimum.

A common approach to solve large, non-convex optimization problems is the use of metaheuristic algorithms, such as genetic algorithms, or simulated annealing, which incorporate both functions. In addition to a mechanism of "hill climbing", metaheuristic algorithms feature some degree of randomness to avoid being trapped in local minima (e.g., mutation in genetic algorithms). Amongst the most popular metaheuristic algorithms are **genetic algorithms**, developed by Holland in the 70's [37]. In genetic algorithms, the architectural information is stored in genes and chromosomes, and good individuals are "crossed", meaning that their genes are combined to produce new individuals, with the hope that some of the children will outperform their parents. Furthermore, some individuals are randomly mutated to increase diversity. Thus, the rules of natural selection are imitated to mime evolution. Although still today, there is controversy about the theoretical foundations that make genetic algorithms work [38], they have certainly been applied with success to a variety of optimization problems. In particular, genetic algorithms have been successfully applied to large set partitioning problems, which are of interest in this thesis (see, for example, [39]).

Another popular metaheuristic algorithm is **simulated annealing** [40]. In simulated annealing, the process of controlled cooling of hot metals is mimed: a "temperature annealing schedule" is used to vary the probability of choosing a solution that is far away from the current solution.

This temperature schedule typically starts from a relatively high temperature and strictly decreases until reaching zero at convergence. Hence, the temperature schedule plays the role of the probability of mutation function in genetic algorithms.

A third metaheuristic algorithm is **particle swarm optimization** (PSO) [41]. PSO, like GAs, uses a population-based approach. This means that, at each iteration, the algorithm has a family of good solutions as opposed to a single optimal solution. In PSO, the phenomenon that is being mimed from nature is the movement of swarms of birds. For a given individual in the population, the next solution in the search is obtained from: 1) the last solution visited by the individual; 2) the best solution visited by the individual; 3) the best solution visited by the group; 4) some degree of randomness.

Finally, we cite **tabu search** as another meta-heuristic algorithm (Nonobe, 1998). In tabu search, on top of the local search procedure, the heuristic used is a tabu list, a list of previously visited and this forbidden solutions, and optionally a list of forbidden attributes. A recent survey of metaheuristics for combinatorial optimization can be found in (Bianchi, Dorigo, Gambardella, & Gutjahr, 2008).

1.3.2.3 Search and constraint satisfaction tools

The duality between optimization and search problems was highlighted before. Indeed, search algorithms can be utilized to solve SAPs that are formulated as search problems.

In parallel to the developments in the operations research community, the artificial intelligence community made a lot of progress in developing efficient **search algorithms**. In search problems, the goal is to iteratively find a point that satisfies a certain number of properties. It is obvious that an optimization problem is an instance of a search problem, in which, for example, constrains are dynamically added to find a new solution that is better than the former.

Search algorithms are also called graph algorithms, and are divided in two large classes: uninformed and informed methods. **Uninformed search algorithms** are those in which the search process is blind: the algorithm can only distinguish a solution from something that is not a solution. Typical uninformed methods are: breadth-first, uniform cost, and depth-first. Sometimes, a heuristic function can be defined that guides the search, and under certain conditions, improves efficiency, leading to i**nformed or heuristic search algorithms** (not to be confused with meta-heuristic algorithms previously defined). One of the most popular heuristic search algorithms is A*. For an efficient implementation of A*, see Brian William's paper on conflict-directed A* (Williams & Ragno, 2007).

A discussion on search algorithms can be found in (Russell, Stuart; Norvig, 1995). Despite the clear correspondence between optimization and search problems, these two communities have taken a long time to collaborate and produce promising hybrid techniques. An example of a hybrid technique that combines genetic algorithms with search algorithms can be found in (Glover, Kelly, & Laguna, 1995).

1.4 Needs of the system architects vis-à-vis SATs

In this section, we are interested in the tasks of the system architect that can be supported by a formulation as an SAP. We start by retrieving Crawley's definition of system architecture: "*the embodiment of concept, and the allocation of physical/informational function (process) to elements of form (objects) and definition of structural interfaces among the objects.*" SAPs appear at different steps or tasks of the system architecting process, most of which are explicitly mentioned in this definition of system architecture. These tasks are listed below, in rough order from general to specific:

- Function-to-form mapping: The allocation of physical or informational elements of function into elements of form is one of the earliest and most important architectural decisions that have to be made. In this case the goal of the SAP is to find the optimal assignment of elements of function to elements of form. Note that in many cases systems have integrated concepts, i.e., the overall concept can be decomposed into a set of function-to-form mapping decisions.

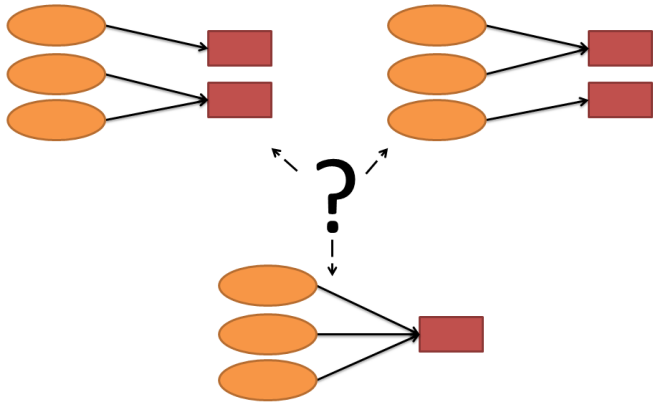


Figure 1: The function-to-form mapping task in the system architecting process

- Decomposition/aggregation (of function and form): Usually, the task of decomposition or aggregation of function and form is a rather qualitative task and the system architects do not require the support of computational tools to conduct it. However, occasionally, system architects must make important decisions concerning the decomposition or, more often, the aggregation of both objects and processes. Let us consider for instance a resource exploration system with a certain number of reservoirs, studied by Golkar and Crawley (Aliakbargolkar & Crawley, 2012). The architectural decision of choosing the number of platforms to build can be seen as a form aggregation problem. Another instance of the decomposition/aggregation problem is the decision of the number of modules in the architecture of a large software system. Note that in both cases there is a set of elements (objects or processes) that are given, and we are concerned about how to decompose or aggregate these elements in groups.

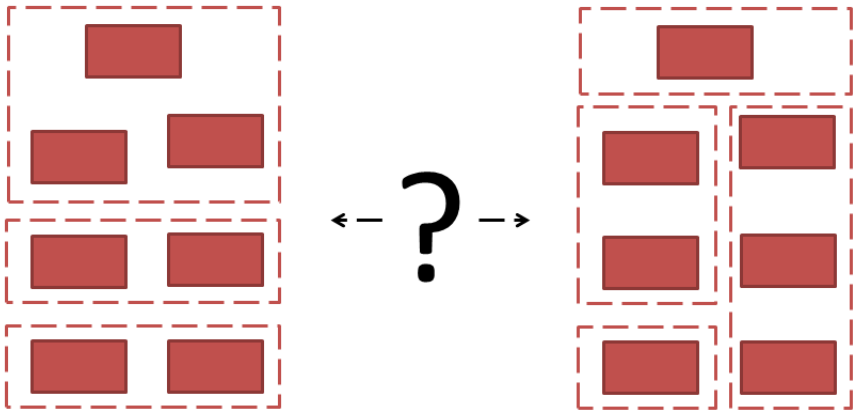


Figure 2: The function/form decomposition/aggregation task in the system architecting process

- Specialization of function and form: During the system architecting process, the system architect must go from solution neutral objects and processes to solution-specific objects and processes. This specialization process consists of many architectural decisions in which for every solution neutral or object or process, the system architect has to choose between several possible solution-specific objects or processes. For instance, let us consider the simple solution-neutral function of "food preserving". Food preserving is accomplished by chilling in a refrigerator, but it can also be accomplished by freezing, salting, or even irradiating with X-rays. The choice between these solution-specific functions is an instance of the function/form specialization task.

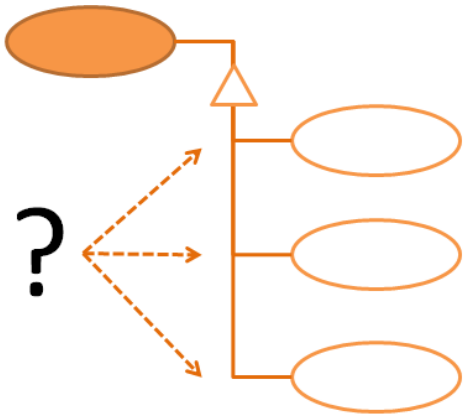


Figure 3: The specialization task in the system architecting process (OPM)

- Architectural attributes selection: Architectural attributes play a key role in the definition of the system architecture. Examples or architectural decisions at the attribute level for a launch vehicle are the number of boosters (an attribute whose possible values can be for example {0, 1, 2, 4}), and the type of propellant (an attribute whose possible values could for instance be {LH2, RP1, CH4}).

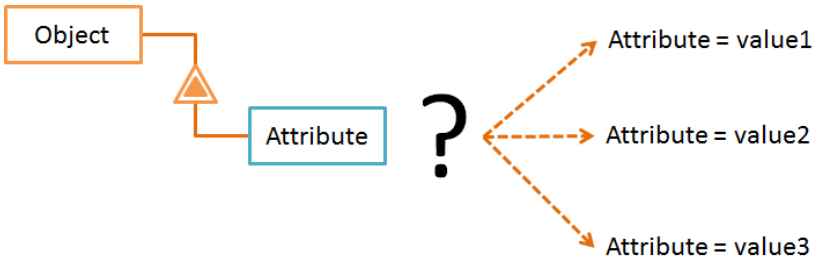


Figure 4: The architectural attribute selection task in the system architecting process (OPM)

- Structural relationships and interfaces definition: In almost any imaginable definition of system architecting, some notion of connectivity between elements, interfaces, or more generally structural relationships between elements must be mentioned. This task typically occurs in the form domain, and these relationships are often, but not always, physical connections between elements. Continuing with the example of the resource exploration system, an important architectural decision to make the connectivity between platforms and reservoirs, as well as between platforms.

  Another example can be found in the architecture of Guidance, Navigation and Control (GN&C) systems for space vehicles. Given an architecture defined by a number of sensors, computers, and actuators, the decisions of which elements to connect will affect both cost and other emergent system properties such as its reliability (Dominguez-Garcia, Hanuscahk, Hall, & Crawley, 2007).

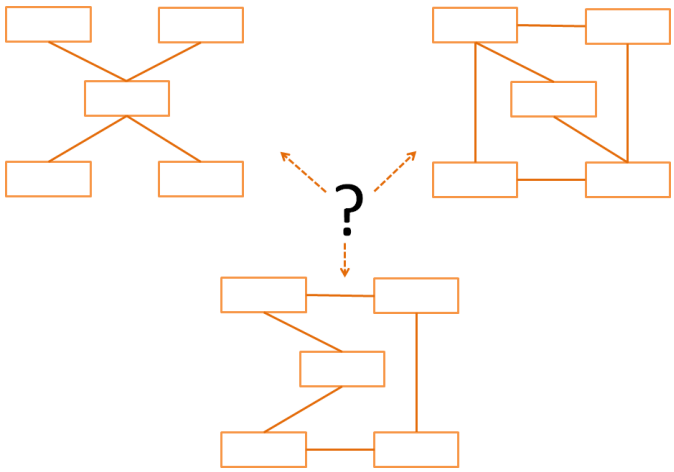


Figure 5: The structural relationships and interfaces definition task in the system architecting process

In addition to these tasks, two higher level tasks appear that are linked to the topic of commonality in product families. Commonality is defined by Cameron as the reuse and sharing of assets such as components, processes, technologies, interfaces or infrastructure across a product family. We identified two different classes of architectural decisions in commonality in which SAPs are useful:

- Reactive commonality: In this task, a set of elements or building blocks is given, and the goal is to define the optimal family of systems using these building blocks. An example of this task would be to create a family of launch vehicles using a generic booster, a generic first stage cryogenic engine, and a generic upper stage cryogenic engine. A family of launch vehicles that span the range of performance required in the market can be architected by combining different numbers of boosters, cryogenic engines in the first stage, and incorporating or not the upper stage. This situation is illustrated in Figure 6.

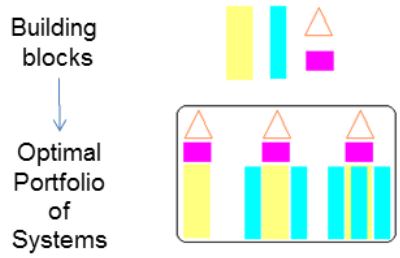


Figure 6: The reactive commonality task in the system architecting process

- Proactive commonality: In this task, a target family of systems is known, and the goal is to find the optimal set of elements building blocks to create this family of systems. Continuing with the example of the family of launch vehicles, in this case the goal would be to architect a booster, first stage, and upper stage, in order to optimize a family with a small launcher to put 1mt satellites in LEO, a medium launcher to put 5mt satellites in GTO, and a heavy lift launcher to bring 20mt of payload to escape. This situation is illustrated in Figure 7. We note that the boundary between reactive and proactive commonality in real-world commonality problems may be somewhat fuzzy. In most problems though, there will be a set of components that are to be reused (reactive commonality), and a set of new components that are to be designed in order to be used in more than one system (proactive commonality).

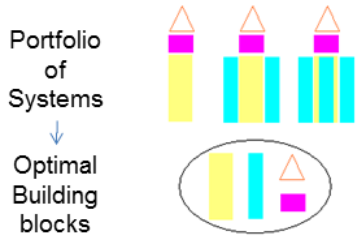


Figure 7: The proactive commonality task in the system architecting process

- Scope selection: In this task, a subset of requirements or capabilities is selected for a certain funding level. Scope selection typically occurs in the form domain. Instead of selecting the scope in terms of requirements, it is more common to select the scope in terms of the corresponding elements of form, as it is elements of form which actually carry the cost. This situation is illustrated in Figure 8.

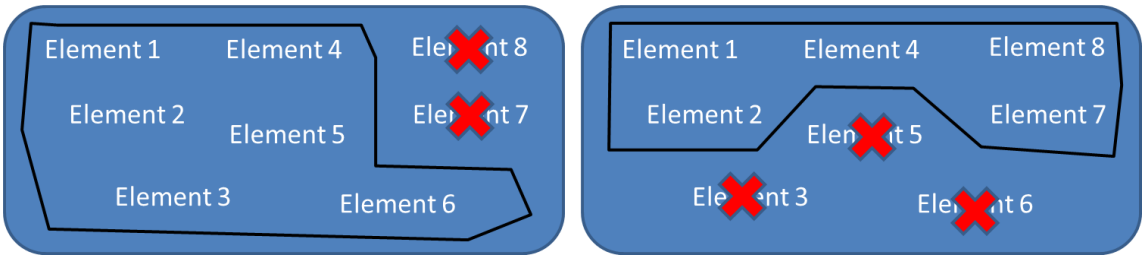


Figure 8: The scope selection task in the system architecting process

The previous list is not an exhaustive list of the tasks of the system architect. However, it intends to be a list of the most common tasks of the system architect that can benefit from a rigorous formulation as an SAP. Therefore, it follows that most SAPs fall into either of the previous categories of tasks. This observation is central because it allows focusing on the thorough description and solution of these tasks and corresponding SAPs. This requires the formulation of these problems as combinatorial optimization problems. This step is explicitly done in Chapter 2.

1.5 General problem statement

1.5.1 Knowledge-intensive SAPs

The Webster's Revised Unabridged Dictionary defines the adjective "**knowledge-intensive**" as "*requiring access to and manipulation of large quantities of knowledge*". In the context of this thesis, we define a **knowledge-intensive SAP as one that requires a large body of domain-specific and domain-independent knowledge in order to be solved efficiently**, i.e., in order to find good architectures in a limited amount of time.

In this definition, **domain-specific knowledge** refers to the domain of application of the SAP (e.g., launch vehicles, Earth observing satellites), whereas **domain-independent knowledge** refers to the tasks being conducted by the SAT regardless of the domain of application (e.g., enumeration, evaluation, search, down-selection).

Since this is a qualitative and deliberately ambiguous definition, one might think that any SAP could fit this definition. In reality, certain SAPs require a more limited amount of knowledge in order to be efficiently solved than other SAPs. These SAPs are much easier to plug into any commercially available optimization toolkit such as CPLEX or the Matlab genetic algorithm tool. If the rule-based approach proposed in this thesis is used for this class of systems, a few rules suffice to model system behavior.

For instance, let us consider the architecture of a constellation of GEO communication satellites. In one particular SAP, the system architect may be interested in optimizing design parameters such as the number of transponders, the number of planes, and the number of satellites per plane in a Walker constellation (in which all other orbital parameters are fixed), in order to maximize profit for a certain lifetime. In this SAP, we have a set of architectures that is small enough to be fully enumerated. The objective function is profit, i.e. revenue minus cost, and it can be readily computed from the design variables: revenue is proportional to total number of channels, and in a first approximation cost is proportional to launch cost + satellite cost + operations cost. Cost estimating relationships exist for communication satellites that relate cost to number or mass of transponders. In short, this SAP can be readily expressed as an optimization or search problem and solved in a matter of minutes. The SAP can be made more complicated by using more involved cost models, or by incorporating complex satellite reliability calculations into the analysis. However, if the two formulations of the problem lead to similar results, then one could argue that there is no need for the more complex formulation of the problem.

The problem becomes more subtle if lifetime becomes a design variable, since both revenue and cost depend on design lifetime in a convoluted way: the longer the lifetime, the larger the revenue, but since now different architectures have different time scales, net present value analysis becomes necessary to conduct a fair comparison of the alternatives. At the same time, if design lifetime is increased, it will be needed to increase component reliability or to add redundancy in order to maintain overall system reliability. This will result in an increase in the mass and power consumption of the electronics, which will require in turn an increase in the mass of the electrical power subsystem, all of which will require a heavier structure. A more massive structure will require more propellant to maintain its orbit and attitude, which implies heavier tanks and feeds back again into a heavier satellite. It is easy to see that much more knowledge about spacecraft design is required in order to solve this version of the same problem. And more importantly, this knowledge is essential in order to get results that make sense. If we were to ignore the aforementioned issues in a simpler version of the model, we would be missing important pieces of the trade-off, and thus presumably the results obtained would be of poor quality.

A real system architecting problem is knowledge-intensive if its simplest formulation as an SAP that leads to useful results requires the handling of large body of expert knowledge, (e.g., a very high number of rules if a rule-based engine is used). The following question that arises naturally from this observation is, "what makes a system be knowledge-intensive?" The answer to this question involves two factors: complexity and emergence.

First, knowledge-intensive SAPs typically arise when the system at hand is complex in the form and/or function domain. In other words, in knowledge-intensive SAPs, either the decomposition of stakeholder needs down to the level of system requirements, or the decomposition of the system architecture down to the level of components that perform relevant functions, is very complex.

Second, knowledge-intensive SAPs are typically characterized by intense couplings between system elements and emergent behavior in the value delivery loop.

As a final note, the knowledge-intensive nature of an SAP may be intensified if one wants to incorporate an explanation facility to have better traceability of the value delivery loop. Indeed, explaining the reasoning behind a fact used by an SAT may sometimes require more knowledge than just using the fact.

1.5.2 Identification of Research Gaps and General Research Goals

Careful analysis of the state-of-the-art in system architecting tools reveals several limitations that are enumerated below, classified according to Simmon's aspects of a decision support tool: representing, structural reasoning, simulating, and viewing. Many but not all of these limitations are particularly applicable to knowledge-intensive SAPs.

- Limitations of current SATs related to the "**Representing**" aspect:
  - Most tools do not offer support to deal with **fuzzy knowledge** (e.g. "high" spectral resolution, 1-3 days revisit time)
  - Most tools are tailored for problems that can be easily formulated as a set of decisions, where each decision has a set of options (i.e., ADG framework (Simmons, 2008)). In reality, there are **SAPs for which the ADG formulation is not natural or not trivial**. For instance, the function decomposition problem illustrated in Figure 2 is hard to express using this formulation. The system architect has **little or no support to formulate such SAPs**.

- Most tools do not offer any kind of support to model **emergent behavior**. Modeling even relatively simple emergent behavior can be cumbersome. For instance, in a radar altimetry satellite, modeling the fact that if an atmospheric humidity measurement is cross-registered with an altimetry measurement, then the accuracy of the altimetry measurement improves, requires the incorporation of an additional logical rule testing for this case.

  Furthermore, this logical rule uses knowledge about the instruments characteristics (technology, accuracy), as well as knowledge about the interface between these instruments (cross-registered or not).

  Therefore, all this information needs to be available. It is easy to see that as we scale the model to include several other cases of emergent behavior, scalability issues may appear. Emergent behavior is best modeled when it is possible to model individual agent behavior in the form of simple rules that can then be automatically combined, without the user having to handle or even predict the result of the simultaneous execution of the individual behaviors.

- Limitations of current SATs related to the "**Structural Reasoning**" aspect:

  - Again, most tools are tailored for problems formulated in an ADG-like fashion. Simmons for example created a set of tools to analyze bilateral dependence of decisions (Simmons, 2008). However, as mentioned before, not all architectural decisions are easy to encode using an ADG formulation.

- Limitations of current SATs related to the "**Simulating**" aspect:

  - Most tools are **limited in the fidelity** of the metrics. While some of them allow including pieces of external code (e.g. Python in OPN), there is no explicit support to the system architect to simulate complex behavior in the objective functions of the model.

  - It is hard to modify existing knowledge or incorporate new knowledge into the simulation model, in particular in the case of knowledge-intensive SAPs.

- Limitations of current SATs related to the "**Viewing**" aspect

- While many tools have some graphing capability, most tools have **no traceability of the value delivery loop.** This is especially problematic in knowledge-intensive problems, because their value delivery loop is often particularly long and complex.

In addition to these limitations, a few more general limitations, not related to any single one of Simmon's aspects, are identified:

- **Reusability**: Current tools do not offer any capability to reuse models between domains. If a new SAP appears that is identical in form to an SAP previously solved but requires changing some of the domain-specific knowledge only, it is necessary to restart the model. SATs are often developed **ad-hoc** for a specific project, and with tight time constraints. As a consequence, knowledge about how to solve specific SAPs is potentially lost from project to project.

  Yet, the procedure to solve an SAP always contains the same functions, and only some of the knowledge is problem specific. Furthermore, there are clear patterns that appear when working on different SAPs

- **Computational complexity**: SAPs are usually very "nasty" optimization problems that can be very hard to solve. Many tools lack the capability explore large tradespaces efficiently.

While this thesis does not address all of these problems completely, it is shown how the methodology that it proposed can help at least partially alleviate many of them.

1.6 Rule-based Expert systems

1.6.1 Definitions

An expert system is "a computer program designed to model the problem-solving ability of a human expert" (Durkin, 1994). In order to do that, an expert system uses large bodies of heuristic – expert – knowledge. In a rule-based expert system (RBES), expert knowledge is encapsulated in the form of logical rules. In the words of Feigenbaum, considered the major creator of the first expert system, these rules map the knowledge "over from its general form (first principles) to efficient special forms (cookbook recipes)" (Feigenbaum, Buchanan, & Lederberg, 1971). This is in opposition to other kinds of expert systems that primarily use different data structures to store expert knowledge, such as frames in frame-based expert systems (FBES), which are very similar to objects in object-oriented programming (Minsky, 1974).

In RBES, a logical rule is composed of a set of conditions in its left-hand side (LHS), and a set of actions in its right-hand side (RHS). The actions in the RHS are to be executed if the conditions in the LHS are all satisfied. An example of a logical rule is the following: LHS:="if the car won't start", RHS := "then check the electrical engine". An RBES infers information from rules in one of two ways: forward-chaining, when logical rules are used from the data to the goal in a deductive process; backward-chaining, when the rules are applied working backwards from a target goal (Buchanan & Shortliffe, 1984).

It has been noted that many expert systems, and ours is not an exception, are actual mixtures of RBES and FBES, as they use both rules and frames to represent the knowledge. We consider our expert system for assessing scientific value of EOSS an RBES, since the primary means – not the only means - of knowledge representation is that of logical rules.

For an excellent introduction to rule-based expert systems, see (Giarratano & Riley, n.d.).

1.6.2 A short history of rule-based systems

The first experiments with expert systems and RBES in particular were DENDRAL and MYCIN, both conducted at Stanford in the late sixties and early seventies in Buchanan and Feigenbaum's research group (Buchanan & Shortliffe, 1984), (Lindsay, Buchanan, & Feigenbaum, 1993). The DENDRAL experiment was a project in organic chemistry. DENDRAL was capable of inferring the molecular structure of a substance from its mass spectra and other experimental data. The MYCIN experiment leveraged from the early experience of DENDRAL, and focused on the diagnostic of the type of bacteria causing an infection in a patient, and the subsequent prescription of the right antibiotic, using fuzzy rules, i.e. rules with certainty factors. With 450 rules, MYCIN was able to perform better than senior doctors, and considerably better than junior doctors.

After DENDRAL and MYCIN, the development of RBES spread around the world, to many disciplines of the sciences and engineering. Duda et al developed the PROSPECTOR system, which was capable of identifying ore deposits (Duda, Gaschnig, & Hart, 1979). Woods et al developed the LUNAR systems, an RBES that answered geology questions about rock samples brought back from the Apollo missions (W. A. Woods, 1973). The early success of RBES allowed for their commercialization, starting with the R1 (later called XCON) system developed by McDermott at CMU to assist in the configuration of DEC's VAX systems using 2500 rules (McDermott, 1982). Clancey adapted the MYCIN program for teaching and tutoring and created NEOMYCIN and GUIDON (William J. Clancey, 1987), (W.J. Clancey, Letsinger, & Dept, 1984). Marsh and Healey developed several RBES for the NASA Johnson Space Center in the 1980's (NAVEX, RPMS, MCCSSES, MRDB) including applications to on-board navigation, electric power management, planning, requirements management, and monitoring radar data from the Space Shuttle in real time and estimate its position and velocity amongst others (Healey, 1986). Dincbas developed an RBES to design digital circuits (Dincbas, 1980). The field of architecture also embraced RBES as a means for automatic form synthesis using shape grammars, first developed at MIT by Stiny in the early seventies (Stiny & Gips, 1972).

The success of early systems such as MYCIN and DENDRAL made some experts think that AI and RBES had no limits. Then, the promises of life-changing technology of the late sixties and early seventies were wiped out by Lighthill in his infamous paper known as the Lighthill report. The Lighthill report gave a very pessimistic view of AI in general and RBES in particular, which led to funding cuts in many research labs in the UK and around the world (Lighthill, 1973). As a matter of fact, as Russell and Norvig point out that "to save embarrassment, a new field called IKBS (Intelligent Knowledge-Based Systems) was defined because Artificial Intelligence had been officially cancelled" (Russell, Stuart; Norvig, 1995). Thus, expert systems became knowledge-based systems (KBS), and the term knowledge engineering was coined to designate the process of developing a KBS.

It wasn't until the late 80's and early 90's, that KBS and RBES in particular started being used in decision making, as a decision support tool (Zimmermann, 1987). The number and types of RBES has grown continuously during the last 40 years, and RBES are present today in almost all science and engineering disciplines, education, finance, and architecture (Stiny, 1980).

We end this section by noting that the role of NASA, in particular of JSC, in the history of RBES is far from being negligible: DENDRAL, NAVEX, and LUNAR were both NASA-funded projects, and CLIPS was entirely developed at JSC.

1.6.3 Structure of RBES

RBES consist of three major elements: a fact database, a rule database, and an inference engine. The fact database contains relevant pieces of information about the specific problem at hand called facts. Information in facts is organized according to predetermined data structures similar to C structures and Java Beans, with properties and values (e.g. a fact of type car may contain a property make, a property model, and a property price, amongst others). These data structures are called templates in many RBES development tools. Facts can be asserted, modified, and retrieved from the database anytime. The rule database contains a set of logical rules that contain the domain knowledge. The LHS of these rules may match one or more facts in the working memory. The RHS of these rules define the actions to be executed for each of these matches, which typically include asserting new facts, modifying the matching facts, performing calculations, or showing some information to the user.

The inference engine performs three tasks in an infinite loop: a) pattern matching between the facts and the LHS of the rules in the working memory and creation of activation records (also known as the conflict set); b) while there remain activation records, select the next rule to execute, or fire (conflict resolution); c) execute the selected rules' RHS (deduction). Most current rule engines are based on the Rete algorithm, developed by Forgy in 1982 (Forgy, 1982). The Rete algorithm is faster than other algorithms because it "remembers" prior activation records in a network in memory called the Rete network. The Rete network is very efficient in speeding up the search process because most of the time, the network does not change much between iterations. Note that the improvement in computational time comes at the price of increased use of memory.

1.6.4 Critique of RBES

Reasons for criticism toward RBES can be grouped in four categories: a) problems in knowledge elicitation; b) problems in knowledge representation; c) problems in computational efficiency; d) problems in overall performance.

Each of these groups is succinctly developed in the next paragraphs. During the development of the early MYCIN and DENDRAL systems, it became obvious that a bottleneck in the development of RBES was the process of eliciting the knowledge from experts. One of the major challenges is the large quantity of tacit knowledge, which the expert is not aware that he or she has. Hence, the primitive view of KBS as a knowledge transfer process was challenged. Today, there is wide agreement that the process of building a KBS is a modeling process rather than a knowledge transfer process (Studer, 1998). A review of modern knowledge engineering techniques can be found in (Schreiber et al., 2000). While much progress has been made since then, the problem of eliciting expert knowledge is still a major challenge for the RBES developer.

In terms of knowledge representation, Newell and Simon postulated in their classic text "Human Problem Solving" that most human expert knowledge can be expressed in the form of logical rules (Newell & Simon, 1972). Earth science is not an exception to this. However, some knowledge is more naturally expressed using other more generic types of data structures such as frames (see for example Minsky's work (Minsky, 1974)). Related to this problem is the difficulty to guarantee the completeness and consistency of the rule base (Suwa, Scott, & Shortliffe, 1984).

In terms of computational efficiency, we mentioned before that the Rete algorithm is the most efficient inference algorithm for rule engines, and it generally improves the naïve approach of comparing all facts with all rules in working memory, which yields a time complexity of O(RFP), R being the number of rules, F the number of facts, and P the average number of clauses in the LHS of the rules. Indeed, the Rete algorithm achieves a time complexity of O(R'F'P'), where R', F', and P' are less than or equal to R, F, and P (Forgy, 1982). However, in the worst case, this still an exponential problem, which can lead to combinatorial explosion very fast in real-time applications.

Finally, in terms of overall performance, RBES have been successful when applied to relatively narrow domains, but have often failed in much larger applications (Studer, 1998). Also, expert systems in general are sometimes called weak methods in the AI community because they use "weak" (uncertain) information about the domain. Paraphrasing Russell and Norvig, "one might say that to solve a hard problem, you almost have to know the answer already" (Russell, Stuart; Norvig, 1995).

Our choice of RBES over other KBS architectures for this work, despite all the aforementioned limitations, was driven by superior characteristics in terms of scalability thanks to the separation between the knowledge (the rules) and the flow control (the inference engine), as well as their transparency achieved through their built-in explanation facilities, two attributes that we consider of outmost importance, and often overlooked by DST developers in system architecting.

1.6.5 The CLIPS language for developing rule-based systems

CLIPS (C-language integration production system) is a public language to write expert systems developed in 1985 at NASA Johnson Space Center as an alternative to the proprietary ART*inference (Riley, Culbert, Savely, & Lopez, 1987). Ten years later in 1995, Dr Friedman-Hill at Sandia National Labs developed an expert system shell in Java based on CLIPS, especially tailored for RBES (E. J. Friedman-Hill, 2000). As any other RBES, Jess deals with rules and facts, and its inference engine is based on the Rete algorithm. CLIPS/Jess syntax is very similar to common LISP, one of the earliest programming languages in artificial intelligence (Steele, 1990).

In CLIPS/Jess, the properties in templates are defined using the deftemplate command and properties are listed using the keywords `slot` (single element attributes) or `multislot` (multiple element attributes). Rules are defined using the `defrule` command. A fact is added into working memory, modified, or removed from working memory using the `assert`, `modify`, and `retract` commands respectively. Whenever the working memory is modified (e.g. a new fact is added or modified), the Rete network is recalculated. Matching rules are fired in the order determined by the Rete algorithm once the run command is sent. The general structure of the definitions of a template, a fact, and a rule are provided in Code 1. For further information on the Jess language, a good overview can be found in (E. Friedman-Hill, 2003).

```
(deftemplate my-template "Description of the template"
    (slot slot1-name) (slot slot2-name) (multislot multislot1-name) (multislot
multislot2-name)
)

(assert (my-template (slot1-name 1.34) (slot2-name high)
        (multislot1-name 1 2 5 12) (multislot2-name red green blue)))

(defrule my-rule "Description of the rule"
    ?this_fact <- (my-template (slot1-name ?x) (slot2-name ?y))
    (test (< ?x 10))
    =>
      (bind ?newx (+ ?x 1.0))
      (modify ?this_fact (slot1-name ?newx))
      (printout t "fact modified" crlf)
)
```

Code 1: Definition of a template, assertion of a fact, and definition of a rule in the CLIPS and Jess languages

Our RBES was developed in Jess rather than in other more common progamming languages for artificial intelligence such as LISP or PROLOG. The main reason for that is that RBES development tools such as CLIPS and Jess allow the user to focus on the application and forget about having to "reinvent the wheel" since much code that is needed to create an RBES is already available. Furthermore, Jess is written in Java, which facilitates the integration with the Matlab environment, very popular amongst both scientists and engineers.

1.7 Thesis Statement

Once RBES have been introduced, it is possible to provide solution-specific research goals that are consistent with the research gaps identified in the previous sections. These solution-specific goals are summarized in the following thesis statement.

To develop a knowledge-intensive framework that provides support to the system architect in all aspects related to solving their most common system architecting problems, namely representing, reasoning, simulating, and viewing, **by**:

1. Characterizing the classes of SAP;
2. Separating domain-independent knowledge from domain-specific knowledge;
3. Solving the domain-independent part of the problem for several classes of SAPs;
4. Developing a flexible and transparent approach to incorporate the domain-specific knowledge
5. Developing a methodology to assess architectural value in knowledge-intensive SAP with acceptable fidelity in limited time

Using rule-based expert systems

1.8 Structure of the thesis

The rest of this thesis is structured as follows.

Chapter 2 lays out the theoretical framework for rule-based system architecting, based on the idea of the separation of different bodies of knowledge. The first section focuses on the domain-independent knowledge that is common to all classes of SAPs. A generic tradespace search algorithm is implemented, based on a metaheuristic population-based search strategy. Four different types of rules are introduced that can be used to augment this generic algorithm with domain-specific and class-specific knowledge: grammars or enumeration rules, approximate evaluation rules, search heuristic rules, and selection rules. The rest of chapter 2 focuses on the domain-independent, SAP class-specific knowledge. Several classes of SAPs are identified based on experience and a literature review. For each class of SAPs, a grammar (i.e., an encoding scheme and a set of enumeration rules) is proposed that can be used for full or partial enumeration of the architectural tradespace. Search heuristic rules are also suggested to perform the actual search through each tradespace in an efficient manner.

Chapter 3 introduces a general methodology to handle approximate evaluation rules. The methodology is called **Value Assessment of System Architecting Using Rules**, or **VASSAR**. This framework analyzes a generic value delivery loop of a system architecture and models it using different sets of rules: attribute inheritance rules, capability rules, emergence rules, requirement satisfaction rules, value aggregation rules, and explanation rules. Each of these sets of rules is described in detail.

Chapter 4 introduces the domain of application of the theory developed in this thesis, namely Earth Observing Satellite Systems (EOSS). First, the main architectural decisions in EOSS are identified. Second, domain-specific knowledge required to architect and EOSS is presented in the form of rules (e.g., orbit selection, mass and power budgets). Third, the VASSAR methodology introduced in Chapter 2 is applied to EOSS. All the relevant templates and rules are described. Finally, the figures of merit used to evaluate different EOSS architectures are introduced. The chapter closes with a succinct introduction of the three case studies.

Chapters 5, 6, and 7 present the results of the three case studies, namely the NASA EOS case study, the Decadal Survey case study, and the Iridium GEOscan case study.

Finally, Chapter 8 summarizes the findings and contributions of the thesis, both on the methodology side and the applications, discusses the main limitations of the analysis, and highlights opportunities for future work.

2 Rule-based system architecting

Chapter 1 introduced the central problem of this thesis, which concerns the adaptation of current system architecting tools (SAT) to handle large quantities of expert knowledge. The main approach to solve this problem consists in separating the different bodies of knowledge required to solve a knowledge-intensive (KI) system architecting problem (SAP): domain-independent knowledge, domain-specific knowledge, and instance-specific knowledge.

The idea of knowledge separation is illustrated in Figure 9 for the particular case of EOSS.

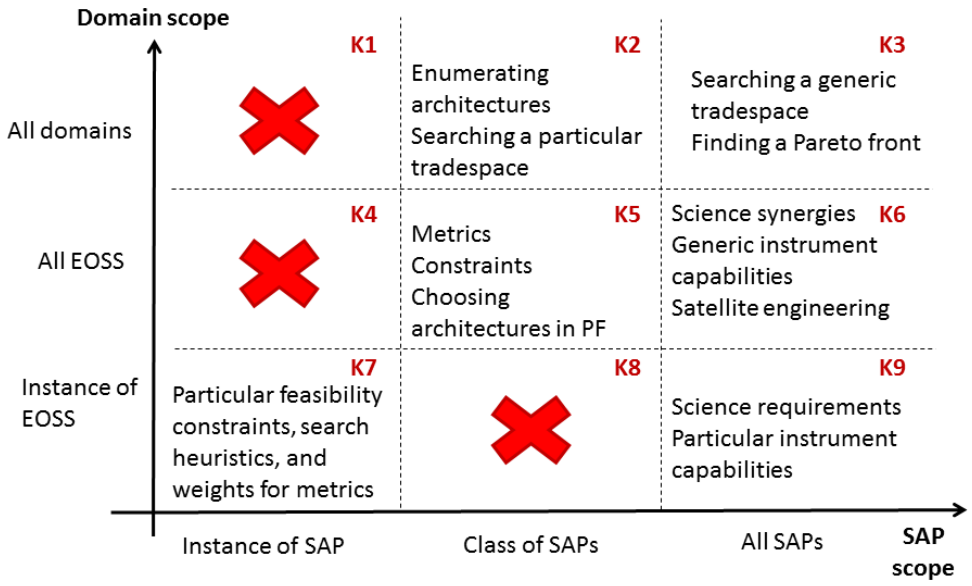


Figure 9: Knowledge separation in rule-based system architecting of EOSS

Knowledge is categorized in Figure 9 according to its scope in both the domain of application (e.g. EOSS) and the realm of search and optimization (labeled SAP). In terms of domain scope, knowledge may be completely independent of the domain, or common to all instances of problem in a particular domain, or really specific to a given instance of problem. In terms of SAP scope, knowledge can be common to all classes of SAPs, or common to a specific class of SAPs, or really specific of an instance of SAP.

These two categorizations are almost independent, but not completely, as indicated by the red crosses. Hence for example, a certain piece of knowledge can be common to all domains of application, but specific to a class of SAPs (K2); this is the case of the knowledge required to efficiently enumerate architectures over a set of permutations.

Conversely, another piece of knowledge may be specific to architecting EOSS, but common to all EOSS, and independent of the class of SAP; the radar equation, or how to do a power budget in a satellite, are examples of this case (K6). Finally, there will always be part of the knowledge that will be completely specific to the instance of EOSS, but some of it may be independent of the SAP (K9). For instance, the scientific requirements of the Decadal Survey belong to this category, as they are specific to the Decadal Survey, and at least part of them may be common to all SAPs considered (e.g., instrument selection, instrument packaging, mission scheduling).

The tool that enables the knowledge separation is the use of rule-based expert systems (RBES). The rest of this chapter describes in detail how the incorporation of an RBES brings about this knowledge separation.

2.1 Overview of the framework

In this section we introduce a framework that can be used to solve a KI SAP using RBES. The framework consists of the following main components:

- A **generic population-based heuristic algorithm** for searching the architectural tradespace. This tool takes care of the domain-independent knowledge that is also common to all classes of optimization problems. This algorithm is architected to trade some computational efficiency against an improvement in flexibility, scalability, and traceability. The algorithm utilizes four kinds of heuristic rules: grammars, approximate evaluation rules, search heuristics, and selection criteria.
- A **library of classes of SAPs** that contains efficient grammars, approximate evaluation rules, search heuristics, and selection criteria for several different classes of SAPs, which span the range of problems that appear more often in system architecting, and can benefit from a formulation as optimization or search problems. These classes of SAPs are directly deduced from the characterization of system architecting tasks done in Chapter 1.
- A general **methodology to assess architectural value** using approximate evaluation rules (VASSAR). This framework takes advantage of the performance of the pattern matching algorithm inside an RBES to perform a systematic comparison of system capabilities (facts) and system requirements (rules). Several other rules are defined to complete the framework, including emergence rules, which allow the description of emergent behavior in the system.

The general methodology to solve a KI SAP is illustrated in Figure 10. Figure 10 shows a hypothetical situation in which the initial problem is decomposed into three sub-problems, but in reality any number of sub-problems is possible. Each step of the methodology is described in more detail below.

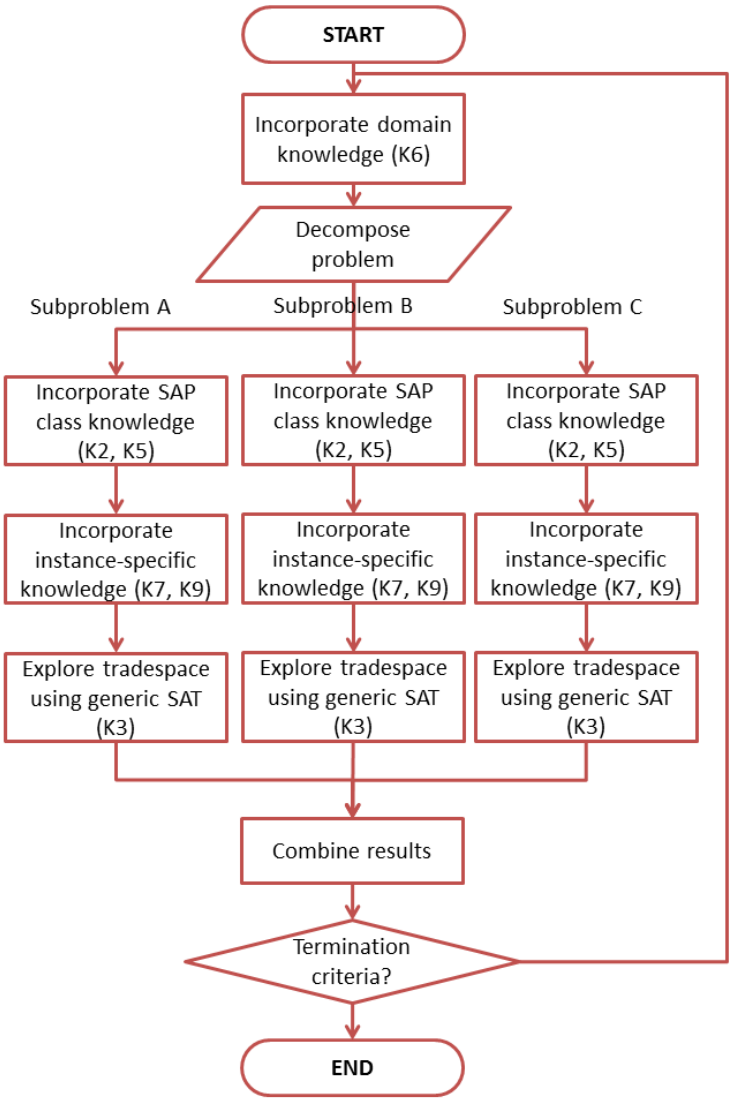


Figure 10: Methodology for solving knowledge-intensive problems in system architecting

Step 1: Incorporate domain-specific knowledge (K6): Before starting the actual architecting process, it will be necessary to have an initial database with domain-specific knowledge that is likely to be applicable to all SAPs. For example, in the case of EOSS, knowledge about generic instrument capabilities, data processing schemes, and satellite engineering may be relevant. Although this database will of course evolve with the state-of-the-art of science and technology, the speed of evolution may be relatively small in some domains of application. Moreover, part of the knowledge will encode physical or mathematical laws that are not subject to change. For example, in EOSS, a rule that computes the ground spatial resolution of an imager given its orbit and characteristics will remain valid. Ideally, this knowledge database would be maintained across different projects, so that knowledge can be reused from project to project.

Step 2: Decompose the problem into sub-problems. The step of the problem formulation is unchanged from the typical system architecting process. The initial problem is generally decomposed into a set of sub-problems. These sub-problems are sometimes called architectural views. An **architectural view** can be defined as a partial representation of a system architecture in which only a subset of its objects, processes, and relationships between them, are shown. For example, in the problem of architecting a resource exploration system, an architectural view may be concerned with the problem of platform allocation, a second view may look at the connectivity between platforms, and a third view may look at the production scheduling of the system. Each of these three views provides a partial description of the architecture of the system, i.e. it zeros in on a subset of architectural decisions to be made. A complete representation of the architecture is obtained when these multiple views are combined. Note that this decomposition, that is necessary due the inability of current tools to tackle to large tradespaces, will result in general in the loss of optimality in the search process. As a consequence, iteration will be required.

Step 3: Incorporate SAP class-specific knowledge (K2, K5). In step 3, the system architects will classify each sub-problem using the library of classes of SAPs described in this thesis. This classification describes several classes of SAPs that appear very often in the tasks of a system architect, and it will be described in detail later in this chapter. In this step, the system architect will recognize that the sub-problem at hand is an instance of one (or more) of the classes of problems defined in the library of SAPs, and will be able to reuse the rules provided in the library that capture knowledge relevant to that specific class of SAPs. For example, if a particular instance of problem is categorized as a set partitioning problem, the library provides a rule database that can perform automatic enumeration (full factorial and heuristic) of architectures. Note that this classification is not free of ambiguity. The same sub-problem can perhaps be formulated using several of the flagship problems because classes of SAPs are not mutually exclusive. In fact, it is well known in computer science that many classical combinatorial optimization problems can be reduced to instances of other classical combinatorial optimization problems. For instance, many scheduling problems can be reduced to instances of the traveling salesman problem, or any generalized assignment problem can be reduced to an instance of a transportation problem. In other words, it is impossible to define a hierarchy of complexity in the classes of SAPs in the library because they are all NP-hard problems.

Step 4: Incorporate instance-specific knowledge (K7, K9). In this step, the system architects can add the knowledge that is specific to the instance at hand. This knowledge can be common to all SAPs (K9), such as the capabilities of the instruments in an EOSS, or specific to a particular SAP (K7), such as particular constraints that apply to the NASA EOS packaging problem.

Step 5 Explore tradespace using generic search algorithm and tailored heuristics (K3). Once the SAP class-specific heuristics have been fed into the generic search algorithm (K3), the tradespace search process can start. This process is iterative in itself: the system architect runs a few generations of the algorithm, refines the model by adding or modifying rules as necessary (K7), and iterates until the quality of the results is satisfactory.

Step 6: Combine results. Once results are available for all the subproblems, it is necessary to combine the results to generate the complete architecture, since each subproblem only concerns part of the overall system architecture.

Step 6: Check termination criteria and iterate if required: As mentioned earlier, this is an iterative process. Therefore, at this stage, termination criteria must be checked. These termination criteria may include amongst others: a) a maximum number of generations; b) a maximum execution time; c) a minimum change in the best architecture or on the spread of the Pareto front in the case of several objectives; d) a minimum change in the metric or metrics of the best architecture or non-dominated architectures. If none of these criteria are satisfied, it is required to iterate by going back to Step 3 (or earlier if required). Revised, instance-specific heuristics can be utilized to further increase the performance of the search algorithm.

In the rest of this chapter, we describe in more detail the elements of this framework, namely the classification of SAPs, the KI heuristic search algorithm, and the process by which SAP class-specific heuristics are incorporated into the algorithm.

2.2 Classes of System Architecting Problems

2.2.1 Overview

In Chapter 1, we identified and described recurring tasks in the system architecting process that are susceptible of being formulated as SAPs. In this section, we take the next step by formulating these tasks as SAPs. A few **classes of SAPs** emerge from the analysis. For each class, we provide a formal definition and a few examples.

We start by recalling the recurring tasks of system architecting identified in Chapter 1: function to form mapping, decomposition or aggregation of either function or form, specialization of function or form, attribute selection, structural connectivity (typically between objects), scope selection, and the two types of commonality tasks: reactive and proactive.

It is also useful to list a few of the most common classical problems in combinatorial optimization. The following list is adapted from Gandibleux's classification of combinatorial optimization problems according to their combinatorial structure (Ehrgott & Gandibleux, 2000). For each problem, a reference is provided. Many of these problems have several hundred years, and were first introduced in mathematical contests or challenges of the time (e.g., Sir Hamilton's Icosian game introducing the traveling salesman problem in the early 1800's). Thus, the reference provided is always representative of the foundational work on methods to solve the problem, but not necessarily the first time the problem was introduced.

- **Assignment and generalized assignment problem** (Ross, G. T.; Soland, 1975): In the original assignment problem, there are a number of agents and a number of tasks, and each agent has a different cost to realize each task. The goal of the problem is to assign each task to exactly one agent, in such a way that all tasks are performed, and the total cost is minimized. In a generalized version of the assignment problem, agents can perform more than one task, and they have a given budget that cannot be exceeded. Furthermore, when performing a task, each agent produces a certain profit, and the goal of the problem is thus to maximize profit subject to not exceeding any agent budget.
- **Travelling salesman problem** (Dantzig & Fulkerson, 1954)**:** In the traveling salesman problem, there is a list of cities and a matrix containing their pairwise distances. The goal of the problem is to find the shortest path that passes through each city exactly once and returns to the city of origin.
- **0/1 knapsack problem** (Drexl, 1988): In the original knapsack problem, there is a list of items, each with a certain value and a certain weight. The goal of the problem is to determine the optimal number of items of each type to choose in order to maximize value for a certain maximum cost. In the 0/1 version of the same problem, the number of each item can only take the values {0,1}.
- **Set partitioning /set covering problem** (Garfinkel & Nemhauser, 1969): In the original set covering problem, there is a list of elements referred to as the universe, and a number of predetermined sets of elements, whose union is the universe. The goal of the set covering problem is to identify the minimum number of these predetermined subsets for which the union is still the universe. The set partitioning problem is a constrained version of the set covering problem where the selected subsets need to be disjoint or mutually exclusive. In other words, each element can only appear in one set in the set partitioning problem, while it may appear in more than one set in the set covering problem.

- **Job-shop scheduling** (Manne, 1960): In the simplest version of the job-shop scheduling problem, there is a list of jobs that needs to be assigned to a set of available resources (e.g., machines). Each job has a certain duration on each machine. The goal of the problem is to find the sequence of assignments of jobs to machines that minimizes the combined duration of the tasks.

  Variations of the problem include for example constraints between tasks (e.g., a task needs to occur before another task), costs for running the machines, or interactions between the machines (e.g., two machines cannot be running simultaneously).
- **Shortest path problem** (Dreyfus, 1969): In a shortest path problem, there is a graph defined by a list of nodes and a list of edges, and each edge has a distance associated to it. The goal of the problem is to find the path between two given nodes that minimizes the total distance.
- **Network flow problem** (Edmonds, 1972): In the most generic formulation of a network flow problem, there is a graph defined by a list of nodes and a list of edges. Nodes can be sources or sinks. Sources have a positive flow (supply) while sinks have a negative flow (demand). Each edge has a capacity associated to it, so that the flow through that edge cannot exceed that capacity. Each edge also has a cost associated to it. The goal of the network flow problem is to transport all the flow from the supply nodes to the demand nodes at minimum cost.
- **Minimum spanning tree** (Graham, 1985): In the minimum spanning tree problem, there is a graph defined by a list of nodes and a list of edges. Each edge has a cost associated to it. A spanning tree is defined as a subset of the edges that form a tree that contains all nodes. The goal of the problem is to find the spanning tree of minimum cost.
- **Maximum satisfiability (MAX-SAT) problem** (Hansen & Jaumard, 1990): In the maximum satisfiability problem, there is a set of Boolean variables, and a list of logical clauses that use that set of variables. The goal of the problem is to find the assignment to that set of Boolean variables that maximizes the number of clauses that are satisfied.

Most of the previous problems arise from graph theory and network analysis, and all of them are NP-complete problems[2]. Although this is by no means an exhaustive list of combinatorial optimization problems, it contains most of the relevant problems to system architects. Gandibleux also provided other classifications according to other criteria such as the "type of problem" (e.g., find the exact or approximate Pareto frontier, optimizing a utility function, finding a compromise solution).

[2] P is the class of problems for which we can find a solution in polynomial time. NP is the class of problems for which we can check a solution in polynomial time. NP-hard problems are at least as hard as the hardest problems in NP, i.e., any NP problem can be reduced to any NP-hard problem in polynomial time, but they do not necessarily belong to NP. NP-complete problems are simultaneously NP and NP-hard. This assumes that $P \neq NP$.

See his paper (Ehrgott & Gandibleux, 2000) for more details on combinatorial optimization problems, and methods to solve them.

From this list of combinatorial optimization problems, and the list of tasks discussed in the previous chapter, we identified five abstract problems that we names **classes of SAP**. A **class of SAP** is an abstraction of one or more of the aforementioned combinatorial optimization problems, formulated in a form that is relevant to the system architect.

In all these classes of SAPs, the variables are architectural decisions, and the goal of the problem is to find a set of *good* system architectures, where a good architecture is one that maximizes value delivery to stakeholders.

In particular, five classes of SAP were identified: down-selecting problems, covering and partitioning problems (considered as the same class), permuting problems, connecting problems, and assigning problems. In the **down-selecting class of SAPs**, there is a set of elements and the system architect must select a subset of these elements to constitute a system architecture. Down-selecting problems are motivated by the scope selection task, and they are similar in nature to the 0/1 knapsack problem.

In the **set partitioning and covering class of SAPs**, there is a set of elements and an architecture is characterized by a number of subsets of elements that define a set partition or a set covering of these elements. The class of set partitioning and covering SAPs is naturally motivated by the corresponding combinatorial optimization problems, and by the observation that several tasks of the system architect, such as the function to form mapping and the function or form aggregation, are naturally formulated using these problems.

In the **permuting class of SAPs**, the architectural decision to be made concerns the sequence or ordering in a set of elements. The permuting class of problems arises as a generalization of problems in which architectural decisions have a strong sequential component. This class includes in particular scheduling problems.

In the **connecting class of SAPs**, the system architecture is represented by a graph, where the elements are nodes, and the edges can be any type of interface between these elements. The goal of connecting problems is to find the set of edges or interfaces that maximizes value delivery to stakeholders. This class of SAPs is motivated by the structural connectivity task of the system architect, and it is closest in nature to the network flow problem.

Finally, the **assigning class of SAPs** is motivated by Simmon's representation of a system architecture as a set of decisions, where each decision has set of available options. This class of SAPs is closest to the generalized assignment problem.

All of these classes of SAPs are described in more detail in the rest of this chapter. The mapping between tasks of the system architect and classes of SAPs is presented in Table 1. Note that this is not a bijective (i.e., one-to-one) mapping, as: a) the same task can be formulated as different combinatorial optimization problems; b) some combinatorial optimization problems can be reduced to others (see discussion about NP completeness above).

For instance, the problem of mapping function to form can be formulated in a number of ways. If a certain group of objects and processes is fixed a priori, then each process simply needs to be assigned to one object. This formulation is similar to the assignment problem in operations research. In another formulation, there may not be a predefined set of objects, and elements of form may be implicitly defined by groups of processes performed by the same abstract element of form. This alternative formulation is much closer to a set partitioning problem.

Table 1: Correspondence between system architecting tasks and classes of SAPs.

	Down-selecting Problems	**Covering/ Partitioning problems**	**Permuting problems**	**Connecting problems**	**Assigning Problems**
Scope selecting	Preferred				Possible
Function to form mapping	Possible	Possible			Preferred
Decomposition /aggregation		Preferred		Possible	Possible
Specialization	Possible				Preferred
Attribute selection	Possible				Preferred
Structural connectivity		Possible	Possible	Preferred	Possible
Reactive commonality				Possible	Possible
Proactive commonality				Possible	Possible

Furthermore, as mentioned earlier, some classes of SAP are variations or generalizations of one or more combinatorial optimization problems. For instance, down-selecting problems are similar to the 0/1 knapsack problem; permuting problems are similar to the traveling salesman problem and to the job scheduling problem. The similarities and differences between the classes of SAP and their corresponding combinatorial optimization problems are discussed in the following sections. Generally speaking**, the classes of SAPs are broader in scope than the corresponding optimization problems, more constrained, and they try to capture better interactions between elements, because these interactions are the origin of emergent behavior, which is important in system architecting.**

In the rest of this section, each class of SAPs is described in detail. For each class of SAP, we provide a formal definition, a few examples, and a discussion of the differences between the class of SAP and the corresponding combinatorial optimization problems. **Class-specific rules are provided in an appendix**.

2.2.2 Assigning Problems

2.2.2.1 Definition

The assigning problem is a very general class of SAPs given in the form of a set of decisions that need to be made. Each decision has a set of options, and the goal of the problem is to assign an option to each decision in an optimal way.

Definition (Assigning Problem): Given a set of *n* generic decisions $D = \{d_1, d_2, \ldots, d_n\}$, where each decision d_i has a discrete set of options $O_i = \{d_{ij}\}_j$, the goal of the Assigning Problem (AsP) is to find the assignment of options to decisions $A = \{d_i \leftarrow d_{ij}\}_i$ that maximizes value delivery to stakeholders:

$$A^* = arg \max_{\{d_i \leftarrow d_{ij}\}_i} V(\{d_i \leftarrow d_{ij}\}_i)$$

where $V(\cdot)$ is a generic value property function that may have more than one dimension (e.g. benefit and cost). An architecture in the AsP is represented by an assignment $A = \{d_i \leftarrow d_{ij}\}_i$. Note that, in general, the solution to the AsP is not unique, but rather a set of non-dominated architectures, due to its multi-objective nature. Furthermore, note that the set of options for each decision is a discrete set, and therefore if a certain decision can be defined in a continuous domain, it is necessary to provide either a finite set of acceptable values, or the boundaries and a discretizing step to construct this set. For example, if a decision has to be made concerning the length of a certain element, since this length could in principle take any real value from 0 to infinity, it is required to provide either a set of acceptable values (e.g., {1.2, 2.4, 3.6}), or a minimum and maximum length and a discretizing step (e.g., 1 to 5, in steps of 0.5).

A pictorial representation of a generic AsP is provided in Figure 11 where three different decisions are shown with three, two, and four options respectively.

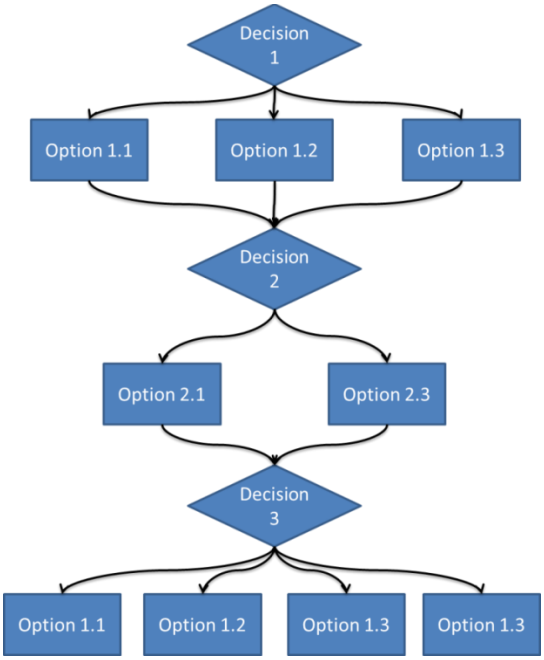


Figure 11: The assigning class of architectural problems

Size of the tradespace: The size of the tradespace of a generic AsP is simply given by the product of the dimensionalities of the option sets of each decision:

$$|\{A_k\}| = \prod_{i=1}^{n} |O_i| = \prod_{i=1}^{n} |\{d_{ij}\}_j|$$

Hence in the example of Figure 11, there are 3*2*4=24 different architectures. Note that if we introduce the geometric average of the dimensionalities of the option sets $\overline{|O_i|} \equiv \sqrt[1/n]{\prod_{i=1}^{n} |O_i|}$, then the size of the tradespace becomes a simple exponential:

$$|\{A_k\}| = \prod_{i=1}^{n} |O_i| = \overline{|O_i|}^n$$

Thus, the size of the tradespace of the AsP grows very quickly with the number of decisions, especially when the average number of options is large. This is typically the case when continuous variables are discretized with a small quantum.

2.2.2.2 Examples

Previous work by Simmons has demonstrated that the system architecting process can be modeled as a set of interrelated decision making problems (Simmons, 2008). In other words, architecting a system is equivalent to making architectural decisions. As a consequence, almost any SAP can be put in the form of an AsP.

- **Example 1** (adapted from (Aliakbargolkar, Wicht, Battat, Calandrelli, & Crawley, 2011)): When architecting a 2 ½ stage launch vehicle, the major architectural decisions are: the number of boosters, the propellant type of each stage, the diameter, and the length of each stage. Each of these decisions has a set of options: for instance, the number of boosters can be 1, 2, or 4; the propellant type can be "solid", "LH2", "RP-1", or "CH4" (it can be different for each stage); the common diameter can range from 3m to 8m in steps of 1m; and the lengths of each stage can range from 1m to 50m in steps of 1m.

  Note that there are eight decisions to make, with 3, 4, 4, 4, 6, 50, 50, and 50 options respectively. With these decisions and options, there are 3*4*4*4*6*50*50*50=144 millions of possible architectures. Note that not all of these architectures are feasible or even make sense. For instance, solid propellant will probably only be chosen for the boosters. Also, geometrical constraints such as minimum and maximum aspect ratios will reduce the size of the tradespace.
- **Example 2**: Let us consider the architecture of an altimetry satellite mission. We can model the system architecting process as a decision making problem with three decisions: composition of the payload, orbit, and number of satellites. For example, the payload can be any combination of the following instruments: a next-generation wide-swath radar altimeter, a classical nadir altimeter, a microwave radiometer, a GPS receiver, a DORIS receiver, and a passive optical imager, i.e. a total of 2^6-1 = 63 options if we don't count the empty set as a valid combination; the orbit can be a classical oceanography orbit (non SSO, 1300km, 65 deg inclination), or an 800km dawn-dusk SSO that would ease the requirements on the power subsystem, improve illuminating conditions, but suffer from aliasing due to poor sampling of the diurnal cycle; finally, we may want to compare a single-satellite mission with a tandem (two satellite) mission, and a constellation with 4 satellites evenly spaced in a single plane. In this example, we have three decisions with 63, 2, and 3 options respectively. Thus, the size of the tradespace is 63*2*3 = 378 architectures.

2.2.2.3 Discussion

The AsP is particular for several reasons. First, all the other classes of SAPs can be reduced to an AsP as it will be shown later. Perhaps due to this, the AsP formulation was chosen both by Koo and Simmons for their SATs (B. H. Y. Koo, Simmons, & Crawley, 2009), (Simmons, 2008). Although it is true that it is an extremely generic and powerful class of problems and many problems in system architecture can be formulated as an AsP, we claim in this thesis that there are other classes of SAP that appear often in system architecting that are not naturally represented using an AsP formulation.

The AsP is similar in nature to the class of assignment problems, where there is a set of tasks that needs to be assigned to a set of agents. However, there are several particularities of the AsP that make it different from typical assignment problems. In the standard assignment problem in operations research, the goal is to find the optimal bijection between the task and agent sets, i.e it is a one-to-one assignment. In the AsP, the sizes of the two sets can be different. This modified version of the assignment problem is sometimes called the *weapon target assignment problem*. Furthermore, the objective functions are non-linear and potentially knowledge-intensive functions of the assignment. This precludes the direct utilization of linear programming techniques.

2.2.3 Partitioning and Covering Problems

2.2.3.1 Definition

Definition (Set Partition): Given a set of *m* generic elements of function or form $U = \{e_1, e_2, \ldots, e_m\}$, a partition $P_i = \{S_i\}$ of a set U is a division of U into a number of non-overlapping subsets S_i that are mutually exclusive and exhaustive. In other words, we call $\{S_i\}$ a set partition if:

1. The union of all elements in $\{S_i\}$ is equal to U: $\bigcup_i S_i = U$ [Constraint PaP-1]
2. The intersection of all elements in$\{S_i\}$ is empty: $\bigcap_i S_i = \emptyset$ [Constraint PaP-2]

Definition (Partitioning Problem): Given a set of *m* generic elements of function or form $U = \{e_1, e_2, \ldots, e_m\}$, the goal of the Set Partitioning Problem (PaP) is to find the set partition $P_i = \{S_i\}$ that maximizes value delivery to stakeholders:

$$\{Si\}^* = arg \max_{Pi = \{Si\}} V(\{Si\})$$

where $V(\cdot)$ is a generic value property function that may have more than one dimension (e.g. benefit and cost). An architecture in the PaP is represented by a partition $P_i = \{S_i\}$.

Note that, in general, the solution to the SSP is not unique but rather a set of non-dominated architectures. A pictorial representation of a generic PaP is provided in Figure 12, where two different set partitions are provided. The partition on the left side divides the eight elements into four subsets, while the partition on the right side divides the eight elements into two subsets.

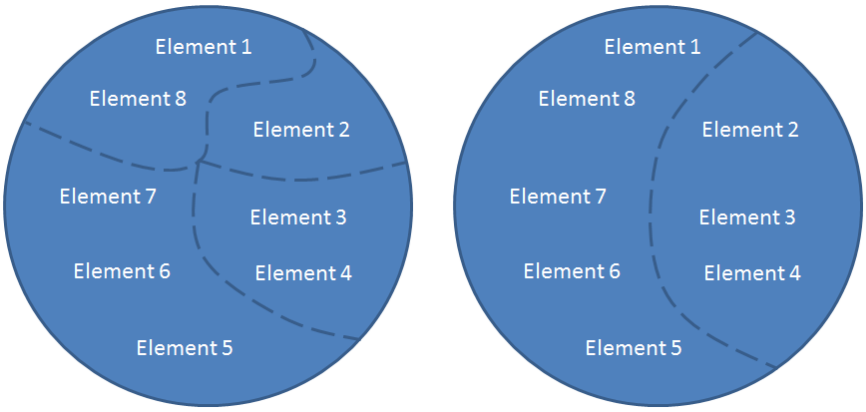


Figure 12: The partitioning class of architectural problems

Size of the tradespace: The number of partitions in a set of m elements with exactly k subsets is given by the Stirling numbers of the second kind, which can be computed using the following explicit formula:

$$\left\{ {m \atop k} \right\} = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^m$$

The total number of set partitions is equal to the sum of all the Stirling numbers of the second kind for all possible values of k.

$$B(m) = \sum_{k=0}^{m} \left\{ {m \atop k} \right\}$$

These numbers are called by Bell numbers, and can be computed for example using the following recurrence:

$$B(0) = 1; B(1) = 1; B(m+1) = \sum_{k=0}^{m} \binom{m}{k} B(k) \;\; \forall m \geq 2$$

The first Bell numbers are thus 1, 1, 2, 5, 15, 52, 203, 877… The first 500 Bell numbers may be found on http://oeis.org/A000110.

Definition (Covering Problem): A variant of the PaP is obtained if Constraint PaP-2 is relaxed, allowing for certain elements to be assigned to more than one subset. For example, if elements are functions, we can allow for a certain function to be fulfilled by more than one element of form (i.e. we can relax Constraint PaP-2), but every function still needs to be fulfilled by at least one element of form (i.e., Constraint PaP-1 cannot be relaxed). This variant is called the set covering problem (CvP).

Both PaPs and CvPs are well known NP-hard problems that appear very often in computer science, operations research, and other disciplines. For instance, the vertex cover problem, the map coloring problem, and the airline crew scheduling problem can all be formulated as instances of PaPs or CvPs (Levine, 1994).

2.2.3.2 Examples

In system architecting PaPs and CvPs, there are different situations where the way in which elements are partitioned affects architectural value. The following are examples of PaPs and CvPs in system architecting:

- **Example 1**: Fractionated spacecraft are alternative spacecraft architectures in which the different functions of a spacecraft are performed by elements that are physically separated from each other, as opposed to a classical monolithic architecture where the bus consist in a single physical element.

  Functions such as communications, payload, or even electrical power generation and distribution, can be performed by smaller independent spacecraft flying in close formation. Fractionated spacecraft promise better emergent system properties (e.g., flexibility, survivability, scalability) than monolithic architectures, albeit at a higher cost and technical risk given the current state-of-the-art of spacecraft technology[3]. The problem of exploring different fractionated spacecraft architectures can be modeled as a PaP in which each architecture is a different partition of functions into subsets that are independent units.
- **Example 2**: When architecting Earth observing programs, program managers often face the question whether to distribute a given set of instruments in a single satellite or on several smaller satellites. More generally, given a set of N instruments or payloads, they can be flown on any number of satellites between 1 and N.

[3] See for example (O. C. Brown, Eremenko, & Collopy, 2009; O. Brown & Eremenko, 2008) for a good discussion of fractionated spacecraft.

The problem of finding the best allocation of instruments into satellites can also be modeled as a PaP. Note that in this case the elements being partitioned are elements of form (instruments), whereas in Example 1, the elements being partitioned where functions (power generation, communications). If multiple copies of an instrument can be flown on different spacecraft, then the problem becomes a CvP.

- **Example 3**: In the context of a space exploration mission, different architectures may use a different number of habitable volumes to satisfy the same overall set of functional requirements. For instance, in Apollo a lunar module was developed to bring the astronauts from Moon orbit to the Moon's surface and back. Another architecture could have reused the orbiting spacecraft as lunar capsule, thus putting the two sets of requirements on a single capsule. This architectural decision affects value in many ways. For instance, the cost of developing two capsules will be different than the cost of developing one larger capsule. Moreover, competing requirements in the two sets may lead to necessity to reach compromise solutions.

2.2.3.3 Discussion

Set partitioning and set covering problems are classical problems in combinatorial optimization. In the original formulation, the goal of the PaP/CvP was to draw from a predetermined list of subsets of elements so that the PaP/CvP becomes a knapsack problem where the cost function is simply the number of subsets. Our PaP is a more general version of this problem in which the metrics are complex functions of the set partitioning. These metrics may include for instance cost models, and fine modeling of the interactions between elements.

Interactions between elements (both positive and negative interactions) play a very important role in the PaP/CvP. In this aspect, it is closer to the class of clustering problems rather than the classical PaP/CvP formulation in operations research. These issues will be discussed in more detail in Section 9.1.2.

2.2.4 Down-selecting Problems

2.2.4.1 Definition

Definition (Down-selecting problem): Given a set of *m* generic elements of function or form $U = \{e_1, e_2, \ldots, e_m\}$, the goal of the down-selecting problem or DsP is to find the subset S_i or subsets of elements $\{S_i\}$ that maximizes value delivery to stakeholders, or more precisely, that optimizes the trade-off between benefit and cost.

$$S_i^* = arg\max_{S_i} [B(S_i) \quad C(S_i)]$$

where $B(\cdot)$ and $C(\cdot)$ are benefit and cost property functions respectively. An architecture in the DsP is represented by a subset S_i. Note that, in general, the solution to the DsP is not unique but rather a set of non-dominated architectures.

The pictorial representation of a DsP is provided in Figure 13.

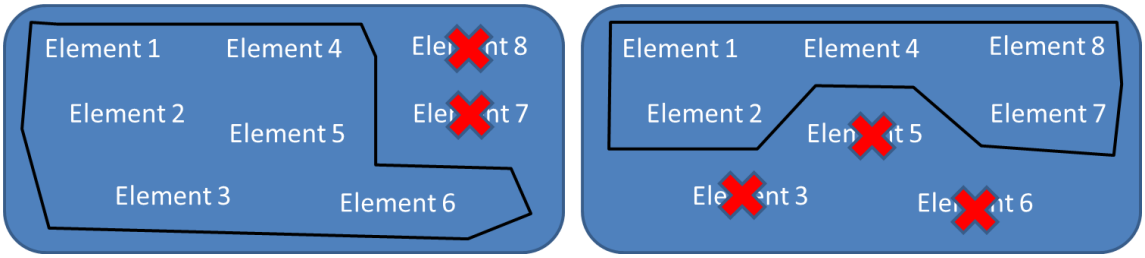


Figure 13: The down-selecting class of architectural problems

Size of the tradespace: The size of the unconstrained architectural tradespace of a DsP is exponential with m.

$$|\{S_i\}| = 2^m$$

Mathematically, the DsP is similar to the classical 0/1 knapsack problem, with the following exceptions:

1. Values of elements are not additive. The value of an element depends on the other elements in the subset. There are positive interactions between elements leading to emergent positive value, and negative interactions between elements leading to emergent negative value.
2. In general, there are multiple constraints in element selection. For example, a constraint may say that if element e_i is chosen, then element e_j must also be chosen; another constraint may say that if element e_i is chosen, then element e_j it cannot be chosen.

It is important to note that exception 1 precludes in particular a classical dynamic programming formulation with recurrent independent decisions because the Bellman equation is not met due to dependence of present reward on both past and future choices of elements. For example, the last element selected may change the value of an element that was previously selected. Concerning exception 2, there has been some work on multi-constrained versions of the knapsack problem (Drexl, 1988), in particular with neighboring constraints (Borradaile, Heeringa, & Wilfong, 2009), but surprisingly we found no previous work concerning a version of the knapsack problem where both exceptions 1 and 2 are considered.

2.2.4.2 Examples

DSPs appear naturally in any real life situation where one has a limited budget to choose between a set of candidate assets with costs and rewards.

For instance, the chief technologist at NASA has a given budget and a list of proposals for technology investments, and he or she has to choose a subset of these proposals that maximizes return for the given budget. This is an instance of a DSP, and note the importance of exceptions 1 and 2 in this example, which make the formulation of this problem as a classical knapsack problem a bad idea. For instance, in the context of human exploration, investing on a technology for boil-off control will affect the value of investing on different types of propellant (e.g., CH_4 versus LH_2).

In Earth observing programs, there is sometimes a phase of instrument selection, where a list of candidate instruments is available as well as a fix budget for instrument and mission development. The goal in this case is to maximize science and societal return by selecting the best and most complementary and synergistic set of instruments to be flown in the program. Again, exceptions 1 and 2 are very important. Particularly, it is possible that one has to choose between several instruments for a specific measurement (e.g. choose between two lidars where one is a more advanced version of the other, or between two equivalent instruments where one is developed nationally and the other one internationally.

2.2.4.3 Discussion

The DSP is similar in nature to knapsack problems. In a general knapsack problem, we are given a set of items, each of them with a given cost and benefit. The goal is to find the number of items of each type that maximizes value at a given cost.

In the 0/1 knapsack problem, it is not allowed to have multiple copies of the same item. In both these formulations, the benefit and cost of the items are independent of the other items selected. In other words, there are no interactions between the elements. In reality, the value of selecting a certain item will depend on the other items selected because there are synergies, redundancies, and incompatibilities between elements. Let us consider a trivial example of a rucksack, and a set of available items that contains, amongst others, three tubes of toothpaste of different brands with benefits B_1, B_2, B_3, a toothbrush (B_4), a sandwich (B_5), a bottle of water (B_6), a towel (B_7), and soap (B_8). In a classical formulation, all the B_i would be constant, and thus the benefit of choosing the three tubes of toothpaste would be $B_1 + B_2 + B_3$. In reality, the benefit of having the three tubes of toothpaste will arguably be much smaller than $B_1 + B_2 + B_3$, due to redundancy between the items. Similarly, in the classical formulation, the value of having the toothpaste will not depend on whether the toothbrush is selected or not.

In reality, the value of having toothpaste without a toothbrush is much smaller. This is the effect of emergence, or synergies, which as mentioned before is very important to capture. This discussion is continued in more detail in Section 9.1.3.

2.2.5 Permuting Problems

2.2.5.1 Definition

Definition (set permutation): Given a set of m generic elements of function or form $U = \{e_1, e_2, \dots, e_m\}$, a permutation O_i is any bijection of U onto itself:

$$O_i: U \to U$$

Informally, O_i is any arrangement of the elements in U into a particular order. For example, if we consider a set of four generic elements $U = \{e_1, e_2, e_3, e_4\}$, one possible permutation is $O_1 = \{e_1, e_3, e_4, e_2\}$, and another possible permutation is $O_1 = \{e_1, e_3, e_4, e_2\}$.

Definition (Permuting problem): Given a set of m generic elements of function or form $U = \{e_1, e_2, \dots, e_m\}$, the goal of the permuting problem or PeP is to find the permutation or ordering of U that maximizes value delivery to stakeholders.

$$O_i^* = arg\max_{O_i} V(O_i)$$

where $V(\cdot)$ is a generic value property function that can have one or more dimensions. An architecture in the PeP is thus represented by a permutation O_i. Note that, in general, the solution to the PeP is not unique but rather a set of non-dominated architectures. The pictorial representation of a PeP is provided in Figure 14.

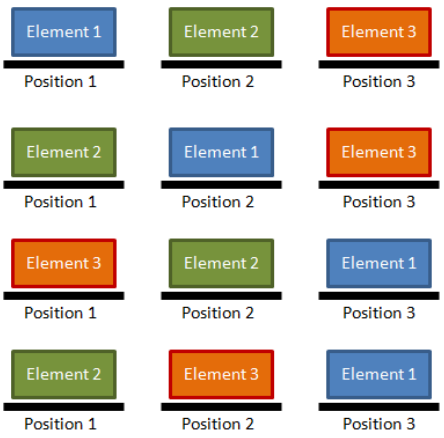


Figure 14: The permuting class of architectural problems

Tradespace size: The size of the tradespace of a PeP is given by the size the set of all possible permutations of a set of m elements:

$$|O_i| = m!$$

2.2.5.2 Examples

PePs appear often in systems architecting, both with function and form elements. When PePs concern the sequence of a set of processes or events, they are typically instances of classical scheduling or planning problems. Examples of these include:

- To find the optimal strategy for the staged deployment of a flexible system architecture
- To find the optimal sequence of destinations in a human space exploration program
- To find the optimal scheduling of a set of Earth observing missions given a certain annual budget, mission costs, and mission societal and scientific benefits.

One may also encounter a few instances of PeP that deal with objects instead of processes. In these cases, the problem is usually about structural connectivity and interfaces between objects.

2.2.5.3 Discussion

While our definition of the PeP may look similar in some aspects to the traveling salesman problem (TSP), algorithms used to solve the TSP will rarely be applicable to PePs in general. In the TSP, we are given a list of cities and pairwise distances between them, and the goal is to find the optimal permutation of visits that minimizes overall distance.

In the PePs, these pairwise distance between cities *i* and *j* could represent the value of selecting *j* as the next element given that our last choice was *i*. However, in this formulation, it is implicitly assumed that the value of selecting an element only depends on the last element selected. In reality, the value of selecting an element depends on all the elements selected before.

More generally, the PeP can be defined as the problem of optimizing over a permutation set. Some work has been done on this topic. Silvio Turrini studied a strategy based on transformation of the permutation into a transformed domain in which a notion of distance between elements can be defined. Efficient hill climbing algorithms can then be applied in the transformed domain, using a Lagrangian formulation with penalties in the objective function to handle infeasibility constraints (Turrini, 1996). This and other possible strategies to tackle PePs are discussed in Section 9.1.4.

2.2.6 Connecting Problems

2.2.6.1 Definition

Definition (Graph): Given a set of *m* generic elements of function or form $U = \{e_1, e_2, \dots, e_m\}$, a graph G_i is a representation of the interrelationships between the elements in the set in which the elements are represented by *nodes* or *vertices*, and their links are represented by *arcs* or *edges*. A graph is thus fully defined by a 2-tuple:

$$G = \langle V, E \rangle$$

where V is a set of vertices, E is a set of edges, and each edge is represented by a pair of vertices.

Definition (Connecting problem): Given a set of *m* generic elements or nodes $U = \{e_1, e_2, \dots, e_m\}$, the goal of the connecting problem or CnP is to find the graph that has U as its set of vertices and maximizes value delivery to stakeholders.

$$G_i^* = arg\max_{G_i} V(G_i)$$

where $V(\cdot)$ is a generic value property function that can have one or more dimensions. An architecture in the CnP is thus represented by a graph G_i. Since all graphs considered in the CnP have the same set of vertices $G = \langle U, E \rangle$, a more succinct definition of the problem is to optimize over the set of edges:

$$E_i^* = arg\max_{E_i} V(E_i)$$

Note that, in general, the solution to the CnP is not unique but rather a set of non-dominated architectures. The pictorial representation of the CnP is provided in Figure 15.

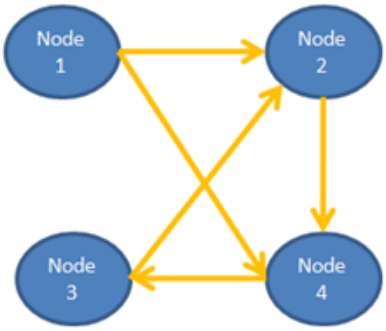


Figure 15: The connecting class of architectural problems

Given a graph $G = \langle U, E \rangle$, the **adjacency matrix** is a data structure that represents the connectivity between the nodes in the graph. In other words, it is a matrix representation of the information contained in E. More precisely, the adjacency matrix A of a graph G is an n x n matrix where $A(i,j) = 1$ if node *i* is adjacent (i.e., directly connected by an edge) to node *j*, and $A(i,j) = 0$ otherwise. Note that in undirected graphs, i.e., graphs in which edges have no orientation, $A(i,j) = 1$ implies $A(j,i) = 1$, and therefore the adjacency matrix of undirected graphs is symmetric. This is not the case of directed graphs.

Tradespace size: The size of the tradespace of a CnP is given by the size the set of all possible adjacency matrices of a set of m elements. The number of different adjacency matrices depends on whether the graphs being considered are undirected or directed, and whether the nodes are allowed to be connected to themselves. This leaves four different cases that are summarized in Table 2:

Table 2: Size of the tradespace for the CnP in four different cases

	Directed graph (non-symmetric)	Undirected graph (symmetric)
Self-connections allowed (diagonal meaningful)	$\lvert\{G_i\}\rvert = 2^{m^2}$	$\lvert\{G_i\}\rvert = 2^{\frac{m(m+1)}{2}}$
Self-connections not allowed (diagonal not meaningful)	$\lvert\{G_i\}\rvert = 2^{m^2-m} = 2^{m(m-1)}$	$\lvert\{G_i\}\rvert = 2^{\frac{m(m-1)}{2}}$

The formulae above assume that there can be a maximum of one connection between two any given nodes or between a node and itself. In other words, it assumes the adjacency matrix is a Boolean matrix.

2.2.6.2 Examples

CnPs are amongst the most prevalent problems in system architecture. Examples of CnPs include:

- To architect a resource exploration system with several reservoirs and platforms (Aliakbargolkar & Crawley, 2012).
- To architect the power network of a small country.
- To architect a Guidance, Navigation, and Control (GN&C) subsystem for the International Space Station using a set of sensors, computers, and actuators that can be connected between them (Dominguez-Garcia et al., 2007).

More generally, many systems-of-systems can be modeled as a graph where the vertices are the individual systems and the edges represent inter-system interfaces.

All the aforementioned examples essentially deal with the structural arrangement of elements of form in the system, i.e., the definition of physical interfaces between system elements. CnPs can also be used to model non-physical interfaces, such as a sequence of decisions over time.

2.2.6.3 Discussion

The CnP is more closely related to the body of network optimization. There is a broad variety of optimization problems in network analysis including for example maximum flow problems, shortest path problems, minimum spanning three problems, transportation problems, routing problems. The most general network optimization problem is the minimum cost network flow problem. In the minimum cost network flow problem, we consider a network with *n* nodes, where each node has a net flow that can be either positive (supply), negative (demand), or zero (transshipment). We also have a predefined set of edges between the nodes, where each edge has an associated cost. The goal of this problem is to minimize the cost of sending the supply through the network to satisfy demand.

All of these classical network optimization problems are of particular interest because they can be solved much faster than linear programs. However, the most general expression of our CnP does not have this problem structure because both cost and value may emerge from interactions between nodes. For instance, reliability is a system property that emerges from overall network connectivity. As a consequence, efficient network algorithms will generally not apply.

2.2.7 Summary and discussion

In this section we have introduced five classes of SAP, namely assigning problems, partitioning and covering problems, down-selecting problems, permuting problems, and connecting problems, which can be used to formulate the recurring tasks of the system architect identified in Chapter 1.

The reader may be surprised by the absence of a satisfiability class of problems. While it is true that many SAPs can be reduced to a satisfiability problem, and that their formulation as satisfiability problems may be beneficial from the computational standpoint, it is often not convenient from the representational and viewing standpoints.

Hence, instead of having a class of problems about satisfiability problems, it is more desirable to have a set of rules that solve a certain class of SAPs by transforming them into satisfiability problems. This is similar to the work by Rayside, Estler and Jackson (Rayside et al., 2009), in which an ADG-like interface is provided to the user, but the actual SAP is internally solved by transforming the original problem into a MAX-SAT problem and using efficient tools for solving SAT problems.

As in any other classification scheme, one may question the completeness and degradedness of this classification. In other words: a) given any SAP, does it always belong to at least one class of SAP? b) given any SAP, does it always belong to at most one class of SAP? Related to this idea is the fact that the process of classifying an SAP is human and therefore implicitly subjective. How hard is it to classify an instance of a problem? How do we choose the appropriate formulation in the case where two classes are clearly possible? While these are valid concerns, we argue that having this "library" of classes of SAPs will only help the system architects make this classification that they would have done in their minds anyway, since solving the problem requires formulating it as a combinatorial optimization problem. Furthermore, we have tried to support the system architects in their choice by providing pictorial representations of each class of problems, as well as several real-life examples that they can use to compare to their own instance.

2.3 A knowledge-intensive heuristic algorithm for searching the architectural tradespace

2.3.1 Overview

This section focuses on the domain-independent knowledge that is also independent of the class of SAP, as illustrated in Figure 16.

Up to this point, we have identified the tasks that the system architect recurrently performs, and we have shown that when formulating these tasks as SAPs, there are patterns that appear.We defined five classes of SAPs from these patterns in the previous section, and provided examples for each class.

When solving these SAPs, a variety of combinatorial optimization algorithms or strategies can be applied, and the success or failure of these strategies will in general be dependent on the specific problem at hand (Ehrgott & Gandibleux, 2000). In other words, we cannot design a strategy that will solve all classes of SAPs at maximum efficiency. This is sometimes known as the **no free lunch theorem in search and optimization** (Wolpert, 1997). Rather, the goal of this research is to develop a framework in which the parts of the problem that are always the same are solved, while providing flexibility to accommodate domain-independent and domain-specific knowledge easily.

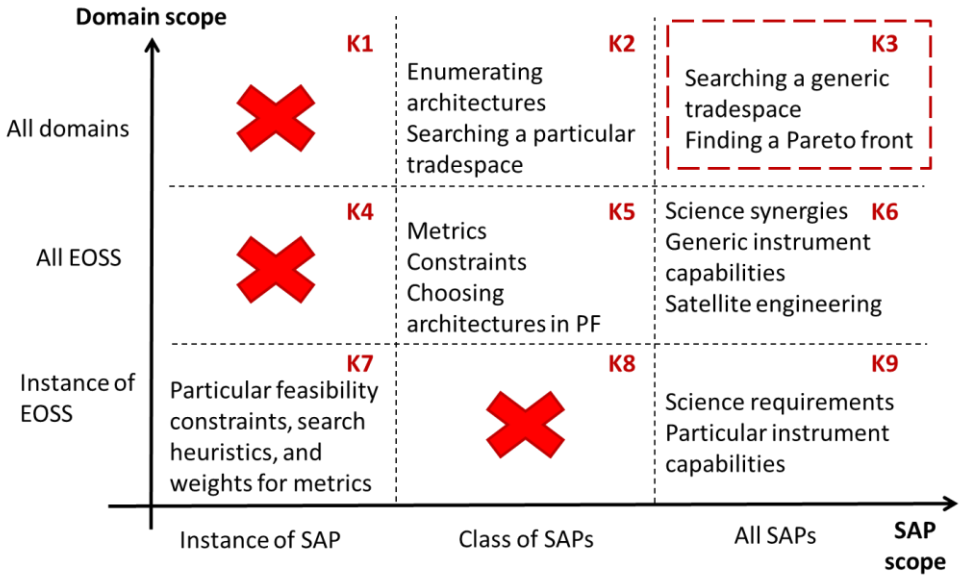


Figure 16: Situation of the generic search algorithm in the knowledge separation chart

Indeed, there are a certain number of tasks that are always the same, namely: *encoding*, i.e. the creation of an enumerable model of the system architecture, *enumeration* of architectures using this model, *evaluation* of architectures using a set of metrics, *search*, i.e. the iterative process of choosing a "direction" in the tradespace to look for new and potentially better architectures, and *down-selection* of a subset of preferred architectures. In this section, we introduce a generic search algorithm based on the incorporation of an RBES, which performs all these functions iteratively while allowing the introduction of a large body of knowledge in the form of rules.

More precisely, we selected a generic **population-based** tradespace search strategy for the framework. The reasons that led to the selection of a population-based algorithm, as well as the steps of this algorithm, are described in more detail in the next subsection.

The customization of this generic search algorithm is accomplished through the incorporation of domain-independent and domain-specific knowledge in the form of rules. These rules make the search process for each class or instance of an SAP much more efficient. Hence, the knowledge that goes into the implementation of the different modules of this algorithm is different for each class of SAP, but the overall strategy is not. Note in particular that this precludes the utilization of other search strategies or optimization algorithms such as branch and bound, cutting planes, dynamic programming, or tree-based AI search algorithms.

As a consequence, there might be a loss in computational efficiency in the solution of certain instances of SAPs. For example, if we consider a simplified version of the down-selecting problem in which the value of elements is independent of elements already chosen in the subset, a very efficient dynamic programming formulation is well-known that solves the problem in polynomial time. The upside of this approach is a flexible framework that allows the system architect to focus on domain-specific knowledge, since a generic search algorithm and a library of heuristics for each class of SAP are provided.

2.3.2 Description of the generic search algorithm

The generic search algorithm is described in this subsection. The algorithm is population-based or multi-point, as opposed to point-to-point. Point-to-point algorithms start with a guess solution and then iteratively update the solution hopefully improving it. Multi-point or population-based algorithms follow the same procedure except the search is performed around multiple points simultaneously[4]. The choice of a population-based algorithm over a point-by-point or multi-point approach was driven by the following considerations: a) evolutionary algorithms (e.g. genetic algorithms, particle swarm optimization), which have been successfully applied to a variety of large-scale combinatorial optimization problems, are all population-based; b) it is very easy to apply parallel computing techniques on population-based algorithms in order to improve computational efficiency, since for example the evaluation of each individual in the population can be parallelized; c) dealing with a population of architectures is more "natural" than dealing with a single architecture since the output of the process is to be a set of preferred architectures as opposed to "the best" architecture.

The generic algorithm is shown in Figure 17. The steps of this generic algorithm are described below. In a way, these are just the meta-steps of any population-based algorithm. For a detailed discussion of a similar population-based optimization algorithm, see (Deb, 2004).

Step 1: Create an initial population. The first step of any population-based algorithms is to generate an initial population. There are three basic sources of architectures for an initial population, and in general any initial population will have architectures coming from these three sources: a) an initial population may be partially or fully supplied by the system architect, containing architectures of interest that the system architect would like to look around; b) it may also be partially or fully randomly generated using a random architecture enumerator; c) finally, results from previous simulations may also be used (this is sometimes called "warm starting").

[4] Here, the word simultaneously is used with some liberty as it does not necessarily require the use of parallel computing techniques.

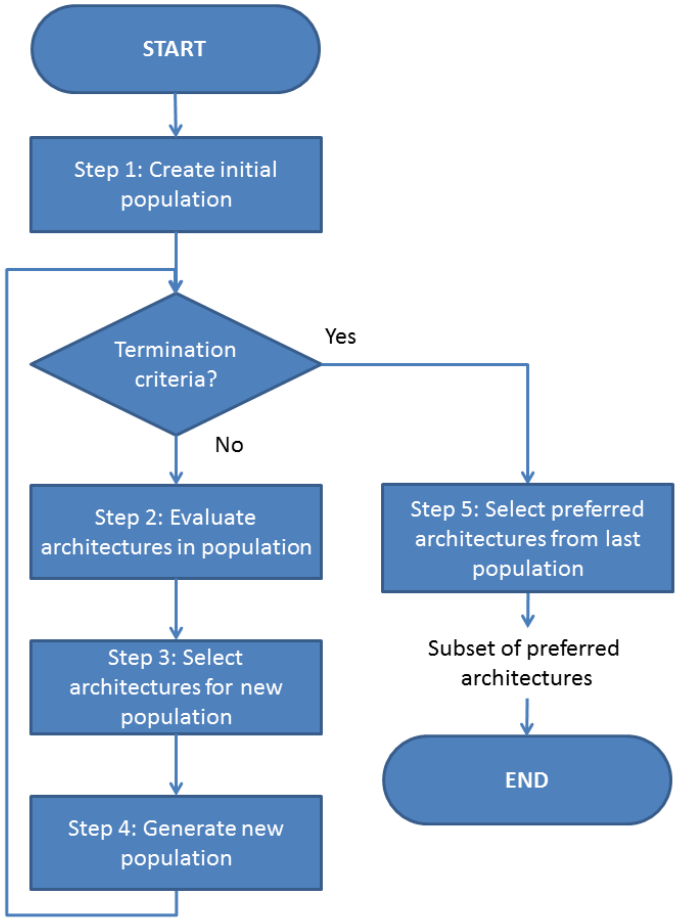


Figure 17: Generic population-based search algorithm developed for the framework

Step 2: Evaluate architectures in population. In this step the architectures in the population are evaluated using the metrics of choice. These metrics will be specific to the problem at hand, but will typically include some assessment of the benefit or performance of the system, and some assessment of its lifecycle cost.

Step 3: Select architectures for new population. Typically, in order to hold the size of the population constant, the next population will be created based on a subset of the population as opposed to the whole population, and therefore a selection step is required. This subset of the population is called the **selection set**. A simple approach would be to select the top X% of architectures in the case of a single metric, or the X-fuzzy Pareto front in the case of multiple metrics.

This kind of approach is called a **greedy approach** or a **truncation selection operator**. Most selection operators also include a few less fit or dominated individuals in the selection set, because it has been empirically proven that doing so leads to better performance (see for example (Miller & Goldberg, 1995)). This will be discussed in more detail in Section 2.3.5.

Step 4: Generate new population. The goal of this step is to generate new architectures from the architectures in the selection set. This step can be further divided into two substeps: a) generation of architectures based on this subset of architectures; b) elimination of infeasible architectures. Concerning the generation of new architectures from the selection set, we consider again several sources of new architectures: a) architectures randomly obtained from small variations around one selected architecture (e.g. mutation operators in genetic algorithms); b) architectures obtained from recombination of two or more architectures from the selection set (e.g. crossover operator in genetic algorithms); c) architectures directly taken from the selection set. The fraction of new architectures that come from each source can be adjusted, but it is desirable that the size of the population is maintained constant throughout the search process.

Termination criteria: The algorithm stops iterating whenever any of these conditions are met: a) a maximum number of iterations set by the user a priori is reached; b1) the best individual in the population has not changed in the last N iterations, N being a user-defined parameter (this applies to single objective problems only); b2) the value of the best architecture has not changed by more than a certain tolerance in the last N iterations (this also applies to single objective problems only); c) the average Pareto spread has not changed by more than a certain tolerance in the last N iterations (this also applies to multi-objective problems only).

Step 5: Down-select preferred architectures. Finally, once termination criteria are met, a subset of preferred architectures are selected for recommendation to the system architect. For single-objective SAPs, these can be done by selecting the top X% of the architectures. For multi-objective problems, it can be accomplished in two steps: first, dominated architectures, or more generally architectures with a Pareto ranking higher than a threshold are discarded; second, a few architectures are selected within the fuzzy Pareto frontier by adding several inequality and equality constraints concerning individual metrics (e.g., cost < \$100M) or combinations of metrics (e.g., utility/cost $\geq$ 0.5 utils/\$M).

Throughout these five steps, there are several places where the algorithm can be customized by incorporating both domain-independent and domain-specific knowledge in the form of rules. In the rest of this section 2.2, we focus on the description of the different classes of rules used in the generic algorithm, namely: **grammars**, **approximate evaluation rules**, **search heuristics**, and **selection rules**.

We called "**grammars**" the set rules that concern the encoding and enumerating processes, which are closely related. These rules are used in **Step 1** and **Step 4** in the algorithm. **Approximate evaluation rules** are "rules-of-thumb" that replace expensive computations in the evaluation step (**Step 2**). **Search heuristics** are rules that make the search process more efficient, including selection, mutation, and crossover operators amongst others. Finally, **down-selection rules** are the set of rules used to select a small subset of preferred architectures.

Note that only a general discussion about each type of rule is provided in this section, as the implementation details depend on the class of the SAP at hand. Heuristics for the different classes of SAPs are provided in Appendix 9.1.

2.3.3 Grammars: rules for synthesis of feasible architectures

The notion of grammars in computer science was first introduced to designate automatic transformations on strings. In this context, (language) grammars are defined as a set of production rules for successive synthesis of strings, typically starting from an initial string. In addition to language grammars, shape grammars also became very popular in the 70's (Stiny & Gips, 1972).

In this thesis we extend the concept of grammar to the system architecting discipline - with some liberty - to designate both the encoding scheme and the set of rules that can be used for automatic synthesis of architectures. A more formal definition follows.

Definition (encoding scheme E): An encoding scheme E is a mapping (i.e., a function) that assigns an enumerable representation model M to a given system architecture A:

$$E: A \rightarrow M$$

Definition (grammar G): A grammar G is defined as a tuple $\langle E, C, R \rangle$ where E is an encoding scheme; C is a set of feasibility constraints that an architecture must satisfy; and R is a set of rules used in combination with C to generate feasible architectures.

2.3.3.1 Encoding schemes

The first step to define a grammar is to select an **encoding scheme** E, i.e. a data structure in which synthesis rules can be expressed. The fields of this data structure must coincide with the architecturally distinguishing pieces of information that characterize a system, that we call architectural attributes or, following Simmon's nomenclature, decision variables (Simmons, 2008).

In a particular architectural trade study, a system architecture differs from the others in at least one architectural attribute or decision. For example, in the case of a launch vehicle, an architecture may be defined by five architectural attributes or decisions: a) number of stages; b) propellant type for each stage; c) length of each stage; d) diameter; e) number of boosters. For each of these attributes, there exists a set of allowed values (e.g. number of stages can go from 1 to 3, propellant type can be "LH2", "RP1", or "CH4"). Assigning one value to each of the decisions yields one architecture.

We defined an encoding scheme as a mapping between a set of possible architectures and a set of enumerable architecture representations or models. In other words, an encoding scheme is a procedure to represent a real system architecture as an instance of an enumerable data structure whose fields represent architectural attributes. In the example of the launch vehicle, one architecture can be represented by an array of variables $[d_1, d_2, d_3, d_4, d_5]$ where d_1 is an integer between 1 and 3 that represents the number of stages, d_2 is a $1xd_1$ array of integers representing the propellant type (e.g. 1 means LH2, 2 means RP1, 3 means CH4), and so forth. Note that in this definition, "enumerable" is in the sense of Koo's Algebra of Systems (B. H. Y. Koo et al., 2009), i.e. which allows the explicit listing of all possible combinations of values for the properties of the data structures.

We can define several properties of an encoding scheme: a) information density, i.e., how big is the data structure representing the architecture; b) type of variables (continuous, discrete, Boolean, mixed); c) bijectivity (i.e., is there a one-to-one correspondence between real architectures and models); d) non-degradedness, or the question of whether small changes in the model domain translate into small changes in the real architecture domain.

These properties affect the process of exploring the architectural tradespace in several ways. First, they have an impact on the inherent complexity of the corresponding optimization problem. Indeed, some optimization problems are much harder to solve than others. Generally speaking, most problems involving only continuous variables will be solved more efficiently than problems involving integer or Boolean variables, thanks for example to the use of gradient information.

Second, they impact the size of the architectural tradespace in the model domain. One could reason that the size of the architectural tradespace is given by the number of architectural decisions and their options being considered, and therefore should be independent of the encoding scheme. However, some constraints may be hard to express as synthesis rules in a particular model domain, and the system architect may decide to relax these constraints, or to express them as penalties in the objective functions instead of hard constraints affecting the enumeration process. By doing this, the system architect is trading efficiency in the search against size of the tradespace.

Bijectivity in an encoding scheme implies a perfect one-to-one correspondence between real architectures and their representations. Bijectivity of an encoding scheme is a desirable property because once the model has been applied in the algebra of choice, the selected architectures in the model domain need to be transformed back into the real domain. If several real architectures have the same representation in that algebra, the problem is ill-posed. Consider the generic example of assigning five functions [f_1, f_2, f_3, f_4, f_5] to a number of elements of form. A possible encoding scheme uses an array of five integers where the ith entry represents the subset to which element *i* is assigned. For example, [1,2,3,4,5] represents an architecture where the five functions are performed by different elements of form, while [1,1,1,1,1] represents an architecture where all the functions are performed by a single element of form. We note that this encoding scheme, without any further constraints, does not satisfy the bijectivity property. Indeed, the following two different architectural arrays: [1,1,1,2,2] and [2,2,2,1,1] represent the same real architecture{$F_A \leftarrow [f_1, f_2, f_3]$; $F_B \leftarrow [f_4, f_5]$}. One can change from one representation to the other by changing the names of the elements of form. As it was pointed out before, in this example the elements of form {F_J} do not have any kind of physical meaning, they are rather just subsets of functions, which explains why the two representations are equivalent.

Another desirable property for an encoding scheme is that similarity in the model domain under a certain definition of distance (e.g. Euclidean distance between two arrays) translates into similarity in the real domain. In other words, if the representations of two architectures are similar (e.g. [1,1,1,2,2] and [1,1,1,1, 2] only differ in one variable), then these two architectures are also similar ({$F_A \leftarrow [f_1, f_2, f_3]$; $F_B \leftarrow [f_4, f_5]$} vs {$F_A \leftarrow [f_1, f_2, f_3, f_4]$; $F_B \leftarrow [f_5]$} which only differ in one function), and viceversa. Note that this is a strong property that is not satisfied in many cases. For example, [1,3,2,3,2] and [1,2,3,2,4] are very different in the model domain but very similar in the real domain. Transfer of similarity between the model and real domains is desirable because many optimization and search techniques utilize an iterative approach where solutions are progressively refined through small changes in the model domain (e.g. hill climbing algorithms). If small changes in the model domain produce big changes in the real domain, the performance of such techniques is reduced.

Finally, it is also desirable that encoding schemes have relatively simple interpretations and meaningful graphical representations. For example, binary representations of architectures where integer options are codified using a certain number of bits are sometimes used for formulation using genetic algorithms. However, these formulations are intuitively very far from the architecture. Just by looking at any graphical representation of the architectural array [0 0 1 1 0 1 0 1 0 1 0 1 0 1], it is very hard to visualize the architecture it is representing.

Hence, one should take the aforementioned issues into account when selecting an encoding scheme, and be aware that their choice may influence not only the ability of the user to intuitively make the connection between the physical world and the modeled world, but also the performance of the optimization or search algorithms used to explore the architectural tradespace.

2.3.3.2 Enumeration rules

Enumeration rules are the procedures or algorithms utilized to enumerate all possible - or some - architecture representations using the selected encoding scheme. For instance, for an encoding scheme of the form $[d_1, d_2, d_3, d_4, d_5]$, where d_i can take values in a set of size n_i a possible enumeration rule would be the following:

```
(define-rule example-of-enumeration-rule
    "This rule recursively enumerates all valid architectures for an SAP with 5
decision variables and several possible values for each decision variable"

IF there is a fact ?arch <- (of type architecture (with assignments $?ass))
AND  (length$ of $?ass) < 5

    => (THEN)

RETRACT ?arch
COMPUTE
    (?d1-opt = (1 2 3))
    (?d2-opt = (yes no))
    (?d3-opt = (Moon Mars NEO L1))
    (?d4-opt = (none 1 2))
    (?d5-opt = (5 10 15))
    (?n = (length$ $?ass))
    (?list =  (str-cat "?d" (?n + 1) "-opt"))
    (for (?i = 1) (?i <= (length$ ?list)) (++ ?i)
        (?new-ass = (insert$ ?ass (+ ?n 1) (nth$ ?i ?list)))
ASSERT (new fact of type architecture (with assignments ?new-ass))))
```

Code 2: Example of enumeration rule for an SAP with 5 decision variables

This enumeration rule would effectively enumerate all $\prod_i n_i$ possible architectures (216 in this case).

2.3.3.3 Enumeration constraints

Enumeration constraints are logical constraints that determine an architecture's feasibility. Enumeration constraints are hard constraints, i.e., an architecture that does not meet all enumerations constraints is immediately discarded. This is in contrast to soft constraints, which are feasibility constraints that are included in the objective functions in the form of penalties. Soft constraints are used to improve the performance of the search process, and therefore they are discussed in the subsection concerning search heuristics.

Furthermore, not all hard constraints are enumeration constraints. There might also be search heuristics that take the form of hard constraints in order for example to reduce the size of the tradespace. While both types of constraints are hard constraints as opposed to soft constraints, enumeration hard constraints are very different in nature from search hard constraints. Enumeration constraints are intimately linked to the encoding scheme and the enumeration rules. Eliminating enumeration constraints would lead to the existence of mathematically non-sensical architectures, i.e. architectural models or representations without corresponding real architectures.

Let us consider an example in which architectures are partitions of a set of 4 elements {A, B, C, D} into 2 disjoint subsets. For instance, an architecture can be $A1 = \{\{A, B\}, \{C, D\}\}$, and a different architecture can be $A2 = \{\{A\}, \{B, C, D\}\}$. There are seven possible different architectures in this sapce[5]. In this example, the grammar of choice will have to ensure that: a) each element gets assigned to no less and no more than one subset; b) that the number of subsets is exactly equal to 2. These constraints can be implemented by choosing a smart encoding scheme and trivial sets of enumeration rules and constraints, or a smart set of enumeration rules and trivial encoding scheme and enumeration constraints, or by having a simple encoding scheme and set of enumeration rules, and then applying enumeration constraints a posteriori. Let us assume that our encoding scheme consists in a 4 x 2 matrix of Booleans, where entry (i, j) is equal to 1 if element *i* is assigned to subset *j*, and it is set to 0 otherwise. In this case, enumeration constraints need to be added to ensure a) that each element gets assigned to exactly one subset. If we do not add this enumeration constraint, non-sensical architectures such as $zeros(4,2)$ or $ones(4,2)$ could be enumerated. Conversely, we can consider an alternative grammar consisting in an array of 4 integers from the binary algebra {1, 2}, so that architecture A1's representation would be [1 1 2 2] and architecture A2's representation would be [1 2 2 2]. This second grammar clearly ensures all of the constraints without the need of explicit enumeration constraints.

2.3.4 Approximate evaluation rules

From all the tasks in the SAT, architecture evaluation is the one that depends the most on the problem at hand, and therefore the hardest to discuss in general terms. Most tools develop ad-hoc metrics for each specific SAP. Sometimes, dedicated simulation tools are used to obtain some of these metrics (e.g., AGI's Satellite Toolkit for coverage metrics in mission analysis). Although these metrics and the objective functions used to compute them can be very different in nature from one SAP to another, there are still certain similarities across all metrics utilized in systems architecting.

[5] This number can be computed using Stirling numbers of the second kind: S{4,2} = 7. See for example (Grimaldi, 2003) for a discussion on combinatorics in general and Stirling numbers in particular.

For example, all metrics are derived or should be derived from stakeholder needs, in a process of logical decomposition in the function domain, from stakeholder needs, to functional and performance requirements. Concerning objective functions, in the most general sense they all relate the architecture (decision variables) to these metrics. Another way of seeing this is that objective functions relate elements of form (e.g., instruments) to capabilities (e.g., measurements) and requirement satisfaction. Note that both capabilities and requirements are expressed in the form of functions + performance. An architecture scores high in a certain metric when capabilities meet or exceed requirements concerning this metric, i.e., when all required functions are executed with at least the required level of performance. Hence, in the most general case, objective functions essentially do four things: a) a logical decomposition of the architecture in the form domain; b) crossing the boundary between form and function; c) propagating performance attributes throughout the architecture hierarchy; d) comparing performance of capabilities to required performance.

These few observations motivated the development of a general framework for value assessment in system architecting based on: a) a systematic development of functional and performance requirements from stakeholder needs; b) a systematic computation of architectural capabilities (function and performance); c) a systematic comparison between requirements and capabilities. The whole framework uses an RBES to allow the system architect to focus on the domain knowledge and the problem-specific knowledge. The rest of the problem (property propagation in the hierarchy, decomposition and aggregation, pattern matching) is left to the inference engine inside the RBES.

This framework is called VASSAR (Value ASsessment for System Architecting using Rules). VASSAR is described in detail in the next chapter.

2.3.5 Search heuristics: rules to constrain and guide tradespace search

Most state-of-the-art algorithms for large multi-objective combinatorial optimization problems utilize some kind of heuristic to make the search process more efficient (Ehrgott & Gandibleux, 2000). Such search heuristics typically include at least some strategy for **local search**, and some strategy for avoiding getting stuck in a local optimum, i.e., a strategy for **global search**. Other possible heuristics are tradespace size reduction strategies, decomposition strategies, and repair operators. Each of these types of search heuristics is discussed below.

2.3.5.1 Tradespace size reduction heuristics

When SAPs are very large, it may become necessary to apply heuristics to reduce the size (i.e., number of architectures) and/or the dimensionality (dimension of the data structure representing the architecture) of the tradespace, i.e., reduce the number of solutions and/or the number of metrics.

In the continuous domain, tradespace size reduction strategies are typically based on a decomposition of the initial space in a set of uncorrelated orthogonal components, and subsequent truncation of the usually infinite series of terms into the terms accounting for most of the variance. This techniques are sometimes called model order reduction techniques, and they can be applied both to reduce the number of metrics and the number of design variables. However, they are mostly applied to continuous variables.

If model order reduction strategies are not applicable, or if they fail to produce a problem of tractable size, it is common to decompose the problem into smaller tractable sub-problems that contain only a subset of the architectural variables and can be more efficiently solved. Formal decomposition strategies are based on clustering algorithms, ranging from the simple k-means algorithm, to more involved statistical methods such as neural networks. A survey of clustering algorithms can be found in (Xu & Wunsch II, 2005).

Finally, as noted earlier in Section 2.3.3, hard constraints can also be used as a means to reduce the size of the tradespace. However, in this case they are different from enumeration hard constraints because eliminating them would still lead to architectures that "make sense", or more formally, to architecture representations that have an image in the real architecture domain once the encoding scheme is applied.

2.3.5.2 Local search heuristics

Local search heuristics perform small variations around a given solution with the goal of finding similar feasible and hopefully better solutions. We distinguish between two types of local search rules: **non-informed** and **informed**. Non-informed heuristics apply essentially random variations. This is the case for instance of **mutation operators** in genetic algorithms. Informed heuristics typically compute a direction in the tradespace where "better" solutions are likely to be. In continuous domains, one such direction is the gradient. In combinatorial spaces, since gradients cannot be readily defined, one relies on heuristics to compute these directions. An example of informed heuristics is **crossover operators** in genetic algorithms. Crossover operators combine characteristics from two "good" solutions called the parents to create a hybrid of the two called the child, which is expected to be "better" than both parents. Note that there is no guarantee that this will be so, since this is only a heuristic.

Other informed heuristic rules include for example the use of **tabu lists** (Glover, 1990a, 1990b). In tabu lists, a list of recently visited poor solutions is maintained with the goal of not revisiting them again in the future. This avoids losing time in regions of the tradespace that have proven not to be very promising.

2.3.5.3 Global search heuristics

In non-convex optimization, i.e., in problems in which there are several local optima, hill-climbing strategies are not enough to guarantee convergence to the global optima, and it becomes necessary to implement a strategy to avoid getting stuck in local optima. This is accomplished in part by mutation operators in genetic algorithms, although mutations tend to be small variations and therefore may not be as effective as other methods in avoiding local optima.

Other algorithms incorporate a random component in the decision of "where to go next". In these strategies there is a non-zero probability of following a suboptimal path during the search. Another strategy consists in the parallelization of the search process, so that multiple local searches are conducted simultaneously[6]. An example of this in population-based algorithms is the division of the population in sub-populations, with the possibility of individuals migrating from one population to another. This is done for example in Deb's non-dominated sorting genetic algorithm, NSGA-II (Deb, Pratap, Agarwal, & Meyarivan, 2002), which is implemented in Matlab's optimization toolbox.

2.3.5.4 Repair search heuristics

It is well known in meta-heuristic optimization that keeping infeasible solutions in the population during the search process can lead to faster convergence, because the best solutions may be obtained by searching around an infeasible solution (Fisher, 2004), (Chu & Beasley, 1998). Therefore, some search algorithms also include a strategy to project an infeasible solution onto the space of feasible solutions, i.e., finding the feasible solution that is "closest" to the original solution given some distance metric (e.g. Euclidean distance for real vectors, or Hamming distance for binary vectors). We call these rules **repair search heuristics**.

2.3.6 Selection rules: rules for architecture down-selection

The down-selection task consists in the selection of a small subset of preferred architectures from a larger set of architectures, based on several criteria. We distinguish between two types of criteria: objective criteria, whose goal is to find the set of non-dominated architectures or Pareto frontier; and subjective criteria, whose goal is to select a few architectures within the Pareto frontier.

[6] Note that in this context, the terms "simultaneous" and "parallel" are taken with some liberty, and in particular they do not have the same meaning as in the field of parallel algorithms in computer science. Hence, these searches need not occur truly simultaneously in different processors.

2.3.6.1 Objective down-selection rules

In single-objective optimization problems, the goal is to find the solution that maximizes (or minimizes) a given metric such as performance or cost, and this solution is generally unique under certain conditions, namely convexity of the objective function. Conversely, in multi-objective optimization, because of different objectives are conflicting, it is impossible to define a single optimal solution. Instead, the concepts of dominance and Pareto optimality are used. A feasible solution is called efficient or non-dominated if and only if there exists no other feasible solution that: 1) is better than or equal to the former in all metrics simultaneously; 2) is strictly better than the former in at least one metric. Strong dominance requires strict inequalities in all metrics. The set of all non-dominated solutions is called the **Pareto frontier**, or trade-off curve. The **utopia point**, or shadow minimum, is a fictitious point that simultaneously optimizes all metrics. Assuming that an optimal solution exists for each metric when considered in isolation, the utopia point would only exist in the rare case where a single solution minimizes all metrics. These individual minima are often called anchor points. In general, we can define a convex hull of individual minima as the set of all convex combinations of anchor points. In 2D, this special line is called the **utopia line**.

Methods that do not combine the multiple objectives in a single objective try instead to find the set of non-dominated points. Several algorithms can be applied for this purpose. The most elementary algorithm arguably is the **weighted sum approach** by Zadeh (L. Zadeh, 1963). The weighted sum approach consists of the following steps: 1) normalize all metrics between 0-1 and transform them into Larger-Is-Better (LIB, or Smaller-Is-Better, SIB) metrics[7]; 2) create a weighted average of the normalized LIB/SIB metrics; 3) solve the single-objective optimization problem to find one point on the Pareto frontier; 4) change the weights, and iterate to find more points on the Pareto front. The limitations of this approach are: 1) it only finds the convex portion of the Pareto front, i.e., in general it does not find all the efficient solutions; 2) the points that it finds are not necessarily well spaced on the Pareto front, and therefore interesting parts of the tradespace may be poorly explored.

These two problems are partially overcome by De Weck's **Adaptive Weighted Sum** (AWS) algorithm for biobjective problems (O. D. Weck, 2004), in which the weights are intelligently adapted in real time from iteration to iteration, as opposed to a priori, in order to focus on non-explored and non-convex regions.

[7] Larger-Is-Better and Smaller-Is-Better are terms coined by Messac in (A. Messac & Ismail-Yahaya, 2002) to increasing or respectively, decreasing, utility functions with respect to a certain metric. For example, cost is clearly an SIB metric, while performance is an LIB metric. Messac also defines Center-Is-Better (CIB) metrics in his paper, in an effort to characterize the uncertainty that exists in decision maker preferences.

Several methods also exist that are based on lines perpendicular to the utopia line. Das and Dennis proposed the **Normal Boundary Intersection** method (I. Das & Dennis, 1998). In NBI, the Pareto frontier is found by solving a series of single-objective optimization problems in directions that are normal to the convex hull of individual minima. It is usually cited as a disadvantage of NBI that it may find non-efficient solutions in non-convex regions (this is not the case for WS or AWS). NBI's main advantage is that it produces evenly distributed Pareto frontiers, like AWS.

Messac's normalized **Normal Constraint (NC) method** (a. Messac, Ismail-Yahaya, & Mattson, 2003) is similar to NBI in both strategy and limitations. NC finds an evenly distributed set of points on the utopia line. Given a point on the utopia line P0, a point on the Pareto frontier P is obtained using the following procedure: 1) define the line that is perpendicular to the utopia line and passes through P0; 2) move on that line, from P0 towards the infeasible region, and find the furthest point on the line that is feasible. The NC method may also find non-Pareto efficient points, for the same reasons as NBI. Therefore, NC features a Pareto filter that eliminates non Pareto-optimal points at the end. Also similar to the NBI and NC approaches, is the **recursive knee algorithm** (Lin, 2010), a specialization of the more general sandwich method to approximate convex functions (Rote, 1992). See (Ehrgott & Gandibleux, 2000) for a comprehensive review of multi-objective constrained optimization problems, and algorithms to solve them. **Any of these algorithms can be used as objective selection rule**, as shown in Code 3.

```
(define-rule DOWN-SELECTION::delete-archs-not-enough-pareto-ranking
    "Delete all archs with a pareto ranking that does not meet min pareto ranking
requirements"
IF there is a fact ?arch <- (of type ARCHITECTURE (with pareto-ranking ?p))
AND there is a fact (of type DOWN-SELECTION::MIN-PARETO-RANK (with min-pareto-rank
?min-pareto-rank&:(< ?min-pareto-rank ?p)))
    => (THEN)
    (RETRACT ?arch)
    )
```

Code 3: A down-selection rule for deleting fuzzy-dominated architectures

We close this subsection by defining the concept of **fuzzy Pareto frontier**, introduced by Smaling and De Weck [6]. In a fuzzy Pareto frontier, designs that are "close" to the real Pareto frontier are also retained even though they are dominated. This is justified for at least two reasons: 1) there is modeling uncertainty that makes it difficult to discriminate similar designs; 2) there is uncertainty in all the other processes of the system development process, and designs on the real Pareto frontier are usually the least robust to these uncertainties [6].

Several approaches exist to compute a fuzzy Pareto frontier. One approach is based on a search algorithm that tries to find architectures in an inward direction perpendicular from the Pareto frontier, at a given maximum distance (Lin, 2010). Another approach, that we use in this thesis, is to recursively apply a simple Pareto filter to a set of architectures obtained by removing from the previous set the non-dominated architectures.

Whatever method is used, the objective selection rule can always take the form of a minimum required Pareto ranking, so that architectures that are not in the first few Pareto frontiers are automatically eliminated.

Note that by considering fuzzy Pareto frontiers instead of standard Pareto frontiers, dominated architectures are kept in the analysis, and therefore one could argue that this rule actually belongs to the category of *subjective* down-selection criteria. We classified this rule as an *objective* down-selection rule because its goal is to find the (fuzzy) Pareto frontier, as opposed to choosing architectures within the Pareto frontier.

2.3.6.2 Subjective down-selection criteria

Computation of the Pareto frontier or fuzzy Pareto frontier is typically not enough to reduce the number of architectures to a manageable number. Therefore, additional rules need to be taken into consideration to select a few architectures within the Pareto frontier.

One possibility to further reduce the number of architectures is to add hard inequality or equality constraints for one or more metrics or combinations of metrics that take the form of hyperplanes and half-spaces in the property variable space. Trivial examples are **maximum cost** rules, **minimum performance** rules, or **maximum risk** rules. More involved hyperplanes can use **weighted sums** of different metrics: for instance, one could define an "overall system engineering performance" as a weighted sum of cost, risk, and performance, and then add a constraint on minimum overall systems engineering performance. De Weck's isoperformance set (O. L. de Weck & Jones, 2006) would fall under this category of rules.

Note that adding these constraints is equivalent to intersecting the Pareto frontier with one or more half-spaces defined by the constraint hyperplanes, as illustrated in Figure 18. Blue crosses in Figure 18 represent feasible architectures in the performance-cost space. Red circles indicate architectures that are fuzzy non-dominated (in this example, the minimum Pareto ranking was set to three).

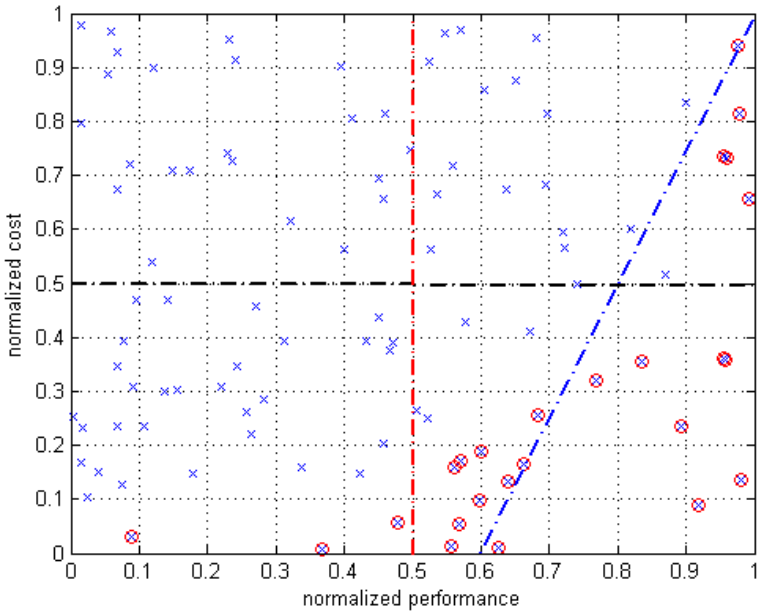


Figure 18: Effect of different types of down-selection rules on Pareto front

Thus, all architectures that are not circled will be discarded. The red dashed line marks the boundary of a minimum performance hyperplane constraint. All the architectures to the left of this line will be discarded. The black dashed line marks the boundary of a maximum cost hyperplane constraint. All the architectures above this line will be discarded. Finally, the blue dashed line marks the boundary of a minimum utility hyperplane constraint. All the architectures to the left or above this line will also be discarded. Note that this set of down-selection rules yields seven architectures. The general code for such rules is provided in Code 4.

```
(define-rule DOWN-SELECTION::delete-archs-outside-of-half-space
    "Delete all architectures that do not satisfy half-space constraint"

IF there is a fact ?arch <- (of type ARCHITECTURE (with metric1 ?m))
AND there is a fact (of type DOWN-SELECTION::HYPERPLANE-CONSTRAINT (with metric1
?min-m1&:(> ?min-m1 ?m)))

    => (THEN)

    (RETRACT ?arch)
    )
```

Code 4: A down-selection rule in the form of a hyperplane constraint

Another typical way of choosing within architectures of a Pareto front is to define a single objective problem in the space of remaining architectures by combining the different metrics in one of two ways:

- using a weighted average
- using a 1-norm, 2-norm (Euclidean), or infinity-norm (i.e., the maximum) distance to, either the utopia point, or the **target point** (i.e., the desired level of performance for each objective). A popular example of this is **goal programming** (Steuer, 1986), which minimizes the square sum of the differences between the metrics and the targets. The **compromise programming** approach proposed by Chen in (Chen, Wiecek, & Zhang, 1999) is a generalization of goal programming, in that it suggests the use of any norm, as opposed to the Euclidean distance.

Using one of these definitions, a possible subjective down-selection rule consists in retaining only the architectures that are top X% under the combined criteria.

```
(define-rule DOWN-SELECTION::delete-archs-too-little-utility
    "Delete all archs with a utility that does not meet min utility requirements"
IF there is a fact ?arch <- (of type ARCHITECTURE (with utility ?u))
AND there is a fact (of type DOWN-SELECTION::MIN-UTILITY (with min-utility ?min-
utility&:(> ?min-utility ?u)))
    => (THEN)
    (RETRACT ?arch)
    )
```

Code 5: A down-selection rule that keeps only architectures with utility higher than a certain threshold

Finally, another possibility for selection rules is to incorporate the notion of robustness into the selection. In this case, a Monte Carlo analysis is run where the architectures are evaluated under a broad variety of scenarios that take into account different sources of uncertainty such as technical uncertainty (e.g., uncertainty in performance parameters of the system components) or market uncertainty (e.g. uncertainty on demand for the system). **Robustness selection rules** retain only the architectures that meet any of the above criteria (e.g., min performance, max cost, min utility) under all or most of the plausible scenarios. For example, instead of enforcing a deterministic rule that the lifecycle cost of the system needs to be below a certain threshold, a robust implementation of the same rule would enforce that the architecture stays below that threshold for 80% of the scenarios considered.

2.3.7 Summary and discussion

In this section, we have described the generic search algorithm that can be used to solve a KI SAP from any of the classes of SAP introduced earlier. This generic search algorithm is a population-based optimization algorithm. We justified this choice over point-to-point algorithms using both empirical evidence that these algorithms have been successful in the past in tackling large-scale combinatorial optimization problems, and a strategic choice to trade exactness and computational efficiency against expressive power.

In other words, we prioritized being able to find a few good solutions to a broad variety of SAPs, than being able to find the exact best solution to a very restricted set of SAPs, or to rough versions of the original problems where important modeling detail has been abstracted out in order to allow for the application of known efficient techniques.

Furthermore, we introduced the four types of heuristic rules that are used to tailor the algorithm to the particular SAP and domain of application: grammars, search heuristics, evaluation rules, and selection rules. This algorithm is intended to be flexible in the sense that it can easily accommodate a large body of class-specific and domain-specific knowledge in order to improve efficiency in the problem-solving process.

A discussion point arose in the differentiation between hard constraints in enumeration and search. We acknowledge that the difference between the actual implementation of these rules may be very small or even non-existent in some cases. In practice, the classification of a certain rule as one or the other will depend mostly on the intent of applying that rule: the intent of enumeration constraints is to permantently eliminate non-sensical architectures from the tradespace, whereas the intent of introducing hard constraints in tradespace size reduction rules is to guide the search towards more promising regions of the tradespace by temporarily eliminating a family of architectures from the tradespace.

While grammars, search heuristics, evaluation rules, and selection rules have been described in this section, their function and implementation will become clearer when we provide specific examples of these rules for each class of SAP.

2.4 A library with heuristics for the different classes of SAPs

Different classes of SAP were introduced in Section 2.2. After that, a generic search algorithm was presented that can be used to solve any class of SAPs. This algorithm was described as knowledge-intensive, because it is designed so that incorporating SAP-specific and domain-specific knowledge into the algorithm can be done easily. This expert knowledge is incorporated in the form of four types of rules: grammars, search heuristics, approximate evaluation rules, and down-selection criteria. In this section, we provide efficient grammars and search heuristics for these classes of SAPs. Note that approximate evaluation rules are discussed in Chapter 3, and down-selection criteria are essentially independent of the class of SAP, and therefore they are out of the scope of this section.

These heuristics are only provided for four out of five classes of SAPs. Suggesting efficient heuristics for a class of SAP requires substantial experience solving that class of SAPs.

Thus, we restricted this section to the four classes of SAPs with which we have had substantial experience in the last four years: assigning problems, selecting problems, partitioning and covering problems, and permuting problems. The class of connecting problems is left for future work.

2.4.1 Assigning Problems

2.4.1.1 Grammars

Encoding scheme: Let us consider a generic AsP in system architecture, with a fix set of *n* decisions $D = \{d_1, d_2, \ldots, d_n\}$, where each decision d_i is associated with a discrete set of options $O_i = \{d_{ij}\}$. One possible encoding scheme for this example is to map each assignment to an array of symbols (e.g. strings, integers, reals) containing the option selected for each decision:

$$E_1: A_k \rightarrow [s_1, s_2, \ldots, s_n]$$

where $s_i \in O_i$. Another possibility is to map the assignment to an array of integers, where each integer uniquely represent a valid option for a certain decision. More precisely:

$$E_2: A_k \rightarrow [z_1, z_2, \ldots, z_n]$$

where $z_i \in [1, |O_i|]$.

A function f that returns the position of each option in the list of options $f(d_{ij}) = j$, is needed to transform back and forth between E_1 and E_2. While E_2 is better suited for integration in optimization packages, E_1 is much simpler to interpret. These two encoding schemes are combined to create a template in CLIPS, as shown in Code 34 in the Appendix.

Enumeration rules: The next step is to create a rule that will enumerate all possible architectures once the template has been created. A simple rule that accomplishes that is provided in Code 35 in the Appendix.

Since the size of the tradespace is exponential with the number of decisions, an alternative partial enumeration rule is required that only enumerates a fraction of the tradespace. Such enumeration rule is provided in Code 36 in the Appendix. This rule utilizes a parameter to tune the fraction of architectures in the tradespace that need to be enumerated. When this parameter is set to 1, this rule will enumerate all architectures, as the previous rule did.

Enumeration constraints: Different types of constraints can be defined in AsPs. Particularly, there may be non-sensical or prohibited combinations of decisions and options. For example, in our Apollo architecture example, there was one decision for the Earth Orbit RDV (EOR yes or no) and one decision for the Earth Launch Type (EL direct or orbit). Clearly, it doesn't make sense to have EOR=no and EL=orbit because it only makes sense to launch to Earth orbit in the EOR mission mode. An example of rule encoding a logical constraint of this type is provided in Code 6.

```
(define-rule HARD-CONSTRAINTS::consistency-mission-mode-and-launch-type-EOR
    "This rule eliminates all architectures in which the mission mode
    is EOR and the type of Earth launch is inconsistent"

IF there is an (APOLLO-ARCHITECTURE (with EOR yes) (and EL direct))

    => (THEN)

    (RETRACT the architecture)
)
```

Code 6: Example of enumeration constraint in the AsP

2.4.1.2 Search heuristics

Tradespace size reduction rules: As mentioned earlier, logical hard constraints can also be used to reduce the size of the tradespace. For instance, one can decide to look around the region of the tradespace that has a particular assignment for a decision. The second strategy to reduce the size of the tradespace is to decompose it or divide it in regions. In the case of assignment problems, clustering needs to take into account the connectivity between decisions.

In other words, decisions that are coupled need to be kept in the same cluster, while decisions that are decoupled can be studied independently. Simmons proposed a way of measuring the degree to which decisions are coupled that he called the *property sensitivity metric* (Simmons, 2008). However, this method requires the computation of the average of the metrics of interest in several regions of the tradespace, and thus it can only be used *a posteriori*. A clustering algorithm can only rely on information that is available before the clustering occurs, such as the equations and algorithms inside the objective functions. Hence, a clustering algorithm for the AsP could scrutinize the different value functions for the use of decision variables, and construct a dependence matrix that contains the decisions that go into the calculation of each metric. Then, an adjacency matrix can be constructed where element (i, j) is set to 1 if there exists a metric that utilizes both decision variables i and j, and it is set to 0 otherwise.

Local search: In terms of non-informed local search rules, a mutation operator that changes the value of a random decision to a new random value is provided in Code 37 in the Appendix.

A single point crossover operator is also included that combines decisions from the "father" with decisions from the "mother" in order to produce new and hopefully better "children" architectures. The code of such crossover operator is given in Code 38 in the Appendix.

Repair operators: Since the search rules proposed do not check for feasibility of the new architectures generated, it is appropriate to define a repair operator. A repair operator can be defined for the AsP that will project an infeasible architecture into the set of feasible architectures by changing the value of a random attribute in an infeasible combination of decisions into a random feasible combination. A trivial implementation of these repair operators is to have as many rules as logical constraints, and have each rule be of the following type:

```
(define-rule HARD-CONSTRAINTS::repair-consistency-mission-mode-and-launch-type-EOR
    "This rule modifies all infeasile architectures in which the mission mode
    is EOR and the type of Earth launch is direct by modifying either one of these
two attributes randomly"

IF there is an (APOLLO-ARCHITECTURE (EOR yes) (EL direct))

    => (THEN)

IF (random-number > 0.5) then (modify architecture with(EL orbit))
ELSE (modify architecture with (EOR no)))
    )
```

Code 7: Example of a repair rule for the AsP

An alternative efficient is to combine several cases in the same rule by using variables for the values of the decisions and then checking several logical constraints on the variables. However, this alternative approach may be computationally less efficient depending on the number of architectures, as computational complexity is proportional to RF^P where R is the number of rules, F is the number of facts, and P is the number of patterns in the fact.

2.4.2 Partitioning and Covering Problems

2.4.2.1 Grammars

Encoding scheme: Let us consider a generic set partitioning problem in system architecture, in which a fix set $U = \{e_1, e_2, \ldots, e_m\}$ of m elements needs to be partitioned into subsets. One possible encoding scheme for this example is an array of integers:

$$E_1: P_i \rightarrow [d_1, d_2, d_3, \ldots, d_m]$$

$$1 \leq d_i \leq 1 + \max_{j<i} d_j$$

where d_i is the number of the subset that contains e_i, for example $d_1=2$ means that e_1 is assigned to subset S_2. Note that the constraint added ensures that no subsets are left empty.

Particularly, let m = 5. One architecture (i.e., one partition P_1) may consist of a unique subset containing the five elements, i.e. $P_1 = \{S_1 = \{e_1, e_2, e_3, e_4, e_5\}\}$. Another architecture or partition P_2 may distribute the five elements in two subsets: $P_2 = \{S_1 = \{e_1, e_2, e_3\}, S_2 = \{e_4, e_5\}\}$.

We propose an alternative scheme that consists in a binary array where each variable represents a possible subset, from a library of *NS* precomputed subsets.

$$E_2: P_i \rightarrow [b_1 \quad b_2 \quad \dots \quad b_{NS}]$$

where b_j is the Boolean architectural variable that contains the information about whether subset j from the subset database is selected ($b_j = 1$) or not ($b_j = 0$). In a valid architecture, $b_j \in \{0,1\} \forall j = 1 \dots NS$. Ideally, NS would be equal to all possible 2^N subsets. However, if a maximum number of elements per subset N_{MAX} is imposed as a heuristic, NS will only grow as the sum of all possible combinations up to N_{MAX}:

$$NS = \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{N_{MAX}}$$

The two architectures from the example encoded using the alternative scheme look as follows:

$$A_1 = [0\ 0\ 0\ 0 \dots . 0\ 1]$$

$$A_2 = [0\ 0 \dots 0\ 1\ 0 \dots 0\ 1\ 0 \dots 0\ 1\ 0 \dots 0]$$

A_1 contains exactly one ‘1’ and 31 ‘0’, the ‘1’ corresponding to the subset containing all the elements. A_2 contains exactly 3 ‘1’ and 29 ‘0’, the 3 ‘1’s corresponding to the three subsets.

The major trade-off between these two representations is whether we want to enforce the set partitioning constraint during the process of generating a valid architecture, or rather in a posterior search for feasible architectures. Classical set partitioning formulations typically use the latter: they precompute all possible subsets of elements (i.e. all possible satellites) and their metrics, and then represent a set partition (an architecture) as a combination of these subsets, (Chu & Beasley, 1998; Garfinkel & Nemhauser, 1969; Levine, 1994; R. Marsten, 1981; R. E. Marsten, 1974). Thus, the instrument packaging problem is effectively transformed into a subset selection problem from a library of pre-evaluated satellites, with an additional equality constraint (the set partitioning constraint).

However, some optimization algorithms (e.g. genetic algorithms) have difficulty handling the set partitioning constraint under this form, as well as other non-linear equality constraints, in the case of large tradespaces. Although some relaxation or penalty schemes exist that might help overcome this problem (Chu & Beasley, 1998), the grammar that we selected is capable of completely bypassing the problem by construction. Indeed, the set partitioning constraint is implicitly enforced at the time of architecture enumeration as opposed to being an ulterior search for feasible architectures, which is a desirable property.

On the other hand, when applicable (i.e. relatively small trasdespaces), the second formulation is much more computationally efficient in the evaluation phase, since most of the computation is done off-line in a pre-processing step.

In addition, the previous example illustrated how in the worst case, E_2 contains exponentially more architectural variables than E_1 , since $NS = \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{N_{MAX}} \approx 2^N$ for $N_{MAX} \to N$. Finally, assuming a certain number of elements per subset on average (e.g., 3), N/3 subsets would be required for a feasible set partitioning. This means that E_2 will be very sparse, since only N/3 out of NS variables will be non-zero, which may lead to computational inefficiencies in some cases.

Enumeration rules: Under the E_1 encoding scheme, an efficient full factorial enumeration rule that leverages from the recursivity of the CLIPS language is provided in Code 39 in the Appendix. Let R_1 be this enumeration rule defined, and let unconstrained grammar $A = \langle E, \phi, R_1 \rangle$. While this rule is very efficient in enumerating all possible set partitions, full enumeration becomes impossible for numbers of elements between 10 and 13 for current computing technology, because Bell numbers grow faster than exponentially. Thus a different grammar is required that allows for non-exhaustive exploration.

Such rule is provided in Code 39 in the Appendix. In these cases, the number of architectures generated is exponential with the number of elements, as opposed to faster than exponential in the original case. The base of this exponential will be equal to the average number of architectures generated by iteration, which for a uniform random function will be close to $0.5\,(N+1)$, N being the number of architectures tentatively asserted by iteration of the algorithm (See Code 40 in the Appendix for details of implementation).

Let R_2 be this enumeration rule, and let unconstrained grammar $B = \langle E, \phi, R_2 \rangle$. For small number of elements, both grammar A and Grammar B will produce the same number of architectures if N is allowed to increase sufficiently. The number of architectures is given by Bell numbers (e.g. $Bell(5) = 52$ in the example with five elements). Grammar B is much less efficient for small number of elements due to the random behavior, but since it is not exhaustive, it can explore much larger number of elements.

For reference, Table 3 shows the difference in computational complexity between the two grammars for numbers of elements ranging from 5 to 13. All values for Grammar B are averages and standard deviations over 20 trials with $N = 4$. Simulations were run and on an Intel Core 2 Duo CPU P9400 @2.4GHz with 4GB of RAM running Matlab 7.13.0.564 (R2011b) 32 bits on Windows 7 32-bits. More trials did not change the means nor reduce variance further. Asterisks indicate trials where algorithm did not terminate after the maximum allotted simulation time, fixed to 10 minutes.

Table 3: Comparison of computational efficiency of Grammars A and B.

# elements (m)	$Bell(m)$	Run time for Grammar A (s)	Run time for Grammar B (s)	% of architectures found by Grammar B
5	52	0.009 ± 0.002	0.009 ± 0.002	45.2% ± 11.3%
6	203	0.024 ± 0.007	0.019 ± 0.005	28.7% ± 7.7%
7	877	0.065 ± 0.017	0.042 ± 0.008	20.4% ± 5.2%
8	4,140	0.198 ± 0.012	0.074 ± 0.021	10.8% ± 3.8%
9	21,147	0.988 ± 0.033	0.171 ± 0.036	6.7%± 1.7%
10	115,975	5.200 ± 0.060	0.427 ± 0.133	3.2% ± 1.1%
11	678,570	32.168 ± 1.740	1.394 ± 0.352	1.9% ± 0.5%
12	4,213,597	> 600.000 (*)	3.602 ± 1.263	0.8% ± 0.3%
13	27,644,437	> 600.000 (*)	11.095 ± 3.387	0.4% ± 0.1%

We note that for $m = 12$ and beyond, the exhaustive enumeration algorithm fails to terminate in less than 600s. Moreover, we note an expected correlation between run time and %architectures found in Grammar B. Both run time and %architectures found in Grammar B are variable, but their variances are almost perfectly correlated.

Note that % of architectures found by Grammar B could have been improved at the price of increased running time by increasing N, i.e., the number of architectures tentatively asserted by iteration of the algorithm.

Grammar A and Grammar B are only applicable to PaPs and not CvPs, as Constraint PaP-2 is implicitly included in the encoding scheme (each element is only assigned to one subset by construction). A third grammar, Grammar C, can be constructed that can be used to solve both PaPs and CvPs. Grammar C assumes the existence *a priori* of a finite list of subsets. For example, in the previous example with $m = 5$, a list would be available containing the $2^5 - 1 = 31$ non empty subsets of 1 to 5 elements and their corresponding numerical benefits as values.

With this hashmap available, the PaP can be transformed into a 0-1 optimization problem where Boolean variable *i* is set to 1 if S_i is chosen for the partition, and it is set to 0 otherwise. Constraints PaP-1 and PaP-2 are then enforced *a posteriori* if required. Note that in this case, the PaP becomes an instance of the 0-1 knapsack problem where the cost function to minimize, or one of the cost functions to minimize, is simply the number of subsets. Grammars for knapsack problems are discussed in section 9.1.3.

Grammar C has always been the preferred grammar for PaPs and CvPs by the operations research community (Garfinkel & Nemhauser, 1969) because efficient methods have been developed to solve 0-1 integer optimization problems. However, in order to apply it to PaPs it requires the utilization of equality constraints to ensure that each element gets assigned to exactly one subset. This kind of equality constraints is hard to handle in large-scale problems for some optimization algorithms, including very popular ones such as genetic algorithms (Chu & Beasley, 1998). Furthermore, Grammar C results in a tradespace of much larger size than Grammars A and B for PaPs, since there are 2^{2^m-1} possible architectures under this grammar, whereas there are only Bell(m) feasible architectures. In order to partially alleviate this problem, the list of subsets available a priori is typically not exhaustive, and approximation algorithms are used to solve the 0-1 integer optimization problem.

Grammars A, B, and C can all be utilized to solve PaPs. However, in instances of the PaP where it is very unfavorable to break Constraint PaP-2, I have found Grammars A and B to be more efficient than Grammar C, which spends most of the computational time trying to reduce redundancy in the partitions.

Enumeration constraints: None of the grammars proposed for PaP require the use of any enumeration constraints.

2.4.2.2 Search heuristics

As explained in the previous section, our generic search algorithm utilizes four main types of operators: decomposition heuristics, tradespace size reduction rules, local search rules (informed and non-informed), global convergence rules, and repair rules. We discuss each of these operators individually in the next paragraphs.

Tradespace size reduction rules: Since large PaPs and CvPs are usually extremely hard problems to solve, it is necessary to keep the size of the tradespace as small as possible. With this goal in mind, we define several search heuristic rules for PaPs that are intended to apply as hard-constraints, i.e., any architecture not meeting these rules is immediately eliminated.

- A rule on the **maximum number of subsets** that a partition can have (Code 41 in the Appendix).
- A rule on the **maximum number of elements** that any subset in a partition can have. This avoids the exploration of architectures that would include subsets that are non-sensical for being too large. For example, if the elements are instruments and the subsets are satellites, this avoids looking at architectures that have satellites that cannot be launched by any current launch vehicle (Code 42 in the Appendix).
- One or more rules to force that **an element is assigned to a subset of size k** (typically, k = 1). This dramatically reduces the size of the tradespace. For k = 1 for example, this is equivalent to considering a PaP with m-1 elements (Code 43 in the Appendix)
- One or more rules **to** enforce **that two or more elements are grouped in the same subset** (Code 44 in the Appendix).
- One or more rules to enforce that **two or more elements are assigned to different subsets** (Code 45 in the Appendix).

In addition to these rules, decomposition of PaPs can be accomplished using any clustering algorithm. In order to use a clustering algorithm it is necessary to provide a metric for the distance between two elements: two elements that are "close" together using this distance tend to be assigned to the same cluster, and they tend to be assigned to different clusters if this distance is "large". Typically this distance is the Euclidean distance between the vectors representing the element on a certain space. Such space could be forever a space of architectural attributes of the elements. However, in some PaPs, this may result in poor heuristics, since there is little correlation in general between similarity in the space of architectural attributes and synergies between elements.

We propose another definition of distance between two elements in PaPs, based on the difference between the positive and negative interactions amongst them. For example, the sign and strength of bilateral interactions can be directly assessed by experts in a semi-quantitative form, following a semi-quantiative scheme such as the one laid out in Saaty's Analytic Hierarchy Process (Saaty, 2008). Alternatively, interactions can be computed as a difference in expected architectural value. This leads to the definition of a new matrix called V-DSM or value design structure matrix. The V-DSM is formally defined below.

Definition (V-DSM): The value design structure matrix or V-DSM is an *m x m* symmetrical matrix where element *(i,j)* of the matrix contains a numerical value representing the strength of the positive or negative interactions between elements *i* and *j*. *V-DSM(i,j)* is computed as follows:

$$VDSM(i,j) = VDSM(j,i) = V(\{e_i, e_j\}) - V\left(\{\{e_i\},\{e_j\}\}\right) = V(\{e_i, e_j\}) - V(\{e_i\}) - V(\{e_j\})$$

where $V(P_i)$ is the value of an architecture represented by partition P_i. $V(P_i)$ can be computed using any evaluation rules, and in particular for example using the VASSAR framework described in the previous section. Note that $VDSM(i,j) > 0$ indicates that elements $e_i, e_j$are synergistic, whereas $VDSM(i,j) < 0$ indicates that e_i, e_jinterfere with each other in a negative way.

We note that value is defined at benefit at cost, and therefore both benefit and cost considerations are included in the definition of V-DSM. In some situations it will be useful to decompose the V-DSM in two matrices, the benefit DSM or B-DSM and the cost DSM or C-DSM. These two matrices are defined below.

Definition (B-DSM): The benefit design structure matrix or B-DSM is an *m x m* symmetrical matrix where element *(i,j)* of the matrix contains a numerical value representing the strength of the positive or negative interactions between elements *i* and *j* in terms of benefit. In other words, *B-DSM(i,j)* is the extra benefit obtained from putting elements *i* and *j* in the same subset:

$$BDSM(i,j) = BDSM(j,i) = B(\{e_i, e_j\}) - B\left(\{\{e_i\},\{e_j\}\}\right) = B(\{e_i, e_j\}) - B(\{e_i\}) - B(\{e_j\})$$

where $B(P_i)$ is the benefit of an architecture represented by partition P_i. $B(P_i)$ can be computed using any evaluation rules, and in particular it can also be computed using the VASSAR framework described in the previous section. Note that $BDSM(i,j) > 0$ indicates that elements e_i, e_jhave positive synergies, $BDSM(i,j) = 0$ indicates that there is essentially an additive behavior in terms of benefit, and in some cases we can even have $BDSM(i,j) < 0$, where the benefit goes down because of the proximity between the two elements.

Definition (C-DSM): The cost design structure matrix or C-DSM is an *m x m* symmetrical matrix where element *(i,j)* of the matrix contains the difference in between a partition with two single-element subsets, and a partition with a single two-element subset:

$$CDSM(i,j) = CDSM(j,i) = C(\{e_i, e_j\}) - C\left(\{\{e_i\},\{e_j\}\}\right) = C(\{e_i, e_j\}) - C(\{e_i\}) - C(\{e_j\})$$

where $C(P_i)$ is the cost of an architecture represented by partition P_i. $C(P_i)$ can be computed using any adequate cost model.

Using for example the V-DSM as a measure of distance, the k-means or any other clustering algorithm can be applied to the initial PaP to decompose into smaller sub-problems.

Local search rules: Six different local search rules are defined, growing in level of information from completely uninformed random mutation operators to informed specialized operators that utilize the V-DSM. These six rules are described below:

- A **mutation operator** that **changes the position of one random element** into a random subset different from its original subset (Code 46 in the Appendix)
- A **mutation operator** that **swaps the positions of two random elements** taken from different random subsets (Code 47 in the Appendix).
- A **mutation operator** that **combines two random "small" subsets** to create a larger subset (Code 49 in the Appendix).
- A **rule that improves an architecture by adding a synergy**: This heuristic rule identifies the missing synergies in an architecture, selects one of these missing synergies randomly, and swaps the position of two elements in order to capture that synergy (Code 50 in the Appendix).
- A **rule that improves an architecture by eliminating an interference**: This heuristic rule identifies a current interference in an architecture, and swaps the position of two elements in order to break that interference (Code 51 in the Appendix).
- A **crossover operator** that combines characteristics of two architectures to create a new and hopefully better architecture. It does so by taking some subsets from one of the parents and then assigning the remaining elemets in a manner that ressembles the most the subsets of the other parent (Code 52 in the Appendix).

Repair operators: A repair operator is provided that projects a non-sensical architecture representation into the set of sensical architecture representations. Here the word "non-sensical" indicates an architecture representation that does not correspond to any real architecture because it does not meet enumeration constraints. In the case of PaP that use PaP-grammar A, an example of a non-sensical architecture is given by vector [1,3,4,4,4,5], since this architecture representation leaves subset 2 empty, which is not allowed. This non-sensical architecture can be turned into a sensical architecture by relabeling subsets 3,4,5 to 2,3,4 respectively so that the new architecture representation is [1,2,3,3,3,4]. The code for a generic repair operator for PaPs is provided in Code 53 in the Appendix.

2.4.3 Down-selecting Problems

2.4.3.1 Grammars

Encoding scheme: Given a set $U = \{e_1, e_2, \ldots, e_m\}$ of m elements, an architecture in an DsP consists in any subset $S_i \subseteq U$. The most straightforward encoding scheme for such a subset is an array of Booleans of size m:

$$E_1 : S_i \rightarrow [b_1, b_2, b_3, \ldots, b_m]$$

where $b_m = 1$ indicates that element I belongs to the subset, and $b_m = 0$ indicates that element e_i does not belong to the subset:

$$b_i = \begin{cases} 1 \; if \; e_i \in S_i \\ 0 \; if \; e_i \notin S_i \end{cases}$$

We identified at least two alternative encoding schemes for DsPs:

- a single non-negative integer $d \in [0, 2^m - 1]$ obtained from transforming the previous Boolean array to decima, i.e. :

$$E_2 : S_i \rightarrow d = \sum_{i=1}^{N} b_i \cdot 2^{i-1}$$

- an array of *s* non-negative integers directly representing the *id* of the instruments selected for the program, e.g. $E_3 : S_i \rightarrow [id_1, id_2, id_3, \ldots, id_{NI}]$, with $NI \leq N$, and $id_i \in [1, N] \; \forall i \in [1, NI]$.

The size of the tradespace is independent of the representation. However, there are differences between them. For example, the length of E_2 is always equal to 1, which is generally less than the length of the two others. This may lead to some advantages in terms of memory expenses. E_2 is also of constant length, whereas the length of E_3 varies across architectures. This is also an advantage since some optimization packages do not accept inputs of variable length. However, E_2 does not satisfy the desired property of "conserved vicinity", i.e. that "similar" architectures should have "similar" representations in that encoding scheme. Furthermore, it is not intuitive, and one has trouble imagining what the architecture looks like by just looking at its representation.

E_3 has the nice property that it is extremely intuitive. Its representation directly contains the list of selected instruments. However, E_3 is of variable length, which may cause problems for some optimization and representation packages. It is intuitively good in terms of the conserved vicinity property, although it poses some mathematical problems when trying to define a distance metric between two architectures, due again to its variable length

E_1 is both intuitive and convenient for optimization packages. For E_1, the Hamming distance between two architectures is exactly equal to the number of different instruments between two programs. Furthermore, a grammar based on Boolean values allows the application of classical binary programming techniques to search the architectural tradespace, as well as some metaheuristic algorithms.

In particular, genetic algorithms are naturally suited for optimization over bit vectors, because of the straightforward mapping between bits and genes. This will be discussed further in this chapter, when I introduce heuristics for searching the tradespace. All these advantages come to the expense of a larger number of variables, but this disadvantage is not critical for reasonable number of instruments (*N<100*).

Enumeration rules: Given this encoding scheme E_1, a simple set of enumeration rules R for the unconstrained DsP is provided in Code 54 in the Appendix.

The performance of this grammar to enumerate architectures was tested as in the other classes of problems, using the same configuration. As a reminder, all simulations were run and on an Intel Core 2 Duo CPU P9400 @2.4GHz with 4GB of RAM running Matlab 7.13.0.564 (R2011b) 32 bits on Windows 7 32-bits. The results are shown in Figure 19. Using this grammar on this machine, it is possible to do full factorial enumeration up to $m = 27$ in about 1hour, and $m = 30$ in about 1 day. Beyond 30 elements, full factorial enumeration becomes impossible and an alternative grammar that enumerates only a fraction of the tradespace becomes necessary.

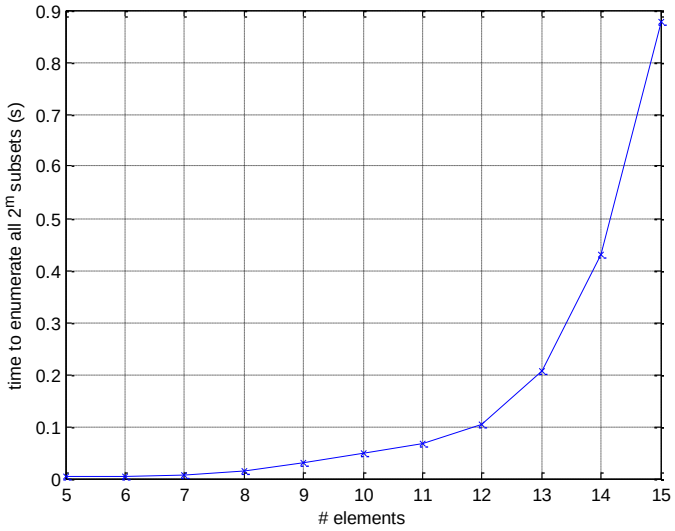
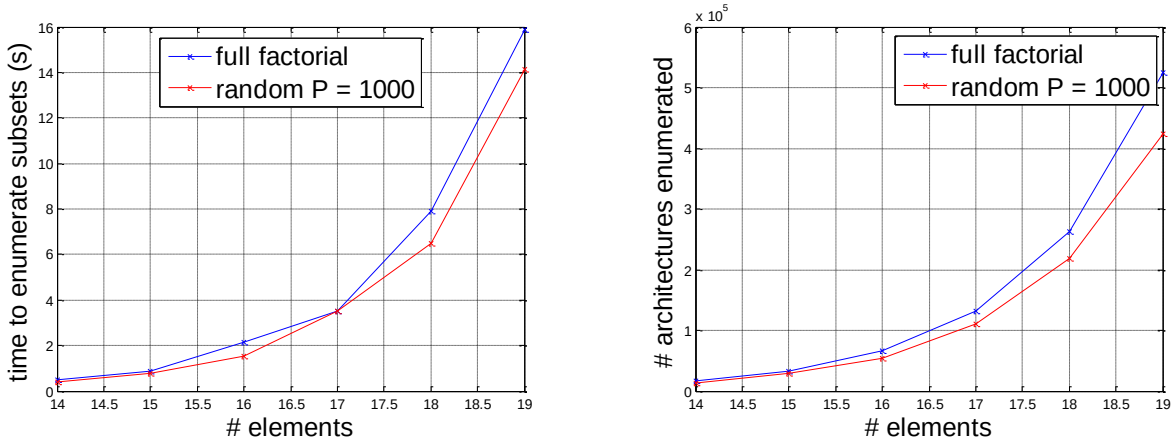


Figure 19: Performance of the grammar proposed for DsPs

Such grammar is provided in Code 55 in the Appendix. It has a few differences with respect to the full enumeration one. The relative performance of the full factorial enumeration and the partial enumeration grammars for the DsP is provided in Figure 20. All values for the partial enumeration are averages over 20 iterations. Note in particular the parameter P which allows the control of the percentage of architectures enumerated.

Empirically, we found that the value of P = 5000 is a satisfactory trade-off between computational complexity and % architectures enumerated for the range of elements between 18 and 20.



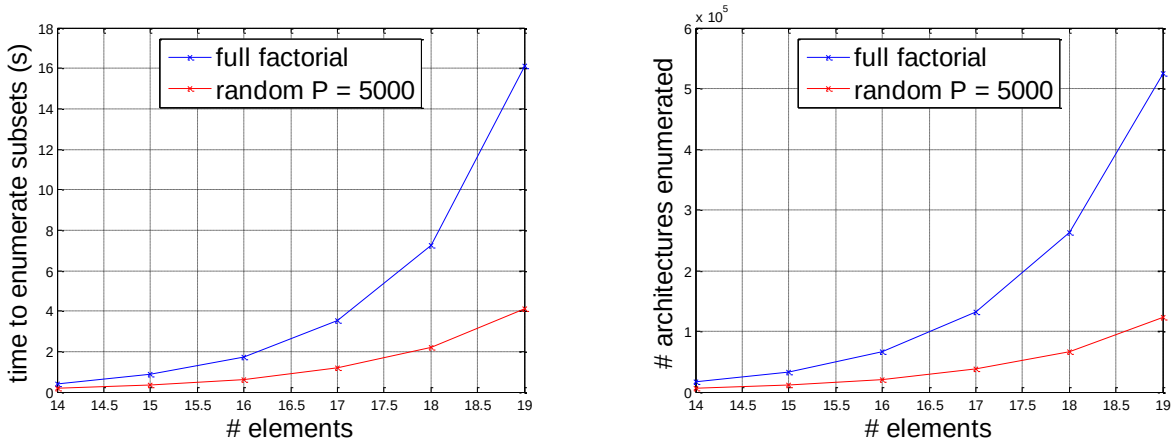


Figure 20: Performance of the two grammars proposed for the DsP for two different values of P

Enumeration constraints: We now consider four types of generic enumeration constraints for the DsP, divided in two opposite pairs:

- $OR(\{e_i\})$ enumeration constraints: this enumeration rule eliminates all subsets in which **none** of elements in $\{e_i\}$ are selected. One possible natural language equivalent of this rule is "AT LEAST ONE FROM". In logic, this is called a disjunction. It is useful to define a situation in which elements belong to groups, and at least one element of each group is required.
- $NAND\{e_i\}$ enumeration constraints: this enumeration rule eliminates all subsets in which **all** of elements in $\{e_i\}$ are selected. Note that: $NAND = \neg OR$. One possible natural language equivalent of this rule is "NOT ALL OF". In logic, this is called an alternative denial. It is useful to define a situation in which two elements are redundant.
- $XOR(\{e_i\})$ enumeration constraints: this enumeration rule eliminates all subsets in which **either more or less than one** elements in $\{e_i\}$ are selected. Note that for m = 2 only: $XOR = OR \wedge NAND$. One possible natural language equivalent of this rule is "EXACTLY ONE FROM". This is useful to define a situation in which elements belong to groups, and at least one element of each group is required, but it makes no sense to take more than 1 element from each group.
- $XNOR(\{e_i\})$ enumeration constraints: this enumeration rule eliminates all subsets **except those in which either all or none** of elements in $\{e_i\}$ are selected. Note that for m = 2 only: $XNOR = \neg XOR$. One possible natural language equivalent of this rule is "EITHER ALL OR NONE OF". This is useful to define a situation in which it really only makes sense to select certain elements as a group, so that the decision to make is whether to select the group or not. Note that this constraint reduces the size of the problem.

The 2-element truth tables of these four enumeration constraints are provided in the table below.

Table 4: Truth tables for the enumeration constraints in the DsP

S_i	$OR(S_i)$	$NAND(S_i)$	$XOR(S_i)$	$XNOR(S_i)$
00	**infeasible**	**feasible**	**infeasible**	**feasible**
01	**feasible**	**feasible**	**feasible**	**infeasible**
10	**feasible**	**feasible**	**feasible**	**infeasible**
11	**feasible**	**infeasible**	**infeasible**	**feasible**

The implementation of these constraints in CLIPS is provided in Code 56 in the Appendix. Note that this list of enumeration constraints is not exhaustive. One could consider for instance an "EXACTLY N OUT OF K" constraint in which exactly N elements from a subset of K elements need to be selected. For K = 2 and N = 1 this is equivalent to the XOR rule. Another useful rule could be "AT LEAST N OUT OF K". The implementation of such constraints from combinations or variations of the others is trivial and shown in the Appendix.

2.4.3.2 Search heuristics

Local search rules: A mutation operator that swaps the value of a bit randomly is provided in Code 58 in the Appendix. A parameter in thus rule allows the definition of the number of new architectures that are generated each time this mutation function is executed. This enables focusing the search on particularly interesting regions of the tradespace.

In addition to this non-informed mutation operator, we provide two informed local search rules that utilize the B-DSM defined in section 9.1.2.2: the first one attempts to improve the subset by completing it with an element that has high synergies with the elements already in the subset; the second one attempts to improve it by removing a redundant element from the subset. Both are provided in Code 59 in the Appendix.

Repair operators: Repair operators can be defined for the DsP so that infeasible architectures that do not satisfy enumeration constraints are projected onto the set of feasible architectures. The LHS of repair rules are almost identical to the LHS of enumeration rules, but the RHS of repair rules perform the projection instead of simply removing the architecture.

The projections are done according to the enumeration constraint that is not satisfied, as shown in Table 5. Table 5 assumes that the constraints refer to a generic set S_i of size K: $|S_i| = K$

Table 5: Repair rules for the DsP

Type of enumeration constraint that is not satisfied	**Reason why constraint is not satisfied**	**Action taken in RHS of corresponding repair rule**
$OR(S_i)$	Architecture does not contain any element from S_i	Add a random element from Si to the architecture
$NAND(S_i)$	Architecture contains all elements from S_i	Remove a random element from Si from the architecture
$XOR(S_i)$	Architecture contains 0 (case A) or N> 1 (case B) elements from S_i	Case A: Add a random element from Si to the architecture Case B: Remove N-1 random elements from Si from the architecture
$XNOR(S_i)$	Architecture does not contain either none or all elements from S_i	Case A (it contains N<K/2 elements from S_i): Remove all N elements from architecture Case B (otherwise): Add K-N missing elements from S_i to the architecture
$EXACTLY\_N\_IN\_K(S_i)$	Architecture does not contain exactly N elements from S_i	Case A (it contains L>N elements from S_i): Remove L-N random elements from architecture Case B (otherwise): Add N-L random missing elements from S_i to the architecture
$AT\_LEAST\_N\_IN\_K(S_i)$	Architecture contains L<N elements from S_i	Add N-L random missing elements from S_i to the architecture

2.4.4 Permuting Problems

2.4.4.1 Grammars

Encoding scheme: A straightforward encoding scheme for the PeP is an array of m integers between 1 and m:

$$E: O_i \rightarrow [p_1, p_2, \dots, p_m]$$

$$p_j \in [1, m]$$

Two different encoding schemes can be defined depending on the meaning of the p_j:

- $E_1: p_j = k$ means that element $e_j$is assigned to position k in O_i
- $E_2: p_j = k$ means that element $e_k$is assigned to position j in O_i

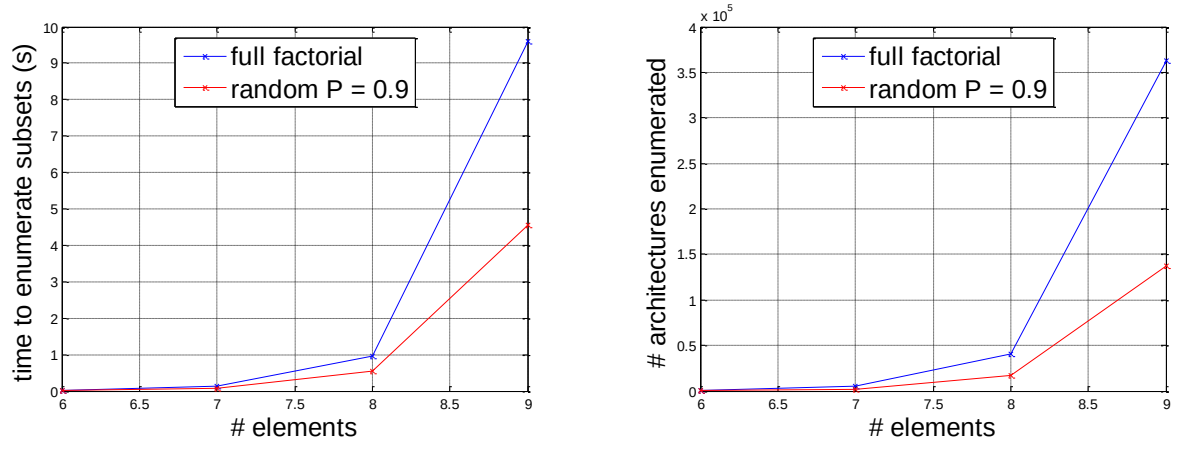
Both encoding schemes are used in this thesis, since some rules are easier to implement under E_1, while others are easier to implement under E_2. Hence, there is a need for **an operator that changes from E_1 to E_2**. The code for such operator is given in Code 60 in the Appendix, together with the template that uses the two different encodings. E_1 was arbitrarily named the sequence, and E_2 the ordering.

In order to differentiate between these two different encoding schemes, we will arbitrarily use O_i to indicate the representation of a permutation in E_1, while $O_i{}'$ will indicate the representation of the same permutation in E_2. For instance, if $O_i = [2,4,5,1,3]$ then $O_i{}' = [4,1,5,2,3]$. In this permutation, the sequence is element 2 – element 4 – element 5 – element 1 – element 3. If we want to know the position of e_j in the sequence we can also look at the jth entry in $O_i{}'$.

Enumeration rules: Given E_1 and E_2, the enumeration rule given in Code 61 in the Appendix can be used to enumerate all possible permutations of m elements.

Since *factorial(m)* grows very fast with *m*, it is required to define an alternative grammar that only enumerates part of the tradespace. This alternative grammar is presented in Code 62 in the Appendix.

The performance of these two grammars was analyzed following the same procedure as with the other classes of problems, using the same configuration. As a reminder, all simulations were run and on an Intel Core 2 Duo CPU P9400 @2.4GHz with 4GB of RAM running Matlab 7.13.0.564 (R2011b) 32 bits on Windows 7 32-bits. The results are shown on Figure 21.



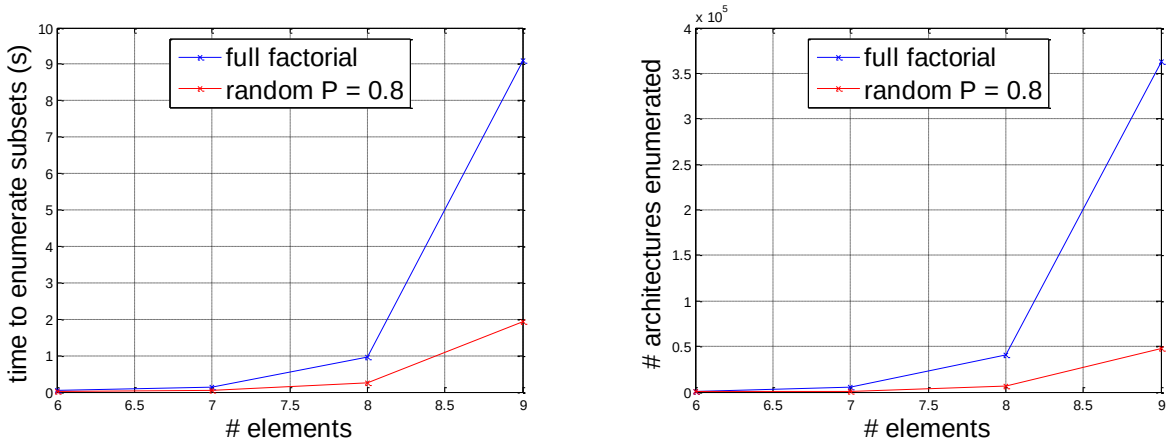


Figure 21: Performance of the two grammars for the PeP

The fraction of architectures can be controlled by adjusting the value of the parameter P. We expected a value of P to produce a fraction of architectures approximately equal to $\frac{\#archs\ enumerated}{size\ tradespace} = P^m$. Empirically, we found good correlation with this relationship: P=0.9 yields $\frac{\#archs\ enumerated}{size\ tradespace} = 46\%, 43\%, 38\%$ for m=7,8,9 (theoretical values are 48%, 43%, 39%), and P=0.8 yields $22\%, 16\%, 13\%$ (theoretical values are 21%, 16%, 13%).

The full factorial grammar takes 3.3h to enumerate all possible partitions for $m = 12$, and 43.2h for $m = 13$, on the same laptop configuration.

Enumeration constraints: In the planning instances of PePs in which, in addition to sequence, dates are assigned to tasks or events, other constraints may appear that make explicit reference to dates instead of relative positions. These rules require a**n operator that transforms a sequence of tasks with costs into a sequence of dates, given a certain budget profile**. Such operator is provided in Code 63 in the Appendix. This function assumes that readiness date for element *i + 1* is equal to readiness date for element *i* plus the cost of the element *i* divided by the budget at time t.

We provide seven possible hard enumeration constraints in generic PePs: BEFORE, AFTER, BETWEEN, NOT-BETWEEN, CONTIGUOUS, NON-CONTIGUOUS, and SUBSEQUENCE constraints. All of these utilize a few generic functions to check precedence between elements in a sequence or ordering. These functions are provided in Code 64 in the Appendix. Note that these hard constraints can also be used as search rules with the purpose of reducing the size of the tradespace. Furthermore, equivalent rules can be defined that look at relative dates of elements instead of looking at positions. The code of these rules is almost identical to the rules shown below and therefore it is not included. The seven types of hard constraints for tradespace size are discussed below.

- $BEFORE(e_i, \{e_j\})$ constraints (Code 65 in the Appendix): this enumeration rule eliminates all permutations in which $e_i$does not precede all the elements in $\{e_j\}$. It is useful to define pre-conditions between tasks or events.
- $AFTER(e_i, \{e_j\})$ constraints (Code 66 in the Appendix): this enumeration rule eliminates all permutations in which $e_i$does not succeed all the elements in $\{e_j\}$. This is the opposite rule of $BEFORE(e_i, \{e_j\})$. It is useful to define post-conditions between tasks or events.
- $BETWEEN(e_i, \{e_j, e_k\})$ constraints (Code 67 in the Appendix): this enumeration rule eliminates all permutations in which $e_i$does not appear between e_j and e_k. Note that this constraint does NOT enforce the subsequence e_j, e_i, e_k since: a) the order between e_j and e_k can be reversed; there may be other elements inside the interval e_j, e_k. Hence for example, $O_1 = [e_j, e_i, e_m, e_k]$ and $O_1 = [e_k, e_l, e_i, e_m, e_j]$ would both satisfy the constraint $BETWEEN(e_i, \{e_j, e_k\})$. It is useful to define simultaneous pre- and post-conditions between tasks or events.
- $NOT\_BETWEEN(e_i, \{e_j, e_k\})$ constraints (Code 68 in the Appendix): this enumeration rule eliminates all permutations in which e_i appears between e_j and e_k. This is the opposite rule of $BETWEEN(e_i, \{e_j, e_k\})$.
- $CONTIGUOUS(\{e_j\})$ constraints (Code 69 in the Appendix): this enumeration rule eliminates all permutations in which $\{e_j\}$ are not contiguous. Note that there are several possible orderings that satisfy contiguity of a subset of elements.
- $NON\_CONTIGUOUS(\{e_j\})$ constraints (Code 70 in the Appendix): this enumeration rule eliminates all permutations in which $\{e_j\}$ are contiguous. This is the opposite rule of $CONTIGUOUS(\{e_j\})$.
- $SUBSEQUENCE(O_j)$ constraints (Code 71 in the Appendix): this enumeration rule eliminates all permutations which do not contain O_j as a subsequence, i.e. all permutations O_i such that $O_j \nsubseteq O_i$.

2.4.4.2 Search heuristics

Tradespace size reduction rules: First, we note that, as mentioned earlier, it is possible to use the seven enumeration rules as search hard constraints. In addition to these seven rules, we propose one rule to fix the position of a certain element FIX-POSITION:

- $FIX\_POSITION(e_i, j)$ constraints (Code 72 in the Appendix): this tradespace size reduction rule eliminates all permutations in which element *i* does not appear in position *j*.

Instead of deterministic rules, a few "fuzzy" constraints can also be applicable to some instances of the PeP. These fuzzy constraints are softer in nature because instead of enforcing a certain position for an element, they enforce a range of positions.

- $BY\_BEGINNING(\{e_j\})$ constraints (Code 73 in the Appendix): this enumeration rule eliminates all permutations in which $\{e_j\}$ do not appear in "the beginning" of the sequence. Note that there are several possible orderings that satisfy contiguity of a subset of elements.
- $BY\_MIDDLE(\{e_j\})$ constraints (Code 74 in the Appendix): this enumeration rule eliminates all permutations in which $\{e_j\}$ do not appear in "the beginning" of the sequence. Note that there are several possible orderings that satisfy contiguity of a subset of elements.
- $BY\_END(\{e_j\})$ constraints (Code 75 in the Appendix): this enumeration rule eliminates all permutations in which $\{e_j\}$ do not appear in "the beginning" of the sequence. Note that there are several possible orderings that satisfy contiguity of a subset of elements.

Local search rules: Amongst the non-informed local search rules, we define **a mutation operator** that **swaps the positions of two elements**. Its CLIPS code is given in Code 76 in the Appendix. Concerning informed local search rules, we define a crossover operator, given in Code 77 in the Appendix.

Repair rules: Finally, we consider the need for repair rules for the PeP. Since the grammar we proposed for PePs avoids the generation of infeasible architectures, the only possibility left is that infeasible architectures are created by crossover and mutation operators. However, the mutation and crossover operators that we introduced generate feasible architectures by construction. Consequently, contrary to what happened with PaPs, it is not necessary to have repair rules for PePs.

2.5 Summary and discussion

In this chapter, we have presented a framework to solve any KI SAP. The main idea behind the framework is that of there are different bodies of knowledge required to solve an SAP, and these can be physically separated by using a RBES. In particular, there is domain-independent knowledge on one hand, which is related to the search, optimization, and decision making. On the other hand, there is domain-specific knowledge, which concerns the domain of application (e.g., EOSS). Furthermore, part of the domain-independent knowledge is common to all SAPS, part is common to a certain class of SAPs, and part is specific to each instance of SAP. Similarly, part of the domain-specific knowledge is the same for all SAPs in a given domain, and part of it is really instance-specific.

Based on this observation, the framework: 1) develops a generic search algorithm that implements the domain-independent tasks that are always the same for any class and instance of SAP; 2) identifies five classes of SAPs, and provides heuristic rules for each class that are incorporated into the algorithm in order to solve the tasks that are class-dependent; 3) is flexible enough to allow easy accommodation of domain-specific rules.

This process is illustrated in Figure 22, in which we can also see how the different components of this framework are interconnected. Note that the VASSAR framework to assess architectural value using approximate evaluation rules is described in more detail in the next chapter.

The framework exploits the properties of RBES to ensure traceability and to provide a more natural environment to model emergent behavior than classical procedural languages. This last point will become more apparent in the next chapter as we describe emergence rules in the VASSAR framework. The importance of emergent behavior has also been emphasized throughout this chapter as the different classes of SAPs (and the corresponding rules defined in the appendix) were introduced. In particular, we introduced the concepts of B-DSM and C-DSM to capture bilateral interferences between elements. The B-DSM and the C-DSM are used by several heuristic rules to capture element interactions.

One of the main discussion points concerns the scope and applicability of the framework. The framework has been introduced as a general problem solving framework for system architecting and as such it can be used to solve any SAP. In particular, note that it contains the assignment class of architectural problems, and therefore it is at least as broad in scope as its predecessors ADG and OPN. Moreover, several other classes of SAPs have been defined, and therefore it should exceed their scope as they model more classes of SAPs. However, the framework was categorized as a knowledge-intensive (KI) framework.

Hence, the framework is most useful for problems that require handling a large body of knowledge. This does not mean that the framework will not work for non-KI problems; it just means that trading computational efficiency or exactness against flexibility and transparency is probably not worth in those cases.

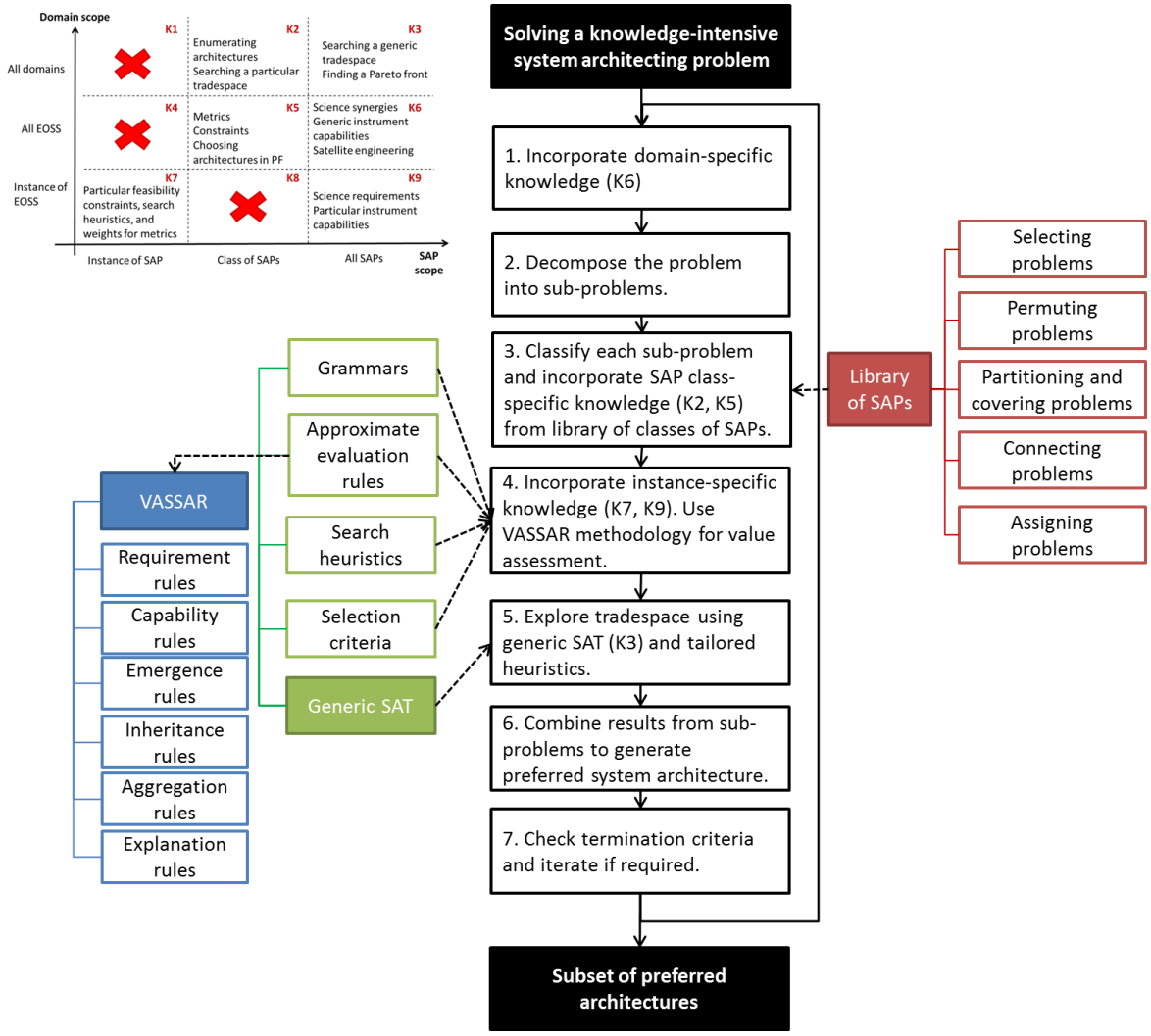


Figure 22: Overview of the framework for solving knowledge-intensive SAPs using rule-based systems

3 VASSAR: A methodology for Value Assessment in System Architecting using Rules

3.1 Introduction

One of the key components of the framework for solving KI SAPs is the use of approximate evaluation rules that replace expensive simulations in the assessment of architectural value. For example, instead of running an expensive simulation on dedicated mission analysis software, one can compute the average revisit time of a constellation of sufficiently wide-swath instruments flying in LEO as 12h divided by the number of satellites. A polynomial can also be easily found that provides the worst case revisit time in the same circumstances of wide-swath, as a function of the number of planes, and the number of satellites per plane. These simple rules are powerful because they contain most of the knowledge of the expensive simulations and can be executed in a fraction of a second.

Approximate evaluation rules assess the value of a system architecture by linking architectural variables to metrics. Approximate evaluation rules can be developed ad hoc for any system. However, this chapter section introduces a general methodology that can be used to assess value – benefit AND cost - to any KI SAP.

VASSAR is a methodology that aims to provide a quantitative or semi-quantitative assessment of the value that a system architecture provides to stakeholders. The main assumptions behind VASSAR are listed below:

- **Value is defined as satisfaction of stakeholder needs**. This is a common assumption in system architecture.
- "Fuzzy" stakeholder needs elicited from stakeholders are used as an input to the methodology. The methodology assumes that **a list of stakeholders and their (fuzzy) needs is readily available**. An example of a fuzzy need may be "to reduce the uncertainty in the amount of water that is in the lands". This thesis does not provide support for the elicitation of stakeholder needs.
- **A list of fuzzy system requirements, inferred from the set of fuzzy stakeholder needs, is readily available**. An example of a fuzzy requirement that follows from the previous need is that "the system shall be able to measure soil moisture with an accuracy of 4% or better, a spatial resolution of 10km or better, and a temporal resolution of 2-3 days or better". This thesis does not provide support for this process of logical decomposition either.

- **The body of knowledge required to infer fuzzy system capabilities from the system architecture is available**. Fuzzy system capabilities concern the functions and related performance that the system architecture is capable to perform. For example, an L-band radiometer can measure soil moisture with high accuracy and low spatial resolution, whereas an L-band radar can measure soil moisture with lower accuracy but better spatial resolution. The numerical values of accuracy, temporal, and spatial resolution can be computed once architectural variables such as orbital parameters have been set. The methodology assumes that this body of knowledge can be elicited from a group of experts. As stated earlier, this thesis does not provide explicit support as to how to conduct this knowledge elicitation process.
- **Requirements and capabilities can be expressed in the form logical rules and facts respectively**, using a common "language" (i.e., a common set of data structures or templates). Although this statement is always true because RBES are Turing complete, the process of expressing requirements and capabilities in the form of rules and facts can be more or less burdensome depending on the characteristics of the system at hand. We note however that, as explained in the section describing the history of RBES, Newell and Simon discovered in the late sixties that expert knowledge is stored in the brain in the form of chunks of information that fit very well the structure of a logical rule (Newell & Simon, 1972).
- The final and perhaps most important assumption of the methodology is that **value of a system architecture can be assessed by systematically comparing fuzzy system requirements to fuzzy system capabilities**. Given this assumption, the value assessment process becomes a pattern matching process between what stakeholders need (requirements) and what the system architecture can do (capabilities). While this assumption can seem innocuous, it has profound implications in the outcome of the value assessment process. For example, the methodology will be unable to discern between two architectures that differ only in capabilities that are not captured in the set of requirements. If one architecture can provide a 4% accuracy soil moisture product, and another architecture can provide a 2% soil moisture product, but the requirement only specifies a need for a 4% accuracy requirement, then the two architectures will have the same score for this particular requirement.

Based on these assumptions, the VASSAR methodology utilizes different types of rules and facts to assess architectural value, as illustrated in Figure 23. To a given system architecture in the top right corner of Figure 23, corresponds a fuzzy value metric in the top left corner. This fuzzy value metric and the corresponding explanations are computed in several steps.

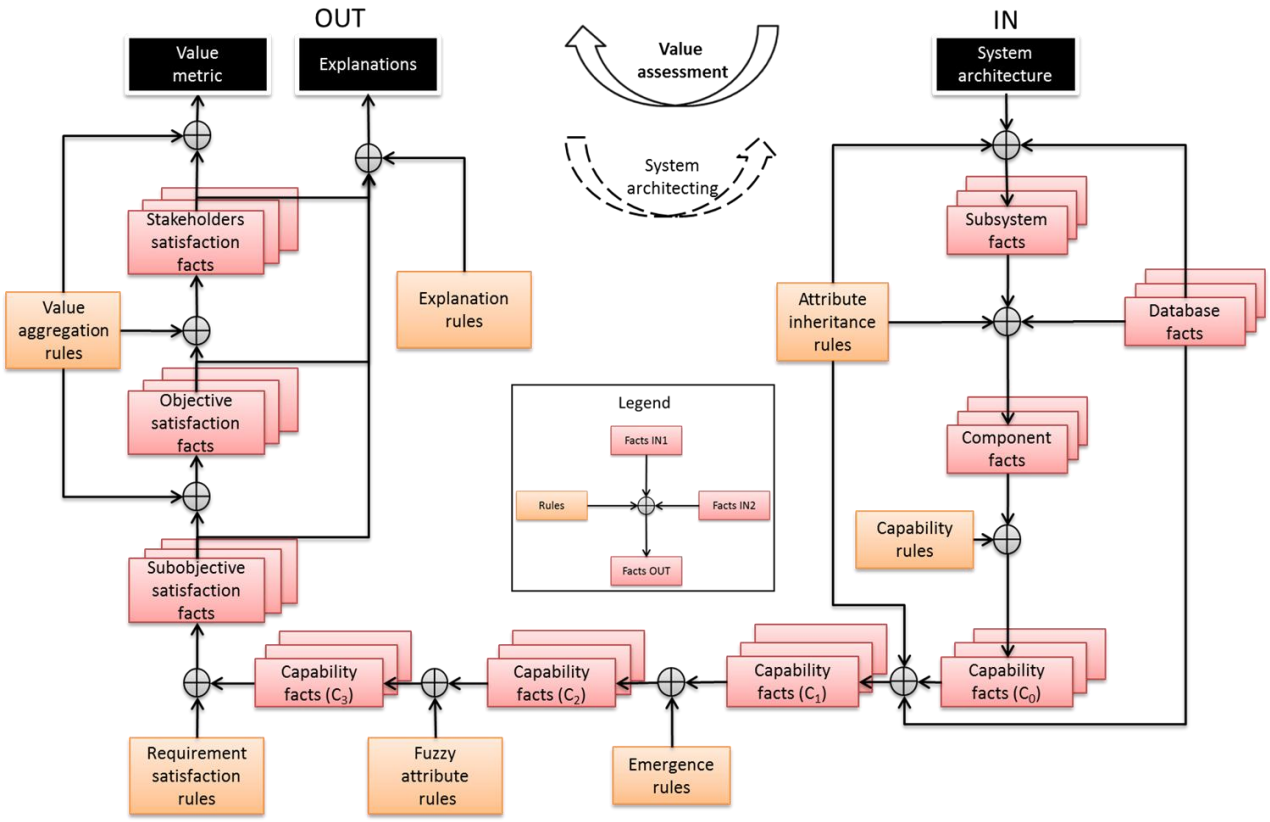


Figure 23: The VASSAR methodology for value assessment of system architectures using rules

We will use the computing needs of a family as a toy example to illustrate the methodology. In the VASSAR methodology, we labeled "system", "subsystems", and "components" the elements of form at levels 1, 2, and 3 following the recommendations in the INCOSE systems engineering handbook (Haskins, 2006). In this toy example:

- the **system** will designate the set of computing assets (e.g., one desktop and one laptop, or three laptops);
- the **subsystems** will be each of the assets;
- the **components** will include the processor, the RAM, the hard disk amongst others.

Hence the "**architecture**" of the system in this toy example will be the given by the characteristics (i.e., number and type of processors, memory, storage options, and network cards) of each of the computing assets.

For example, one particular computer architecture ARCH1 could consist of a desktop featuring a quad-core processor at 3.5GHz with 8M cache, 16GB of RAM, 1.5TB of hard-risk at 7200rpm, a 3GB dedicated graphics card, and an advanced WiFi card, and a laptop featuring a dual-core processor at 2.4GHz with 6M cache, 160GB of solid-state memory, no dedicated graphics card, and a standard WiFi card.

The framework also requires the definition of **capabilities**. In the case of computers, possible capabilities include "running expensive Matlab simulations", "playing graphics-intensive games", or "performing good-quality video-calls". Each of these capabilities allows the definition of several performance attributes. For example, in the case of the Matlab simulations, one can define performance metrics related to the time that it takes to solve a given problem.

Different users or stakeholders can define different requirements in terms of these capabilities. For instance, let us consider that the computer is going to have three different users in a family: a father who is a mechanical engineer, his daughter who is a grad student in science, and his son who is in high school. The father may have special requirements for the computer because he needs to run a CAD software on it. The daughter may have special requirements because she needs to run expensive Matlab simulations, and videochat with her boyfriend in another continent. Finally, the son may have special requirements because he wants to play on-line graphic-intensive games.

The steps of the methodology as applied to this toy example are briefly described below. The legend on Figure 23 shows how to interpret the diagram. The circle with a cross inside represents a point in the procedure where the rules engine is run, i.e., where the inference engine performs the pattern matching between all rules and facts in working memory and executes the right hand sides of all the rules that are activated.

1) S**ubsystem facts are asserted** from the system architecture and a set of rules called the attribute inheritance rules. In the ARCH1 example, two facts will be asserted, one for the desktop and one for the laptop, and their characteristics will be inherited from the system architecture, and/or possibly from a database of computers.
2) **Component facts are asserted** from: a) the subsystem facts; b) a database containing information about available components amongst other data; c) more attribute inheritance rules. In the ARCH1 example, a fact will be asserted for each component of each computer. Attributes of the components can be inherited from the computers, or from the component database.

3) **Capability facts are asserted** from component facts and capability rules. In the ARCH1 example, capability rules link computer components to capabilities. For instance, the dedicated graphics card enables the gaming capabilities, and the presence of at least 8GB of RAM enables the execution of the expensive Matlab simulations. This first set of capability facts C_0 contains very little information about the performance with which each function is realized.
4) **The attributes of these capabilities are inherited** from the subsystem attributes and the component attributes through attribute inheritance rules, possibly using information from the database. For example, the time that it takes to execute a given Matlab simulation can be estimated from the CPU speed. This yields a second set of capability facts C_1.
5) **A new set of capability facts C_2 is created through the emergence rules, which modify and generate new capabilities from combinations of existing capabilities**. Capability rules that link components to capabilities alone often fail to capture an important part of the value provided by the system that comes from interactions between capabilities, i.e., emergent behavior. For example, the performance related to the gaming capabilities of a computer will emerge from a combination of factors including the RAM, the memory of the graphics card, and the screen resolution amongst others.
6) **Fuzzy attribute rules transform back and forth between numerical and fuzzy attributes**. This step is necessary because some requirements are expressed numerically (e.g., execution time < 20s), while others use fuzzy values (e.g., "good quality" videoconference). Thus, fuzzy attribute rules make the translation between fuzzy statements and numerical statements when necessary, using interval analysis.
7) **Subobjective satisfaction facts are asserted through the application of requirement satisfaction rules**. These facts store information that can later be used to trace the numerical value to the reasons behind it. More precisely, it contains information about which combination of components is at the origin of this activation, or in case of partial satisfaction, the attribute or attributes that are missing from the achieved capabilities.
8) For example, a subobjective concerning the gaming capability may be partially satisfied by the laptop because although the RAM and screen resolution are good enough, the quality of the graphics card is not sufficient to provide 3D gaming experience. At this point, a list of facts is available that assesses how well each of the subobjectives is satisfied by the current system architecture.

9) **Value aggregation rules aggregate individual subobjective satisfaction facts into a single metric, which represents the value of the system architecture**. While in some cases the information about individual subjective satisfaction could be enough, in many others it will be necessary to condense this information into a handful of numerical values that can be plugged in in any search of optimization algorithm, or more generally any decision making process. This step is accomplished by using value aggregation rules, which compute one or a handful of value metrics from the initial subobjective satisfaction facts. These rules may capture for example the relative importance of the different requirements of the father, daughter, and son, as well as the relative importance of the father, the daughter, and the son to the family.

10) **Explanations behind these value metrics are created by applying the explanation rules to the subobjective, objective, and stakeholder group satisfaction facts**. These explanations are then provided to the user as a complement to the metric. This ability to provide the explanation of the results to the user is a key capability of this methodology and is enabled by the use of rule-based expert systems, which have built-in explanation facilities as explained in Chapter 1.

In the rest of this chapter, each of these steps is described in more detail. The toy example with the computer assets is used to support the explanations all throughout the chapter.

3.2 Requirement satisfaction rules

Description: Requirement satisfaction rules link capabilities to stakeholder subobjective satisfaction. Their general structure and two examples are shown in Code 8.

```
 General structure (in natural language):
(Define-rule full-satisfaction-of-subobjective-X
"Description of the rule: conditions for full satisfaction of subobjective X"

IF there is a Capability (of type T) (performed by a certain combination of
components who1) (with performance attribute A1 = x1) (with performance
attribute A2 = x2) (+ requirements concerning other attributes)
AND (Performance x1 is val1 or better))
AND (Performance x2 is val2 or better))

=>(THEN)

ASSERT a fact indicating full satisfaction of subobjective X (taken by who1))

Example (in the CLIPS language):
(defrule REQUIREMENTS::subobjective-dad1-3-nominal
   "Conditions for full satisfaction of subobjective daughter1-3"
```

```
(Matlab-simulations (max-memory-GB ?x1) (running-time-sec ?t1)
(thanks-to ?whom))
(test (>= ?x1 4))
(test (<= ?t1 10))

 =>

 (assert (REASONING::fully-satisfied (subobjective daughter1-3)
(objective "Matlab simulations") (thanks-to ?whom)))
)
```

Code 8: General structure and examples of requirement satisfaction rules

The framework distinguishes between nominal requirement satisfaction rules and degraded requirement satisfaction rules. **Nominal requirement satisfaction rules** express the fuzzy requirements needed for a given subobjective to be fully satisfied. If the capabilities described in a nominal requirement satisfaction rule are fulfilled by the architecture with the required attributes, then the rule asserts a fact that traces the full satisfaction of this subobjective. This fact contains information that will be used later by both the value aggregation rules, and the explanation rules.

Typically, only one nominal requirement satisfaction rule exists for each subobjective. However, additional nominal requirement satisfaction rules can be added to account for alternative capabilities that stakeholders consider as full satisfaction of the same subobjective. For example, a user may tolerate a longer execution time if more memory is available to handle larger problems. In the case of EOSS, a subobjective concerning the measurement of atmospheric CO may be fully satisfied by a gas filter correlation radiometer (e.g. EOS MOPITT) or by a Fourier transform spectrometer (e.g. EOS TES), and atmospheric chemists may accept both options as full satisfaction of their requirements in terms of this subobjective (Luo et al., 2007).

Degraded cases in which a subobjective is somewhat but not completely satisfied are captured in **degraded requirement satisfaction rules**. The number of possible degraded cases that lead to partial satisfaction can be very large. However, at least in the case of Earth observing systems, a handful of degraded cases is enough to cover most realistic situations in practice.

Discussion: The scores that are assigned to partial satisfaction rules are very important, especially when the tool is used to assess the value of system architectures in cardinal scales as opposed to ordinal scales. These scores are elicited from experts through an interview process, and therefore they are subject to the known limitations of interviews. For example, it is not impossible that experts disagree on their assessment of a particular set of capabilities. In theory, a formal process from social sciences such as the Delphi method could be used to foster stakeholder consensus or compromise (Dalkey & Helmer, 1963).

However, because of time constraints, no formal method from the social sciences was applied to obtain these scores in the context of this thesis. Furthermore, this points out a limitation of the methodology: if the problem at hand uses several hundreds of degraded requirement satisfaction rules, it means that it will have several hundreds of parameters that are, in theory, also degrees of freedom of the tool, as one could in principle assign any value to these scores. In order to alleviate this problem, Likert-like scales (Jamieson, 2004)can be used that only allow a few cardinal values of subobjective satisfaction (e.g., all value, most, some, marginal, none)

Another question may arise from the small number of degraded requirement satisfaction rules. Because of the RBES essentially is a pattern matcher, if a set of capabilities does not match any of the degraded requirement satisfaction rules, the system will not assert any partial satisfaction facts concerning that particular subobjective, and thus the score of the architecture with respect to that subobjective will be zero. The situation is thus similar to having a response surface where only the values of a function on a few points on a surface are known, and the rest are interpolated. An opportunity for future work would thus be to add a statistical machine learning layer on top of the RBES, so that some sort of interpolation could be applied to account for degraded cases that are not described using for example neural networks.

3.3 Value aggregation rules

Description: Thus far, we have explained how to assess the level of satisfaction of a particular subobjective using requirement satisfaction rules. In theory, each subobjective could be used as an independent metric. However, in most real life situations, the number of subobjectives is too large, and a scheme for aggregating the values of different subobjectives becomes necessary.

Value aggregation rules are used to reduce the dimensionality of the objective satisfaction space, from potentially hundreds of subobjectives to a handful of stakeholder group satisfaction metrics, or even a single metric that captures overall architecture value. Typically, this handful of metrics will contain at least one metric for lifecycle cost and one or several metrics for performance or benefit.

The general structure of a value aggregation rule is provided in Code 9.

```
General structure (in natural language)
(Define-rule aggregation-of-objective-X
"Description of the rule: aggregation of all subobjectives in objective X"

IF there is a (subobjective satisfaction fact (of subobjobjective X1)
(of objective X) (with score s1) (taken by who1))
AND there is a (subobjective satisfaction fact (of subobjobjective X2)
(of objective X) (with score s2) (taken by who2))
AND there is a (subobjective satisfaction fact (of subobjobjective X3)
(of objective X) (with score s3) (taken by who3))
```

```
=>(THEN)

ASSERT a fact indicating satisfaction of objective X (with score (generic
combinator of scores s1 s2 s3)) (taken by “who1 and who2 and who3”)
)

Example (in the CLIPS language):

(defrule REQUIREMENTS::aggregation-objective-ECO6
   "Aggregates satisfaction of all subobjectives in objective ECO6"

 (REASONING::subobj-satisfied (subobjective ECO6-1) (objective ECO6) (from
?whom1) (score ?s1)))
 (REASONING::subobj-satisfied (subobjective ECO6-2) (objective ECO6) (from
?whom2) (score ?s2)))
 (REASONING::subobj-satisfied (subobjective ECO6-3) (objective ECO6) (from
?whom2) (score ?s3)))

   =>

(assert (REASONING::obj-satisfied (objective ECO6) (from (str-cat ?whom1 " and
" ?whom2 " and " ?whom3)) (score (weighted-average ?s1 ?s2 ?s3))))
   )
```

Code 9: General structure and example of a value aggregation rule

Different implementations of the **generic combinator** shown in Code 9 can be used. The most straightforward implementation is perhaps a simple weighted average of several subobjectives. More sophisticated operators such as the ordered weighted averaging aggregation operator developed by Yager can also be used (Yager, 1988). More generally, any arithmetic and logical operation on one or several subobjective satisfaction facts is allowed.

Discussion: The problem of value aggregation essentially is a multiple criteria decision making problem, for which several methodologies are well known. A multi-objective optimization approach based on the approximation of the Pareto front will fail to reduce the tradespace to a manageable number of architectures, due to the large dimensionality of the objective space. A goal programming approach (Ignizio, 1983) based on the minimization of some distance to a target vector in the objective space can also be utilized, but the problem lies then on the a priori selection of the target vector.

An isoperformance approach (O. L. de Weck & Jones, 2006) has the same problem, on top of the unknown and potentially large dimensionality of the isoperformance set. Other similar methods are based on distance to the utopia point, see for example TOPSIS (Lai, Liu, & Hwang, 1994). A multi-attribute utility theory approach (Keeney & Raiffa, 1993) will come to the price of subjectivity as it is hard - and cumbersome - to elicit relative weights between competing objectives from decision makers.

Another class of these formal approaches is based on the systematic use of pairwise comparisons between the different options. Weights are then deduced from the results of all the pairwise comparisons using some kind of formal mathematical algorithm. See for example Saaty's Analytic Hierarchy Process, in which relative weights are the eigenvalues of the pairwise comparison matrix (Saaty, 2008). Despite their subjectivity, decision making processes based on weighted average operators continue to be widely used in industry and academia, perhaps due to their simplicity and transparency.

3.4 Capability rules

Description: Capability rules are at the core of the methodology and contain an important part of the expert knowledge, as they link components to capabilities. Their general structure and an example are provided in Code 10.

Capability rules may, but do not necessarily have to, take care of all the capability attributes. Instead, attribute inheritance rules can be used to inform capability attributes from subsystem and component attributes. Furthermore, capabilities that emerge from combinations of existing capabilities are captured by emergence rules.

Discussion: At a first glance, the structure of capability rules may seem to restrict the scope of applicability of the whole framework, since for many systems, the mapping between components and capabilities is very complex. However, the framework remains general because capability rules can contain more than one component and be arbitrarily complex, as shown in the example.

```
General structure (in natural language):
(Define-rule capabilitites-of-component-X
"Description of the rule: defines capabilities of component X"

IF there is a (manifested component (of type T) (named X) (with attribute1 x1)
(with attribute2 x2))
AND (Attribute1 x1 is val1 or better))
AND (Attribute2 x2 is val2))

=>(THEN)

ASSERT a (Capability fact (of type T1) (from X) (with)) (taken by "who1 and
who2 and who3")
)

Example (in the CLIPS language):

(defrule CAPABILITIES::high-end-gaming
   "Asserts the capabilities of high end gaming"

(COMPONENTS::RAM (total-memory-GB# x) (memory-speed-MHz# y))
(COMPONENTS::graphics-card (dedicated-memory-GB# z))
```

```
(COMPONENTS::monitor (screen-size-in u) (screen-resolution 1920x1080))

(test (>= x 16))
(test (>= y 1333))
(test (>= z 3))
(test (>= u 19))

=>
(assert (CAPABILITIES::Gaming (ThreeD yes) (graphic-quality High-end)))
   )
```

Code 10: General structure and example of a capability rule

Another point worth mentioning concerns the difference between capability rules and emergence rules. It is clear that part of the actual system emergence is captures in the capability rules. For example, the rule showed in Code 10 encapsulates the emergent behavior between RAM, a monitor and a graphics card to yield a high end gaming capability. Emergence rules will capture the part of the emergent behavior that results from combinations of capabilities, as opposed to combinations of components. The balance between the number of capability rules and the number of emergence rules may differ widely across systems, but the general methodology will remain valid.

3.5 Attribute inheritance rules

Description: Attribute inheritance rules encapsulate the "downward" dependences between subsystem, component, and capability attributes ("downward" because they descend in the form hierarchy from the system to components). Subsystem attributes can be inherited from the system architecture, or from a database. Component attributes can be inherited from a subsystem, or from a database. Capability attributes can be inherited from a component, or from a combination of component and subsystem attributes. The general structure of an attribute inheritance rule and an example are shown in Code 11.

```
General structure (in natural language):
(Define-rule inherit-component-attribute-X-from-database
"Description of the rule: inherits component attribute X from parent database"

IF there is a Component in the SYSTEM (of type T) (with attribute1 unknown)
(with attribute2 unknown)
AND there is a Component in the DATABASE (of type T) (with attribute1 att1)
(with attribute2 att2)

=>(THEN)

MODIFY the fact concerning the SYSTEM component (with attribute1 att1)
(with attribute2 att2)
)

Example (in the CLIPS language):
(defrule CAPABILITIES::inherit-CPU-from-database
"Inherits CPU data from database of CPUs"
```

```
?x <-(COMPONENT::Processor (Make ?make) (Model ?model) (CPU-speed-GHz# nil))
 (DATABASE::Processor (Make ?make) (Model ?model) (CPU-speed-GHz# ?speed&~nil))

=>
 (modify ?x (CPU-speed-GHz# ?speed)))
```

Code 11: General structure and example of an attribute inheritance rule

More involved attribute inheritance rules are also possible. For example, the temporal resolution of a measurement will depend on both the orbit and the instrument angular resolution. The level of sophistication of this dependence can be customized. For instance, it can be modeled using simple geometric considerations, spherical trigonometry, or even a dedicated simulation tool like AGI's STK.

Discussion: At this point, the boundary between capability rules, emergence rules, and attribute inheritance rules that regulate inheritance of attributes from components to capabilities may seem a little fuzzy. Capability rules are intended to assert capability facts from components, but they only take care of a few attributes. Attribute inheritance rules do not assert any facts, they only modify them to inform the value of certain capability attributes.

Beyond the logical rationale, this classification allows an easier automation of both types of rules from spreadsheets. As for emergence rules, they typically modify or assert facts from combinations of existing capabilities as opposed to asserting facts from existing components.

Another question that may arise concerns the possible existence of upward inheritance of attributes, i.e. the bottom-up inheritance of attributes from components to the system. In the framework, this kind of behavior does not exist, because attributes that are susceptible of being inherited upwards are relevant to the calculation of the value metric and therefore they are declared as capabilities rather than being simple attributes. A typical example is system cost, which to a first approximation is a sum of the costs of the subsystems and therefore could be considered an attribute inherited upward. Instead of having cost as a system attribute that is inherited upwards from components and subsystems, we define capabilities and requirements concerning system lifecycle cost. Capability rules can then estimate system lifecycle cost from components.

For example, a cost estimating relationship for a certain subsystem or component as a function of one or more of its attributes will be encoded as a capability rule. Emergence rules may also be used to model non-linear multi-component cost penalties. This way of treating cost in the same way as other capabilities is a more elegant solution, and allows the definition of related objectives and subobjectives if required (e.g. stakeholders are satisfied if system cost is kept below a certain threshold).

3.6 Emergence rules

Emergence rules are a double-edged sword in the framework. On one hand, they capture fundamental emergent behavior without which it is not possible to adequately model a system, and in some cases they are actually extremely powerful in generating emerging capabilities. On the other hand, precisely because they are very powerful in the generation of new capability facts, they may lead to computational complexity issues, and they also render the code development process a bit more unpredictable and hard to debug.

Description: Emergence rules modify existing capabilities and create new capabilities from interactions between existing capabilities. Their general structure and examples are shown in Code 12.

Discussion: We opened the subsection on emergence rules by raising a concern about their ability to become double-edged swords in the framework. Indeed, during the development of the framework, emergence rules were systematically at the origin of our biggest successes and failures. It is thanks to emergence rules that we could explore truly unforeseen combinations of measurements that would otherwise be very hard to explore.

```
General structure (in natural language):
(Define-rule emergence-of-new-capability
“Description of the rule: emergence of a new capability of type T3 from two
capabilities of types T1 and T2”

IF there is a Capability (of type T1) (performed by a certain combination of
components who1) (with performance attribute A1 = x1)
AND (Performance x1 is val1 or better))
AND there is a Capability (of type T2) (performed by a certain combination of
components who2) (with performance attribute A2 = x2)
AND (Performance x2 is val2 or better))

=>(THEN)

ASSERT a new Capability (of type T3) (with performance attribute A1 = x1)
(with performance attribute A2 = x2) (performed by “who1 and who2”)
)

Example (in the CLIPS language):

(defrule EMERGENCE::parallel-computing
   "If we have computing capabilities on two different assets, plus an internet
connection, we can parallelize computations"

(CAPABILITIES::Matlab-simulations (max-memory-GB ?x1) (running-time-sec ?t1)
(thanks-to ?whom1))
(CAPABILITIES::Matlab-simulations (max-memory-GB ?x2) (running-time-sec ?t2)
(thanks-to ?whom2&~?whom1))
(CAPABILITIES::Network-connection (between ?computers))
```

```
(test (str-contain ?computers (create$ ?whom1 ?whom2)))

 =>

 (assert (CAPABILITIES:: Matlab-simulations (running-time-sec
(/ 1 (+ (/ 1 ?t1) (/ 1 ?t2))))) (thanks-to (str-cat ?whom1 “and” ?whom2))))
)
```

Code 12: General structure and examples of emergence rules

This is so because very complex behavior can arise from a handful of capability facts and a couple of very simple rule such as the ones shown in Code 8.

The power of rules to simulate emergent behavior is not a new discovery, and in fact cellular automata are based on rules. Amongst its proponents, Stephen Wolfram, the creator of *Mathematica*™, wrote a 1500 page book titled “A new kind of science”, in which he states that cellular automata would become the new simulation paradigm for physical and biological systems in the 21st century. Rules are arguably a better way of modeling systems emergence than traditional imperative programming, in part due to the natural recursivity of functional and declarative programming languages in general.

However, this powerful recursivity also brings forth several disadvantages. First, computational complexity issues may appear if inadequate rules are created. Computational complexity for RBES is on the order of RF^P where R is the number of rules, F is the number of facts in working memory, and P is the average number of premises per fact. Although the Rete algorithm is very efficient in this pattern matching process, for very large systems computational complexity may be an issue. This risk can be partially mitigated if limits are set for the maximum level of recursion in order to prevent exponential explosion. For example, one may allow for a certain emergence rule to fire a certain number of times. This kind of approach results in a loss of creativity, but it may be required for some large systems.

Second, it may be harder to predict system behavior when a rule is added or modified in large systems. If one is not careful enough during development, it is not hard to create a rule that will result in an endless loop. The explanation facility will play a key role during the tool development process as well as during the production phase.

Third, it may be harder to program and debug just because most programmers are used to imperative languages in which flow of execution is fully controlled by the developer. The use of debugging interfaces such as the ones provided in open-source Eclipse can be of some help.

3.7 Fuzzy Attribute rules

Description: The need for fuzzy attribute rules comes primarily from the requirement to be able to use inexact reasoning in the creation of requirement satisfaction and capability rules and facts. More often than not, experts and stakeholders use imprecise attributes when describing the capabilities of a component, or the requirements for a system. For example, an L-band radiometer can measure soil moisture with "high accuracy" and "30 to 50 km spatial resolution". A particular community of hydrologists may require for instance a soil moisture data product with a revisit time "on the order of 2 or 3 days", and they may state that they would lose "some utility" if the revisit time falls to "3-7 days", "most of the utility" if it falls to 7-14 days, and "virtually all utility" if it falls beyond two weeks. The precise numerical values of these variables are not known early in the system architecting process because of a variety of sources of uncertainty and ambiguity. Some of them will resolve as the system development process goes forward, while some may stay unresolved well into system operation (e.g. uncertainty related to the scientific value of a data product).

Hence, precise numerical values (e.g. radiometric accuracy of 5%) and imprecise fuzzy values (e.g. "high radiometric accuracy") have to coexist in the framework, so that requirement and capability rules can be created using both precise and imprecise statements.

Fuzzy attribute rules are the dictionary that translates back and forth from the precise world to the imprecise world. Note that the names of numerical attributes are required to end with the character '#', while the names of string and fuzzy attributes can end in any character except for '#'.

The general structure and an example of a fuzzy attribute rule are given in Code 13.

```
General structure (in natural language):
(Define-rule numerical-to-fuzzy-attributeX
"Description of the rule: "Transforms a numerical value into a fuzzy value"

IF there is a Capability (of type T1) (with numerical performance attribute A1#
x1) (and fuzzy performance attribute A1 unknown)

=>(THEN)
COMPUTE ?val = (transform-numerical-to-fuzzy attribute A1 value x1)
MODIFY Capability (fuzzy performance attribute A1 ?val)
)


Example (in the CLIPS language):

(defrule FUZZY::numerical-to-fuzzy-running-time
   "Transforms a numerical running time value into a fuzzy value"
?m <- (CAPABILITIES::Matlab-simulation (running-time nil)
 (running-time# ?t&~nil))
```

```
  =>
 (if (< ?t 1) then (bind ?val Less-than-1s) elif (< ?t 10) then (bind ?val
Between-1s-and-10s) else (bind ?val More-than-10s))
 (modify ?m (running-time (numerical-to-fuzzy ?val))))
```

Code 13: General structure and example of a fuzzy attribute rule

Note that in addition to fuzzy attribute rules, it is also needed to define a number of fuzzy operations in order to operate with fuzzy numbers. General algebraic operations such as "+", "-", ">", "<", and "==" are augmented in order to be able to handle fuzzy numbers. These fuzzy operations were implemented in CLIPS/Jess from interval arithmetic. Hence for example:

$[a, b] + [c, d] = [\min(a + c, a + d, b + c, b + d), \max(a + c, a + d, b + c, b + d)] = [a + c, b + d]$

$[a, b] - [c, d] = [\min(a - c, a - d, b - c, b - d), \max(a - c, a - d, b - c, b - d)] = [a - d, b - c]$

$[a, b] \times [c, d] = [\min(a \times c, a \times d, b \times c, b \times d), \max(a \times c, a \times d, b \times c, b \times d)]$

$[a, b] \div [c, d] = [\min(a \div c, a \div d, b \div c, b \div d), \max(a \div c, a \div d, b \div c, b \div d)]$ when 0 is not in $[c, d]$.

Discussion: In the beginning of this subsection, we motivated the use of fuzzy attribute rules by a requirement to handle inexact reasoning. This is thus a matter of ambiguity in language, rather than uncertainty. Indeed, while inexact reasoning is purely deterministic and deals with intervals of numbers and membership functions, the classical interpretation of uncertainty is necessarily probabilistic. However, it is apparent that ambiguity and uncertainty in requirements and capabilities may have common sources, and that they can be treated in similar ways. In particular, scientists and engineers often use fuzzy terms when expressing requirements and capabilities precisely because there is uncertainty in their assessments. Therefore the use of fuzzy numbers allows for the incorporation of some of the uncertainty into requirements and capabilities in a deterministic way. In a sense, this is a conservative approach because it does not try to provide probability density functions for every uncertain variable. The addition of probability density functions and other statistical methods to the framework could be considered as an opportunity for future work.

Another question may concern the choice of simple interval theory over other more involved theories. The framework could very well be augmented with Zadeh's fuzzy sets (L. A. Zadeh, 1965), assuming triangular membership functions and Zadeh's extension principle. Even more recent frameworks could be used, such as Buckley's fuzzy numbers (Buckley, 1985), or Fortin's gradual numbers (Fortin, Dubois, & Fargier, 2008). The latter were recently applied in particular to inversion problems by Boukezzoula with success (Boukezzoula, Foulloy, & Galichet, 2011). All these methods are more powerful than simple interval arithmetic, but they are also more or less cumbersome to implement. Their study is left for future work.

3.8 Explanation rules

The inclusion of explanation rules in the framework responds to the primary requirement of transparency and traceability of results, and was an important factor in the decision to implement the framework using a rule-based language as opposed to more traditional imperative languages.

Description: We have seen earlier that both nominal and degraded requirement satisfaction rules assert explanation facts as they fire. These facts trace the combination of capabilities that satisfied the requirement, and the components and subsystems that generated that capability. Note in particular that a capability satisfying a requirement may not result directly from a capability rule; it may also have been created by a synergistic combination of capabilities after the execution of several capability and emergence rules. Thus, traceability becomes even more important.

Explanation rules detect when these facts are asserted and send them to an explanation facility that processes them. This explanation facility could be arbitrarily complex and include amongst other things the use of graphical user interfaces to show the information to the user.

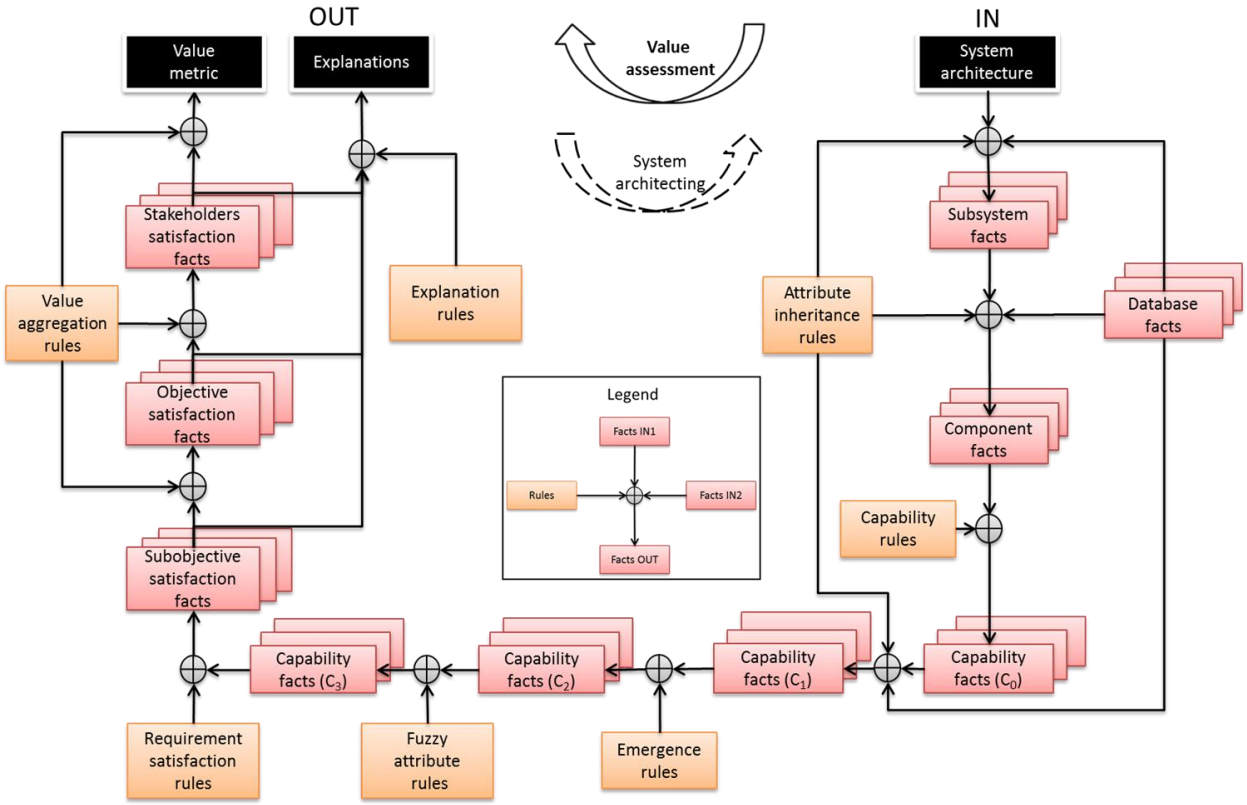
The code for the explanation facility is obviated for now because it is highly domain-specific, and thus will be discussed in Chapter 4.

Discussion: Explanation facilities have been utilized in expert systems since the very beginning with MYCIN (W.J. Clancey et al., 1984). Moreover, research has shown that the use of an explanation facility increases the acceptance by decision makers of results from DSS tools (Ye & Johnson, 1995). The features expected from an explanation facility range include graphical interfaces and explanation queries. Since MYCIN, a variety of techniques have been used in explanation facilities including amongst others automatic programming.

As future work, we propose to develop a simple, elegant, and scalable strategy based on the idea of wrapping every numerical value inside an object that contains, in addition to the numerical value itself, an expression of the uncertainty around that value if required, and a string containing the explanation that led to that particular value.

3.9 Summary

In this section we described the VASSAR framework for assessing the value of a system architecture using rules. Figure 23, which illustrates the overall framework, is repeated here for convenience.



The framework provides both a fuzzy or numerical value assessment and an explanation for this assessment for a certain system architecture. Subsystem and component facts are asserted from the system architecture. An initial set of capability facts is then asserted through the capability rules. Attributes are inherited from subsystems to components to capabilities as expressed in the attribute inheritance rules. An augmented set of capabilities is then created by the application of the emergence rules to the initial set of capabilities.

Requirement satisfaction rules assert subobjective satisfaction facts according to this augmented capability set. Fuzzy attribute rules are simply used as a dictionary between numerical values and fuzzy values. Instead of having one metric for each subobjective, it is more practical to aggregate subobjectives and objectives into a handful of metrics, which is handled by value aggregation rules. Finally, explanation rules process the information in satisfaction facts and show it to the user in different formats.

In addition to these different types of facts and rules, Figure 23 shows the presence of a database that is used in the attribute inheritance process. The use of specialized databases complements such as mission analysis databases or component databases facilitates the integration of knowledge into the RBES.

At different points in the discussion we mentioned the possibility to develop Excel user interfaces to facilitate the input of rules. Indeed, many rules of the same type may be almost identical in structure for many systems, and therefore the automation of this process is possible.

Interesting aspects of the framework and its major limitations were addressed during the individual discussion sections for each type of rules. We discussed the scope and applicability of the framework, which even though may seem restricted at first glance, is maintained general through the appropriate use of capability and emergence rules.

We mentioned in particular the treatment of cost metrics in the framework as capabilities and requirements. In other words, the framework makes no distinctions between benefit and cost other than the specificity of each kind of rules.

4 Rule-based system architecting of Earth Observation Satellite Systems

The rule-based system architecting methodology and the theory behind it have been introduced in Chapters 2 and 3. These chapters have focused on domain-independent knowledge. In this chapter, the methodology is applied to Earth Observation Satellite Systems (EOSS). Thus, the chapter focuses on domain-specific knowledge, as illustrated below.

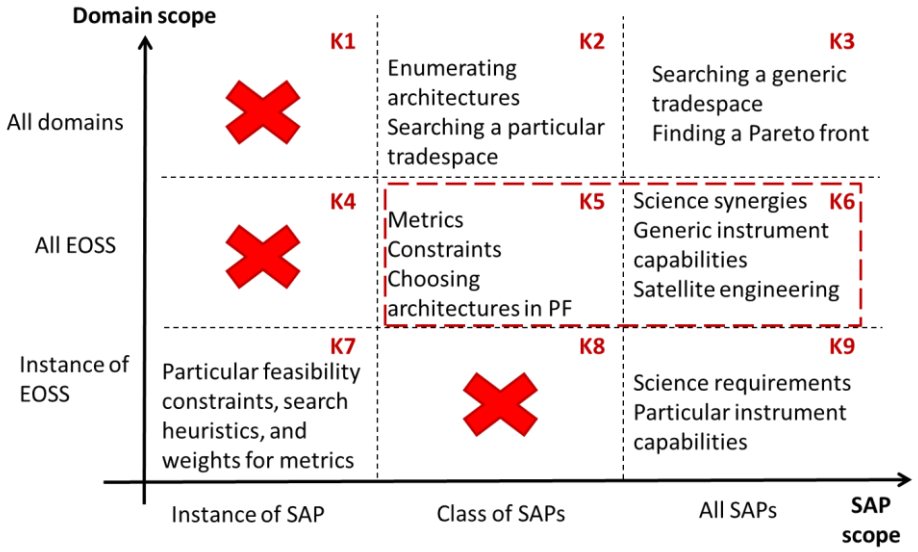


Figure 24: Situation of the content of chapter 4 in the knowledge separation chart

The chapter starts with a discussion about what constitutes the architecture of an EOSS. The main architectural decisions for EOSS are identified and discussed. Second, domain-specific rules that are only relevant for the high level design of EOSS, but independent of the EOSS at hand, are introduced. Third, the VASSAR methodology for assessing value, presented earlier in Chapter 3, is applied to EOSS. Fourth, the figures of merit (FOM) used to evaluate EOSS are presented. These FOMs build on the domain-specific rules presented earlier in this chapter. Finally, the methodology used for the three case studies is described, and the three case studies are briefly introduced.

4.1 Architecture of Earth Observation Satellite Systems (EOSS)

4.1.1 Overview of the architectural framework

Basic system architecture theory teaches us that, when trying to identify the architecture of a system, it is appropriate to start by asking ourselves who the main stakeholders of the system are, and how the system provides value to the stakeholders.

At first sight, it seems intuitive to say that the main stakeholders of EOSS are scientists, since the primary objective of Earth science missions is to take valuable datasets that are then fed to algorithms to produce new knowledge and information. However, in reality, there are a variety of stakeholders that are beneficiaries of, or put their assets at stake with, EOSS. Atmospheric temperature and humidity profiles can be assimilated into numerical weather prediction models to increase the performance of weather forecasting. Global maps of the concentration of several pollutants in the atmosphere are also used to produce air quality forecasts that are routinely read by millions of users. High resolution passive optical imagery of the land and oceans is used to support disaster monitoring activities, assess changes in land use, and provide intelligence, surveillance, and reconnaissance to military units amongst others. Sea level height and ice sheet thickness measurements are used as primary markers for melting of ice caps in climate change. SAR images can be used to track storms and even predict earthquakes. In short, EOSS provide value to a broad variety of stakeholders that goes well beyond the realms of science. For a thorough stakeholder analysis of EOSS, the reader is redirected toward Tim Sutherland's thesis (Sutherland, 2009).

Although there is a broad variety of stakeholders and stakeholder needs, we propose that primarily, EOSS provide value to their stakeholders because they take valuable **measurements** (data products) that are then transformed into information and knowledge, which actually bear the value. Therefore, most stakeholder needs and goals can be projected into a set of measurement requirements. Using words from the system architecting jargon, measurements are the fundamental elements of function of EOSS. This is not to say that all exchanges in the value network of an EOSS are exchanges of measurements and data products (and money on the cost side). Other sources of benefit that are not directly traceable to measurement requirements are for example partnerships with international organizations, or sustaining an industrial base. However, measurement requirements undeniably constitute a majority of the requirements in any EOSS, or at least the majority of requirements that vary across system architectures (i.e., that are architecturally distinguishing).

In the form domain, an EOSS can be defined as a collection of Earth observing satellite missions. A mission consists of one or more spacecraft, and each spacecraft carries a number of instruments. Measurements are taken by instruments, which in our model are the fundamental elements of form of EOSS.

An instrument-centric view of an EOSS, as opposed to a mission-centric view, provides the right level of abstraction for architectural studies: going one level deeper in detail would imply delving into instrument subsystem design, which is not needed at this early stages of mission development; on the other hand, staying at the mission level would preclude the study of important architectural trade-offs, such as deciding whether two instruments should share a common bus, or rather fly on dedicated buses (Selva & Crawley, 2010).

A pictorial summary of our architectural model for EOSS is provided in Figure 25.

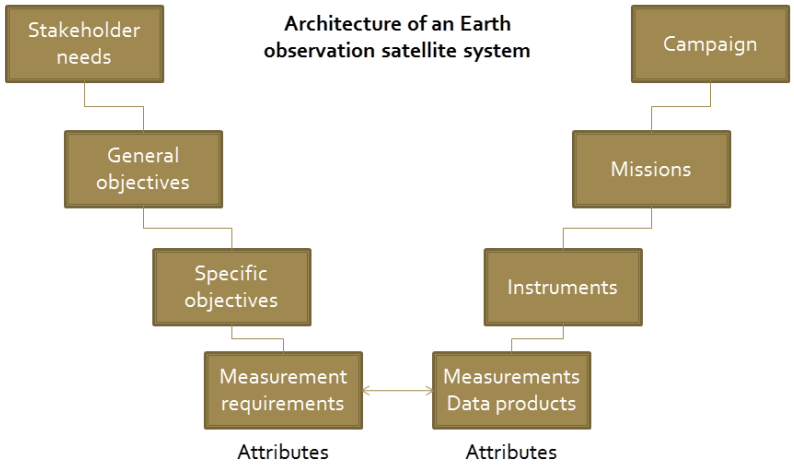


Figure 25: Architectural model for an EOSS.

On the left hand side of Figure 25, stakeholder needs are decomposed into general system objectives, specific system objectives, and finally a set of measurement requirements. On the right hand side, we find another set of measurements that are taken by the instruments flown on a collection of missions that constitutes the EOSS. The fundamental idea behind the methodology is that the value of an EOSS can be assessed by performing a comparison of the measurement requirements and the measurement capabilities at the attribute level (e.g., spatial resolution, temporal resolution). The discussion in Section 4.3 further develops this idea.

4.1.2 Architectural decisions and architectural views for EOSS

The previous section defined what we understand by the **value of an EOSS**. This section goes on to define what we understand by the **architecture of an EOSS**. The major architectural decisions for an EOSS can be inferred from the framework presented in Figure 25. Essentially, the architecture of an EOSS is given by the following decisions:

- **Instrument selection**: which payloads do we choose to perform the required measurements? For example, in order to satisfy topography requirements, we may choose to develop a laser altimeter, a synthetic aperture radar, or both.
- **Instrument packaging**: which mission architectures do we choose to fly these payloads on? More precisely, the major mission-level architectural parameters are the mission mode (e.g., single satellite, tandem, constellation, train), the number of planes and the number of satellites per plane in the case of a constellation, and the orbital parameters (mostly altitude, inclination, and RAAN for SSO).
- **Mission scheduling**: how do individual missions coordinate with each other as a program? Missions cannot be considered in isolation since they are linked through a programmatic environment including common resources such as a common budget.

Hence, in the context of this thesis, an EOSS architecture is fully defined once: a set of instruments has been selected; these instruments have been assigned into spacecraft; the order in which the spacecraft need to be launched has been established. Note that the process of making these decisions requires at least an approximate idea of the main design parameters of each mission, such as the constellation design, and a sense of the size of the spacecraft (mass, dimensions, launch vehicle).

Hence, the problem of architecting an EOSS is effectively decomposed into three sub-problems (instrument selection, packaging, and mission scheduling). In the rest of this section, each of these problems is briefly described. For each sub-problem, we proceed as follows: 1) we state the goal of the sub-problem (i.e., describe the output); b) we discuss qualitatively the main trade-offs of the sub-problem; c) we designate architectural variables; d) we classify the problem as one or more of the classes of SAP described earlier; e) we discuss the relevance of the encoding scheme that the library of classes of SAPs suggests for this particular class; f) we provide one or more examples of an architecture; g) we reason about the size of the tradespace.

The section ends with a discussion about the degree to which these three sub-problems are coupled.

4.1.2.1 Instrument selection

Goal: The goal of the instrument selection problem is to find the best possible set of instruments to be flown in a program given scientific and societal objectives, as well as programmatic constraints (e.g. budget).

Qualitative discussion: When selecting a set of instruments, the major trade-off is obviously the classical benefit-cost trade-off. The more instruments one decides to fly, the greater the benefit, with typically diminishing returns as the number of selected instruments increases. In other words, the highest performance requirements are usually much more expensive to achieve than the basic functionality.

A key point that was mentioned in the description of down-selecting problems is that **the benefit of a certain instrument is not independent of the other instruments in the subset**. Hence, it will be necessary in the instrument selection problem to evaluate each subset of instruments as a whole, rather than as the sum of the benefits of each instrument. The issue of scientific synergies between instruments will be described in greater detail in the next section, as it is a first-order effect for the instrument packaging problem while being a second-order effect for the instrument selection problem.

Furthermore, if we allow selecting multiple copies of the same instrument, it becomes essential to be able to quantify the scientific and societal benefit as a function of number of copies. Indeed, the benefit of getting N copies of an instrument is arguably less than N times the benefit of one instrument.

Architectural decisions: The major architectural decisions for the instrument selection problem are the suite of instruments to be flown in the program.

SAP class: The instrument selection problem is most naturally formulated as a down-selecting problem. A possible alternative formulation is the assigning problem with m decisions and two options for each decision (instrument selected, "yes" or "no").

Encoding scheme: We assume, without loss of generality, that there are *N* candidate instruments for the program, and that a one-to-one mapping exists that assigns a unique identification number *id* to each candidate instrument. The selected encoding scheme for this down-selecting problem is E_1 suggested in 9.1.3, i.e., an array of Boolean variables of length equal to the number of instruments being considered. Variable *i* in this array is set to 1 if instrument *i* is selected for the program. It is set to 0 otherwise.

$$A_{IS(N)} = [b_1 \quad b_2 \quad \dots \quad b_N]$$

where A_{IS} represents an architecture for the instrument selection problem; b_j is the architectural variable that contains the information about whether instrument j is selected ($b_j = 1$) or not ($b_j = 0$). In a valid architecture, $b_j \in \{0,1\} \forall j = 1 \dots N$.

Example of valid architecture: For a program in which we consider 8 candidate instruments, the following architecture chooses to fly 5 instruments, namely instruments 1, 2, 3, 6, and 8. Instruments 4, 5, and 7 are not selected for the program.

$$A_{IS(8)} = [1\ 1\ 1\ 0\ 0\ 1\ 0\ 1]$$

Tradespace size: The length of the architectural vector is exactly equal to the number of instruments being considered (N), and the size of the tradespace grows exponentially with N.

$$\left|\{A_{IS(N)}\}\right| = 2^N$$

Note that $\left|\{A_{IS(N)}\}\right|$ is larger than 10^6 for $N \geq 20$. In other words, if there are 20 or more candidate instruments, there are more than a million possible architectures.

4.1.2.2 Instrument packaging

Goal: The goal of the instrument packaging problem is to find the best possible assignment of instruments into satellites given positive and negative interactions between instruments.

Qualitative discussion: The instrument packaging problem can be viewed as a clustering problem, as the goal is to group the instruments in clusters. Therefore, as in any clustering problem, there are two main classes of forces that drive the instrument packaging architecture: positive interactions between instruments (i.e. attractive forces that tend to create large satellites), and negative interactions between instruments (i.e. repulsive forces that tend to create small satellites). In the case of EOSS, these forces lie on three different domains: the science domain, engineering domain, and the programmatic domain.

In the science domain, there are positive synergies between the instruments, or between the measurements taken by these instruments. More precisely, in the context of the instrument packaging problem, we say that two instruments are synergistic if the value of the data products that can be produced from a spacecraft flying the two instruments together is greater than the value of the data products that can be produced from two dedicated spacecraft. This situation arises when the datasets taken by the two measurements are combined in some way to produce new or enhanced data sets. For example, in an altimetry mission, the accuracy of the altimetry measurement is improved if a microwave radiometer is flown together with the altimeter because the wet tropospheric contribution in the error budget of the altimetry measurement is decreased.

While not all interactions in the science domain are positive, it is generally true that negative interactions in the science domain appear through the engineering domain. More precisely, negative interactions in the science domain arise due to design compromises between competing instrument requirements. The most straightforward example is orbit selection.

Let us consider for example an oceanography mission carrying an altimetry payload (radar altimeter, microwave radiometer, GPS receiver), and an ocean color instrument. Ideally, the altimetry payload wants to fly in a high (1000-1300 km) non sun-synchronous orbit to maximize coverage and avoid tidal aliasing by having a diurnal sampling.

On the other hand, the ocean color instrument would much rather be on a lower (600-800km) SSO in order to get rid of lighting variations through the day and obtain a good trade-off between coverage and spatial resolution. These two orbits are obviously not compatible, and no matter which orbit we choose for the spacecraft, the performance of one of the payloads will be affected. **Competing requirements for orbit selection can be driving factors in a packaging problem.**

Other examples of negative interactions that affect the science output are the limited resources aboard the spacecraft. Resources such as power, data rate, or simply space on the nadir surface of the spacecraft are shared between all the components of the spacecraft, and limited by the state-of-the-art of subsystem technology and other external constraints, such as a mandate to launch on a certain launch vehicle, or to fit on a certain standard bus. Hence, combining two instruments on the same platform may lead to situations in which one or more of the instruments are not used to their full capacity. For instance, in ESA's Envisat spacecraft, the SAR was only used with a duty cycle of about 1%, due to the extremely high data rates of this instrument, and the presence of nine other instruments on the spacecraft.

In the engineering domain, most interactions are negative interferences between instruments. Generally speaking, from the engineering perspective, it is more desirable to have dedicated spacecraft for each instrument, since that softens several requirements on the subsystem, and allows for a design of the bus tailored to each specific spacecraft. Note that cost issues are arbitrarily not considered as engineering issues but rather as programmatic issues.

Finally, in the programmatic domain, including cost issues, we have positive and negative interactions in all the aspects of one of the multiple descriptions of the iron triangle in program management (cost, schedule, and risk).

From the cost perspective, **the cost of a packaging architecture is not proportional to the number of satellites**. In some cases, very distributed architectures may indeed be very costly due to the cost of the different buses and launches, but monolithic architectures with very few satellites may also be very expensive because they may require for example the use of much larger spacecraft or launch vehicles.

The optimum point will in general be somewhere between a completely monolithic architecture and a completely distributed architecture, with the key factors being: a) the packaging efficiency of instruments on the bus; b) the packaging efficiency of spacecraft in the launch vehicle; c) whether the buses are ad-hoc dedicated buses that have to be developed from scratch or rather commercial buses are used; d) the specific buses and launch vehicles that are available to choose from; e) the aforementioned engineering issues, which will obviously also have an impact on the cost of developing individual spacecraft.

From the schedule perspective, distributed architectures are more desirable in order to minimize the risk of late instrument delivery. Indeed, in multi-instrument missions, a single instrument may delay a whole mission that would otherwise be ready to be launched. Other considerations are related to the data continuity issues, which pertain to the scheduling problem. Indeed, one may want to include an instrument on a particular spacecraft that is going to be launched first, just because that instrument covers a particular data gap.

Similarly, one may argue that distributed architectures are also more desirable from the launch risk perspective. Intuitively, and without considering the relative reliability of launch vehicles of different sizes, one may not want to "put all their eggs in the same basket".

It is interesting to do the exercise of computing the probability mass function of the number of instruments successfully put into orbit in different packaging architectures. This can be simply done by considering each launch as a Bernoulli trial, and then using a binomial distribution to compute the probability of having k out of n successful launches. For example, in a completely monolithic architecture with 4 instruments, the probability mass function of the number of instruments successfully put into orbit $N_S$only has two possible values: $Prob(N_S = 0) = 1 - R_{LV}$ and $Prob(N_S = 4) = R_{LV}$. In an architecture with 4 satellites, the probability mass function takes four different values.

It is easy to show that the *average* number of instruments successfully put into orbit is independent of the packaging architecture when all launch vehicles are assumed to have the same reliability, and is equal to the total number of instruments times the reliability of the launch vehicles, assumed the same. However, the shape of the probability mass function will depend on the packaging architecture. More monolithic architectures will have a smaller entropy from the information theory perspective than more distributed architectures. Thus, risk-averse decision makers will generally prefer more distributed architectures, where the risk is more spread. Note however that in reality, at each moment in time there may be a finite number of launches available with different reliabilities, and this would also be an important factor.

In summary, a good packaging architecture maximizes positive synergies while minimizing negative interferences. Interactions occur in the science, engineering, and programmatic domains and the sign of these interactions (positive or negative) is non-trivial. A thorough discussion of packaging EOSS architectures can be found in (Selva & Crawley, 2010).

Figure 26 is a good summary of this discussion, and illustrates the advantages of the two main classes of packaging architectures: larger, multi-instrument missions, versus smaller dedicated missions.

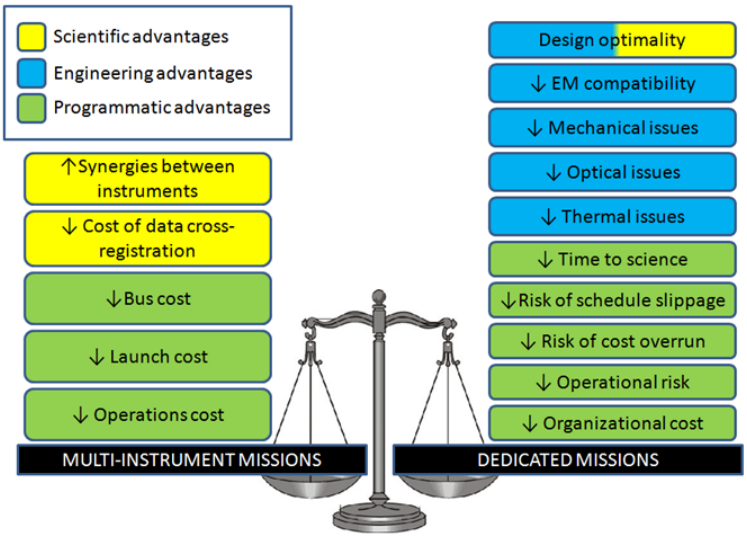


Figure 26: Advantages of the two main components of packaging architectures: multi-instrument missions and dedicated missions

The box labeled as design optimality includes the issues related to design compromises regarding orbit selection, as well as the issues related to the finite resources available in the spacecraft.

Finally, note that the packaging problem is more knowledge-intensive than the selection problem. Obtaining useful results for the packaging problem requires handling a much larger body of domain-specific knowledge that includes satellite sizing, launch vehicle selection, orbit selection, and scientific synergies between instruments and measurements of all the disciplines of the Earth sciences.

Architectural decisions: The major architectural decisions for the instrument packaging problem are the satellites to which we assign each instrument in the set.

Encoding scheme: For the instrument packaging problem, we assume again without loss of generality that there are *N* selected instruments in the program that need to be assigned to a number of satellites $N_{sat} \in [1, N]$. Note that this number *N* in the instrument packaging problem is generally smaller than the number of candidate instruments considered in the instrument selection problem, since several instruments may not have been selected. Furthermore, we assume again a one-to-one mapping exists that assigns a unique identification number id_s to each candidate satellite.

An additional assumption is that each instrument shall fly on exactly one satellite, which precludes two cases: a) an instrument flying on more than one satellite; b) an instrument not flying on any satellite.

This might seem like a very constraining assumption, but in reality it is not, for two reasons: a) the case of the same kind of instrument flying on multiple satellites can be easily modeled by having multiple copies of the same instrument in the input set of instruments; b) the case of instruments not flying on any satellite has already been taken into account in the instrument selection problem.

Under these assumptions, the packaging problem fits the structure of the set partitioning problem described in Section 2.2.3. Therefore, we can retrieve the encoding scheme proposed for partitioning problems, which is an array of non-negative integers of size N, where element *j* of the array is the id_s of the satellite to which instrument *j* is assigned, as suggested in Section 9.1.2.

$$A_{IP(N)} = [i_1 \quad i_2 \quad \dots \quad i_N]$$

where A_{IP} represents an architecture for the instrument packaging problem; i_j is the architectural variable that contains the identification of the satellite to which instrument *j* is assigned. In a valid architecture, the i_j verify that: $i_j \in \{1, N_{sat}\}\ \forall j = 1 \dots N$.

Example: For a program with 5 selected instruments, two examples of architectures are shown below: one consisting of a single satellite carrying the five instruments, and one consisting of 3 satellites carrying 2, 2, and 1 instrument respectively:

$$A_{IP(5),1} = [1\ 1\ 1\ 1\ 1]$$

$$A_{IP(5),2} = [1\ 1\ 2\ 2\ 3]$$

Note in particular that in the second case, instruments with *id* 1 and 2 are on the same satellite (satellite 1), instruments 3 and 4 share another platform (satellite 2), and instrument 5 flies alone.

Tradespace size: The length of the grammar is again exactly equal to the number of instruments being considered (*N*), and the size of the tradespace grows worse than exponentially with *N*, as predicted by the Bell Numbers:

$$\left|\{A_{IP(N)}\}\right| = B_n$$

$$B_n = 1, n = \{0,1\};$$

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k\,, \forall n > 1$$

where B_n is the Bell number of n, as defined in Section 2.2.3.

4.1.2.3 Mission scheduling

Goal: The goal of the mission scheduling problem is to find the optimal sequence of missions given a certain budget.

Qualitative discussion: There are six key issues in the mission scheduling problem, all related of course to time, which is the key parameter in any scheduling problem:

- **Cost/budget**: the spending profile given by the mission costs needs to be consistent with the yearly budget. Given a certain mission cost, mission development time strongly depends on yearly budget.
- **Technology readiness**: in addition to investing on the mission itself (e.g., bus development, integration and testing, launch, operations), it is necessary to invest in the technologies used by the instruments in the program. In particular, a mission cannot be launched until all the technologies have reached the required maturity level. This needs to be taken into account by the scheduling algorithm.

- **Data continuity**: a key constraint driving the launch date of many missions in real life is data continuity. More than fifty years of Earth observation satellites have provided long uninterrupted series of measurements for some parameters of interest such as sea level height, ocean color, vegetation indexes, Earth's radiation budget, or aerosol optical depth. The continuity of such data records is essential in order for them to be a useful input to climate models. A multi-year gap in one of these series can for example have important consequences in the utility of the data records for policy advising.
- **Calibration**: another important constraint for launch dates is inter-mission calibration in order to ensure the consistency of two data sets. These constraints usually take the form of a mandatory period of overlap between two missions. For example, the altimetry mission Jason-2, a.k.a. OSTM, was required to be launched before the end of life of Jason-1, so that the two altimetry measurements could be inter-calibrated in order to eliminate relative biases.
- **Fairness**: Colson identified in his thesis a fifth important element in the mission scheduling problem that he coined "fairness" (Colson, 2008). Fairness in this context means equality across panels or stakeholder groups. A fair campaign of missions is one in which the sequence of missions provides value to the different stakeholder groups in such a way that deviations between the different curves of panel satisfaction over time are minimized. For example, a campaign that flies all the climate missions first and leaves all the solid earth missions for the end of the decade is not fair for the solid earth community.
- **Opportunity cost:** Finally, also in Colson's thesis, the idea of opportunity cost is introduced in the metric of discounted value. When investing on a certain mission, there is an implicit decision to delay investment in other missions, which has a cost, or more precisely a lack of benefit, that is usually called opportunity cost in business and project management. This suggests that scientific and societal benefit could be time-discounted, in a similar way to which cash flows are discounted in project management to compute the net present value of a project. The question arises then of how to choose the discount rates. There is suggestive evidence that different stakeholder groups may have different discount rates. Indeed, in some situations, it could be urgent to obtain a certain data set in order to make an important policy decision, whereas for some other disciplines, it may be less important to get the data fast, due to the slow varying nature of the observable (this is the case perhaps of some disciplines in solid Earth). This will be discussed in more detail in Section 4.4.

Architectural decisions: The major architectural decisions for the mission scheduling problem are the sequence in which the missions are to be launched, and the corresponding launch dates.

Encoding scheme: For the mission scheduling problem, we chose the encoding schemes suggested in 9.1.4.1, i.e., an array of m integers between 1 and m, where m is the number of missions:

$$E: O_i \rightarrow [p_1, p_2, \dots, p_m]$$

$$p_j \in [1, m]$$

Two different encoding schemes can be defined depending on the meaning of the p_j:

- E_1: $p_j = k$ means that element $e_j$is assigned to position k in O_i
- E_2: $p_j = k$ means that element $e_k$is assigned to position j in O_i

Both encoding schemes are used in this thesis, since some rules are easier to implement under E_1, while others are easier to implement under E_2.

Example: For a program with 5 missions, we show below two examples of architectures expressed in E_1 and E_2:

$$A_{1(E_1)} = [1\ 2\ 3\ 4\ 5];\ A_{1(E_2)} = [1\ 2\ 3\ 4\ 5]$$

$$A_{2(E_1)} = [3\ 5\ 2\ 1\ 4];\ A_{2(E_2)} = [4\ 3\ 1\ 5\ 2]$$

The first architecture starts with mission 1 and flies the missions according to the sequence of their ids: mission 2 flies in second position, mission 3 in third position, and so forth. The second architecture starts with mission 3 and ends with mission 4. Mission 1 flies in 4th position.

Tradespace size: The size of the tradespace is given by the factorial of the number of missions:

$$\left|\{A_{MS(m)}\}\right| = |O_i| = m!$$

4.1.2.4 Coupling between architectural views

So far we have described the three architectural views or SAPs as a sequence of completely uncoupled problems that starts with instrument selection, continues with instrument packaging, and ends with mission scheduling. This sequence makes sense because it is absolutely essential to know the instruments selected in order to think about the packaging architecture, and it is absolutely essential to know what the missions are before starting to schedule them.

In reality though, the three sub-problems are coupled: one cannot, for example, make an optimal decision in the instrument selection problem without thinking about the packaging of the selected instruments. This could lead for example to inefficiencies in the packaging factor of a certain bus or launch vehicle. Similarly, it is hard to make an optimal packaging decision without thinking about the scheduling of the missions, since one may want to fly an instrument that closes an important data gap in the spacecraft that is to be launched first. This is illustrated in the N^2 diagram below.

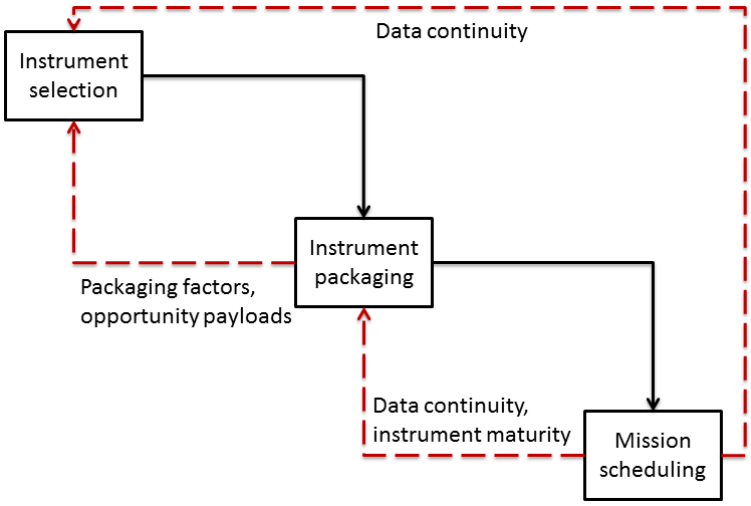


Figure 27: N^2 diagram showing coupling between selection, packaging, and scheduling problems

As a consequence of this coupling, simply solving the individual sub-problems and combining the results will in general lead to suboptimal solutions, just as designing the aerodynamics of an airplane without coupling it to the structures dynamics leads to suboptimal designs. This problem can be treated in three different ways:

- **Solve the global problem without decomposition**. This would imply having a single SAP with an architectural vector containing the three types of decisions: instrument selection, instrument packaging, and mission scheduling. This approach would ideally lead to the true global Pareto front. However, in reality, this problem is intractable for very small number of instruments because the size of its corresponding tradespace is to the first approximation on the order of (although smaller than) the product of the size of the three individual tradespaces, since for every instrument selection, all instrument packaging architectures need to be considered, and for every packaging architecture, every scheduling needs to be considered.

- **Iterate**. The classical approach to solve complex system of coupled equations is iteration. Similarly, when optimization sub-problems are coupled, a simple and usually effective approach is to solve them individually and then iterate until a set of termination criteria are met. The problem with this approach is that there is no guarantee of convergence in general, and termination may occur before convergence (e.g., on a criterion of maximum simulation time exceeded). Furthermore, iteration can also be extremely time-, and resource-consuming.
- **Model the coupling using more rules**. A third strategy to take into account the coupling between the problems is to incorporate knowledge about an SAP in the other SAPs in the form of search heuristic rules, down-selection rules, and metrics. For example, in the instrument selection problem, a rule can be used to improve a selection architecture by adding an instrument that covers an important data gap. These rules will guide the search process to favor architectures that are likely to be good when the other SAPs are solved. This strategy can be combined with iteration in order to reduce the number of iterations needed for convergence.

In the context of the case studies, the first strategy is infeasible due to the computational reasons, and the chosen strategy will be a hybrid of the second and the third, where iteration will be used, and rules capturing the couplings between the problems will be incorporated with the goal of reducing the number of iterations needed.

4.2 Domain-specific rules for architecting EOSS

4.2.1 Overview

The three architectural views and their corresponding SAPs were introduced in the previous section. SAP-class specific knowledge is available for the three problems through the use of the library of classes of system architecting problems presented in chapter 2. This section is concerned with the domain-specific knowledge as opposed to the SAP-class specific knowledge. Thus, the rules that are described in this chapter are only applicable to the architecting of EOSS.

Most domain-specific rules for architecting EOSS belong to the categories of search heuristics or of approximate evaluation rules. The generic rules from the VASSAR framework (namely attribute inheritance rules, instrument capability rules, synergy rules, requirement rules, aggregation rules, fuzzy attribute rules, and explanation rules) are described in next Section 4.3. In this section, search heuristics rules that are specific for EOSS selection, packaging, and scheduling problems are presented. After that, the rules used to do preliminary mission analysis (orbit selection, mass, power and volume budgets, and launch vehicle selection) are described.

4.2.2 Approximate evaluation rules for Earth Observing Mission Analysis

In order to assess the value (both benefit and cost) of an EOSS, it is necessary to have the means to conduct a rough mission analysis for a given mission. The inputs to this mission analysis process include at least the instruments characteristics (e.g., mass, power, data rate, dimensions). The outputs of this simplified mission analysis process are: 1) the main mission orbital parameters, namely altitude, eccentricity, inclination, RAAN, and argument of perigee; 2) a preliminary power budget; 3) a preliminary mass budget; 4) an estimate of the dimensions of the satellite; 5) the launch vehicle "class" selected for the mission. Furthermore, from this preliminary sizing process it is also possible, if required, to assign the most appropriate between a set of standard buses.

In the following paragraphs, the rules needed to obtain these outputs from payload characteristics are described in detail.

4.2.2.1 Rules for orbit selection

Orbit selection is a necessary and critical step in early mission feasibility studies, because orbit parameters dictate the time evolution of ground tracks, and illumination conditions, which in turn affect both scientific performance and engineering considerations (and thus ultimately, affordability).

Science is affected for several reasons: a) some applications require a sun-synchronous orbit to ensure similar illumination conditions across observations, while some applications require not sun-synchronous orbits to avoid aliasing of lower frequency events (e.g. tides); b) some applications prefer observations at specific times of the day which maximize sensitivity to events of interest (e.g. the sensitivity of passive radiance measurements to some vegetation parameters peaks early PM).

Engineering is affected because illumination conditions affect the design of both the electrical power subsystem and the thermal control subsystem. The electrical power subsystem is affected because the frequency and duration of eclipses, both of which depend very much on the orbit, will drive the design of the batteries. The thermal control subsystem is affected because, in addition to eclipses, the evolution over time of the angle between the sun and the spacecraft, which is determined by the orbit, will affect the heat budget.

Following these considerations, different instruments may have different preferred orbits. For example, high energy instruments such as lidars or synthetic aperture radars are typically placed in lower orbits in order to minimize power requirements, while passive optical imaging instruments are placed in higher orbits to maximize coverage.

In general, the optimal orbit for a multi-instrument mission will result from a compromise between all these individual considerations, with some consideration to relative importance of instruments, i.e. typically weighted toward primary mission payloads. For example, in an atmospheric chemistry mission with a lidar and a passive optical imager, the preference of the lidar to fly on a lower orbit for power considerations is likely to be prevalent over the preference of the imager to fly higher for coverage considerations.

High-fidelity mission analysis tools will typically include commercially available or custom-made orbit propagators to predict ground tracks over time and perform accurate coverage and illumination calculations. These expensive calculations are made for each architecture, which results in a decrease in search breadth, as explained before.

On the other side of the spectrum, low-fidelity architectural models will typically abstract out orbital parameters and make assumptions such as: "we assume that all missions have SSO at 600km", or simply consider that the optimal orbit has been selected for all instruments.

We now show how a small set of rules can assign an optimal orbit to satellites in a similar way an expert would do it, with reasonable success and in a very efficient way. For this purpose, a set of allowed orbits considered by the model is provided in Table 6. Note that only LEO circular orbits are considered. The eccentricity and argument of the perigee are thus irrelevant. The mean anomaly is only used in the definition of constellations with more than one satellite per plane.

Table 6: Templates for the rule-based orbit selection

Orbit parameter	**Allowed values**
Altitude	Altitude={250km, 400km, 600km, 800km, 1300km}
Inclination	Inclination={SSO, polar, near-polar, near-tropical, equatorial}
Local time of ascending node	LTAN (for SSO) = {Dawn-dusk, AM, Noon, PM}

Coverage and illumination data was computed for each of these orbits and put into a database. For a given instance of SAP, any subset of these orbits can be considered. If orbits that are not in Table 3 were to be added, their coverage and illumination data would need to be computed and introduced in the database.

The orbit selection strategy works as follows. The RBES is initialized, and all possible instruments and orbits are asserted, i.e. added into working memory. A mission fact is asserted carrying a certain number of instruments, without any orbit assigned. A rule is added that from this initial fact, will assert all possible orbit assignments to all allowed orbits, i.e. all possible combinations of altitude, inclination, and LTAN (circular orbits are assumed). This rule is shown in the appendix, Code 77.

Different instruments have different orbit requirements depending on their type and the mission orbit will be a trade-off between the preferred instrument orbits. An instrument flying in an orbit slightly different than its preferred orbit will still work in most cases but its performance can be severely affected. Hence, this is an instrument-centric orbit selection model, and the orbits in which each instrument is flown need to be deduced from the mission orbits. All these assignments of instruments to orbits are initialized to a goodness of zero.

Then, a number of orbit-selection approximate evaluation rules are added into working memory. The structure of these rules is shown in Code 14. These rules will increase or decrease the goodness slot of orbit assignments, so that at the end, the assignment with the highest goodness will be retained as optimal orbit for the mission.

```
(define-rule ORBIT-SELECTION::rule-name
"Justification for this rule"

IF there is an (orbit-assignment (of-instrument ?ins) (in-mission ?name) (to-
orbit ?orb) (goodnesss ?g))
AND the (Instrument (with Name ?ins) (has  Intent ?int) (and Concept ?c) (and
Power ?p))
AND the (Orbit (orb-name ?orb) (altitude ?h) (inclination ?i) (LTAN ?ltan))

AND ;; conditions on intent (e.g. if it is an atmospheric sounder)
    ;; conditions on concept (e.g. if it is a vegetation instrument)
    ;; conditions on power (e.g. if P > 1000W)
    ;; conditions on illum source (e.g. if active instrument)

=> (THEN)

    (modify orbit-assignment (goodness (- ?g X)); where X is some penalty that
comes from expert knowledge
    )
```

Code 14: Pattern for the orbit selection rules

Depending on the nature of the approximate evaluation rule, we will choose to penalize a certain orbit assignment (i.e. with a positive penalty), or to favor it with respect to all other orbits by assigning a negative penalty. For example, a rule may penalize low orbits for high energy instruments due to power budget considerations, while another rule may favor higher orbits for passive imagers in order to optimize coverage.

Note that the numerical values of the penalties will play a key role in the selection process, and therefore it is important that they be carefully selected. In this respect, this kind of tool is convenient for two reasons: a) they allow for rapid what-if analysis; b) RBES naturally have the ability to explain the reasoning behind a decision by simply pointing out which rules were fired.

This methodology would benefit from the application of a formal process leading to a rigorous derivation of these penalties. Several methodologies exist for that purpose, based on pairwise comparisons (e.g. AHP (Saaty, 2008)), or on iterative negotiation process (e.g. Delphi method (Dalkey & Helmer, 1963)). In this context, this step was not necessary: as it will be empirically shown, the system is capable of correctly predicting the orbits of most Earth observing satellites.

The 19 orbit selection rules are described in the Appendix, classified according to the orbital parameter they concern. These rules were applied to several different existing multi-instrument missions, and the RBES was capable of predicting their correct orbit in most cases, as shown in Table 7. In particular, the RBES was capable of predicting 94% of the orbital parameters correctly for the list of satellites shown in Table 7. 83% of the orbits were exactly classified, and for 13% of the orbits, one parameter was not correctly classified.

Table 7: Validation of orbit selection rules with a set of real multi-instrument missions

Mission	Actual orbit class[8]	Best orbit class according to RBES	payload
ACRIMSAT	SSO-600-SSO-AM	LEO-600-polar-NA	radiation budget wants true polar
AQUA	SSO-800-SSO-PM	SSO-800-SSO-PM	OK
AURA	SSO-800-SSO-PM	SSO-800-SSO-PM	OK
ICESAT	LEO-600-polar-NA	LEO-400-polar-NA	lidar wants to fly low
JASON-1	LEO-1300-near-polar-NA	LEO-1300-near-polar-NA	OK
SEAWIFS	SSO-800-SSO-AM	SSO-800-SSO-AM	OK
QUIKSCAT	SSO-800-SSO-DD	SSO-800-SSO-DD	OK
SORCE	LEO-600-equat-NA	LEO-800-equat-NA	passive wants to fly high
TERRA	SSO-800-SSO-AM	SSO-800-SSO-AM	OK
LANDSAT-7	SSO-800-SSO-AM	SSO-800-SSO-AM	OK
SMAP	SSO-600-SSO-DD	SSO-800-SSO-DD	passive imager wants to fly high
ICESAT-2	LEO-400-polar-NA	LEO-400-polar-NA	OK
DESDYNI-LID	SSO-400-SSO-DD	SSO-400-SSO-DD	OK
DESDYNI-SAR	SSO-800-SSO-DD	SSO-800-SSO-DD	OK
ASCENDS	SSO-400-SSO-AM	SSO-400-SSO-AM	OK
ACE	SSO-400-SSO-PM	SSO-400-SSO-DD	to minimize cost
HYSPIRI	SSO-800-SSO-AM	SSO-800-SSO-AM	OK
GRACE	LEO-500-polar-NA	LEO-275-polar-NA	max sensitivity to gravity
GPSRO	LEO-800-polar-NA	LEO-800-polar-NA	OK
LIST	SSO-400-SSO-DD	SSO-400-SSO-DD	OK

[8] The true orbit class is the orbit from the set of available orbits that is closest to the true orbit

SCLP	LEO-800-polar-NA	LEO-800-polar-NA	OK
XOVWM	SSO-800-SSO-DD	SSO-800-SSO-DD	OK
3DWINDS	SSO-400-SSO-DD	SSO-400-SSO-DD	OK
GACM	SSO-800-SSO-AM	SSO-800-SSO-AM	OK

In one case out of 24, two parameters (type of orbit and inclination) were misclassified. In addition to these orbit selection rules, the **RBES can also select the optimal orbit, where optimality is defined as maximization of the trade-off between science and cost.**

4.2.2.2 Rules for simplified power budgets

The second output of a mission analysis is a power budget. The rules for computing the power budget are provided in the appendix Code 89. Most of the relationships utilized are taken from (Reeves, 1999). Note that the sizing of the power subsystem depends on orbit selection and on payload power requirements, as well as on mission lifetime. The average fraction of time with sunlight, worst Sun angle, and maximum duration of eclipse, were precomputed for each orbit using AGI's Satellite ToolKit™ and the results were put in the mission analysis database.The assumptions for the power budget, based on (Reeves, 1999), are as follows:

Table 8: Assumptions for power budget calculations

Parameter	**Value**	**Justification**
Energy efficiency from solar arrays to equipment	80%	(Reeves, 1999) page 412
Energy efficiency from solar arrays to equipment through batteries	65%	(Reeves, 1999) page 412
Power output from solar array for perpendicular direction (assumes GaAs technology)	253W/m^2	(Reeves, 1999) page 412
Theoretical solar array efficiency (for GaAs)	77%	(Reeves, 1999) page 412, 414
Solar array performance degradation per year (for GaAs)	2.75%	(Reeves, 1999) page 412
Specific power of solar arrays	25W/kg	(Reeves, 1999) page 412
Efficiency of battery to load	90%	(Reeves, 1999) page 422
Specific energy density for batteries (assumes Ni-H2 technology)	40Whr/kg	(Reeves, 1999) page 420
Payload to spacecraft power requirements ratio	40%	(Reeves, 1999) page 340
Duty cycle x% (power requirements are computed $as\ x \cdot P_{max} + (1-x) \cdot P_{avg}$)	20%	Average over several real instruments
Depth of discharge for batteries for GEO, LEO (DD), LEO (not DD)	80%, 60%, 40%	(Reeves, 1999) page 422

The power budget is computed for each satellite using the assumptions in Table 8. Once the preliminary mass of the electrical power subsystem is computed, the complexity penalties provided in Table 9 are applied for the very high power cases. The final mass of the power subsystem is computed as the product of the initial mass and the penalties from Table 9.

Table 9: Complexity penalties for power budget

Satellite BOL power interval	Mass penalty
$P < 7.5kW$	1.0 (none)
$7.5kW < P < 10kW$	2.0
$10kW < P < 15kW$	3.0
$P > 15kW$	4.0

While these complexity penalties arguably capture real life effects, their goal is to penalize architectures that have too many high energy instruments.

4.2.2.3 Rules for complexity-corrected mass budgets

Mass budgets, i.e. the masses allocated to each spacecraft subsystem, are extremely important in satellite missions because lifecycle cost strongly depends on mass for space systems.

There are two approaches to do mass budgets: the high-fidelity approach is a bottom-up approach where the formal decomposition of each subsystem is known and the mass of each component is simply added (e.g. the mass of the attitude determination and control subsystem is equal to the mass of 4 reaction wheels, plus the mass of three sun sensors, plus the mass of 2 3-axis magnetometers, plus the mass of 4 magnetic torquers, plus the mass of electrical connections, etc. This approach however requires a very detailed knowledge of the system that may not be available at the early stages of the mission that are relevant to system architecting.

The lower-fidelity approach is to use simple linear relationships between subsystem masses and payload mass (Reeves, 1999). This top-down approach is based on the assumption that most of the variance of subsystem mass is due to payload mass, and that that is true for most subsystems (with the exception perhaps of the electrical power subsystem that depends strongly on payload power rather than mass).

This approach is generally adequate for system architecting purposes, but for some applications it may not be enough to produce architecturally distinguishing results. This is the case for example of the problem of allocating instruments into satellites, a.k.a. instrument packaging problem. Since the set of instruments is fixed, and in general two packaging architectures only differ in how instruments are divided in satellites, such a linear approach would lead to a total mass (sum of all spacecraft masses) that would be identical for all architectures. In reality, this is obviously not the case, due essentially to three effects that appear mostly in multi-instrument missions: a) non-linearities in subsystem-to-payload mass dependences; b) penalties due to overdesign of the whole spacecraft due to requirements by a single instrument; c) penalties due to interference between instruments.

An example of non-linearities in subsystem-to-payload mass dependences is that of thresholds due to changes in technology. Let us consider for example the design of the communications subsystem. Let alone the fact that payload mass and data rate are not necessarily well correlated, there exist some payload masses and data rates for which the mass of the communications subsystem will increase considerably, due to a change in technology, from S-band to a more performing X-band. Hence, as payload data rate goes beyond a certain threshold value around a few hundreds of Mbps, the rate at which subsystem mass increases with data rate will increase. This can be modeled as a varying subsystem-to-payload mass ratio, depending on the characteristics of the payload.

Penalties due to requirements by a single instrument appear when the design of a subsystem is driven by the requirements of one instrument, and as a result the subsystem is overdesigned from the point of view of the other instruments due to non-linearities in subsystem-to-payload mass dependences as described in the previous paragraph.

An example can be found in the design of the ADCS subsystem of a multi-instrument mission carrying an instrument with very high pointing requirements, such as a limb sounder. The design of the ADCS for a spacecraft carrying this and other less sensitive instruments will be driven by the high pointing requirements of that specific instrument, but will still have to be designed for a payload mass including all the instruments. This design will probably be more expensive than the design of the ADCS subsystems for several single-instrument satellites, from which only one has high requirements.

A trivial example of interactions between instruments can be found in electromagnetic compatibility issues: if an active and a passive microwave instrument share a platform and that they operate on the same spectral region, it might be necessary to add some structural mass in the form for example of a long boom in order to protect the passive instrument from the emissions of the active instrument.

If those instruments were flown on separate satellites, this extra mass would not appear in either of the two satellites, it is an emergent property. Even if some extra mass in the form of a structural appendix is not needed, it will at least be required to think carefully about the spacecraft configuration, so that the instruments are not too close to each other. Whichever option is taken, the fact that these two instruments are sharing the same platform will have a true penalty in terms of development cost, whether it is in the form of a real mass penalty, or just an addendum to the number of work hours for bus development.

Rules-of-thumb can be effectively used to modify mass budgets with payload-dependent subsystem-to-payload mass ratios, complexity penalties due single instrument requirements, and interference between instruments (Selva & Crawley, 2010). In this example, we considered two penalties due to interference between instruments (electromagnetic compatibility issues and vibration issues), and four subsystems had payload-dependent subsystem-to-payload mass ratios: a) thermal control; b) ADCS; c) structures; d) communications.

The mass budgets are calculated in a 3-step process: first, penalties are calculated from the characteristics of the instruments in the spacecraft; second, the mass of each subsystem is calculated using information from these penalties to decide which subsystem-to-payload mass ratios are appropriate; third, complexity penalties are applied in the appropriate cases. Rules are used in each of these three steps. These rules are briefly described in the following paragraphs, and the code is provided in the Appendix.

Mechanical interactions: Instruments affect each other from the mechanical point of view in many ways:

- Moving parts of spacecraft such as scanning instruments create vibrations that can excite the natural frequencies of the structure. This needs to be taken into account in the dynamic behavior study of the satellite in order to avoid harm to the platform or to the instruments.

  For example, the Aqua's AMSR-E is a 300kg scanning parabolic antenna; one can imagine that such a massive device with a diameter of 1.6m in continuous rotation at 40rpm induces heavy perturbations on its co-passengers. This can be modeled as a mass complexity penalty, as shown in the rule in Code 15. Note that a single scanning instrument activates this penalty for the whole spacecraft in which it is flown, regardless of the other instruments.

```
(define-rule MASS-BUDGET::check-scanning-penalty
" This rule finds out whether there is any instrument in the mission
    with a scanning requirement and updates the scanning-penalty boolean flag"

IF there is a satellite whose (scanning-penalty nil) has not been computed yet
AND There is a scanning instrument in the payload of that satellite
AND There is an instrument with precise pointing requirements in the payload
```

```
=> (THEN)

MODIFY (satellite fact (scanning-penalty 1)))
```

Code 15: Rule computing the scanning complexity penalty in satellite mass budgets

- Mechanisms: mechanisms are very often used in satellites to deploy very large solar arrays or antennae. This way some of the problems pointed out in the section concerning limited resources on the spacecraft can be overcome. An example of a relatively simple mechanism is NASA's Aquarius mission with its complex instrument featuring one scatterometers and 3 microwave radiometers. An animation of the deployment of the instrument in orbit can be found in the NASA's Aquarius website[9]. The deployment consists of four independent movement or steps and each of the steps needs to be successful in order for the satellite to work correctly.

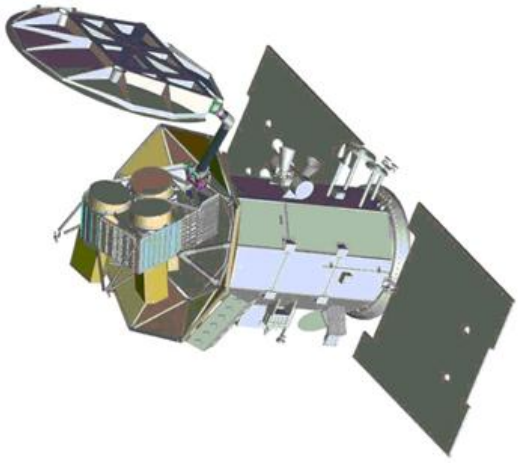


Figure 28: NASA's Aquarius spacecraft featuring one scatterometer and 3 radiometers (Image credit: NASA)

Probably one of the most complex mechanisms never to be deployed is NASA's James Webb Space Telescope. An astonishing animation of the deployment of the James Webb Space telescope can be found in the project's website[10]. It is intuitive to understand that such complex mechanisms involving a large number of steps can substantially lower the reliability of the mission. As a numerical example let us assume that the probability of each individual step being successful is 95%. Then the reliability of the mechanism is 95% ^30 steps = 21%. While the hope is that the value of 95% is highly pessimistic, this calculation illustrates the danger of mechanisms. A rule computing a complexity penalty related to the presence of mechanisms is provided in Code 91 in the appendix. The structure is identical to that of the scanning penalty.

[9] http://www.esr.org/aquarius_sat/AQU_Animation_050624_0811.mpg

[10] http://jwst.gsfc.nasa.gov/videos/09jwsta_depedit_720p_4mbps

- The same moving parts perturb instruments with high pointing accuracy requirements or high integration times. An example of this is the case of IASI, an infrared sounder on the Metop platform. During the testing phase of Metop, the IA&T team noticed that the performance of the instrument was much lower than expected because the instrument was highly sensitive to the micro-vibrations induced by the other instruments on the platform. Eventually it was necessary to add dampers to the instrument to improve its performance. This was obviously not without cost. This kind of problem is modeled through the rule shown in Code 92 in the Appendix.

Thermal interactions: In Earth observation, thermal control is usually integrated in the instrument. Some sensors require to be cooled down to extremely low temperatures in order to achieve functional SNRs. This is typically the case of short wave infrared sensors like ADEOS/GLI, MIPAS and AATSR from Envisat and ASTER, MOPITT, AIRS and HIRDLS from EOS. Temperatures of up to 180K can be achieved by thermoelectric coolers, but to achieve lower temperatures it is necessary to use mechanical coolers such as Stirling coolers, which induce vibrations on the platform. Cryocoolers are usually part of the instruments, so that the spacecraft's thermal subsystem does not have to provide that much cold. However, the design of the thermal subsystem may be more complicated in the presence of cryocoolers because extremely low temperatures can harm other spacecraft components, typically batteries, for which the lowest functional temperature is around -40 deg C. Hence, heaters and radiators may be needed in order to assure that the temperature of the batteries is maintained over -40deg C. A rule capturing this effect is provided in Code 93 in the Appendix.

Even in the case of purely passive thermal control using radiators, another problem of thermal origin appears: all the instruments "fight" to get a good view of cold space which makes the configuration design process much more complicated. For example in the case of Envisat the cold face was the top of the satellite, which explains the accumulation of instruments in that zone. In addition to that, instruments are sources of heat. If two instruments are put next to each other on a platform, there will generally be an interchange of heat with between the instruments and the platform and between the instruments. This needs to be taken into account by the thermal specialist.

Electromagnetic interactions: Virtually all remote sensing sensors emit and/or receive electromagnetic radiation in a certain part of the spectrum. Consequently a number of problems appear:

- Active instruments may jam passive instruments or tracking, telemetry and command equipment working on the same frequency band, and even those that are in different bands because of harmonics and inter-modulation products.

- High RF power instruments can induce currents in nearby electronic devices. This is solved by adequate shielding of all electrical wires.

The consequence of these problems is that the architect needs to think carefully about the configuration of the satellite in order to avoid interferences between instruments. In some cases, it is enough with setting the instruments in opposite sides of the satellite for example, but when this is not possible, structures such as long booms are necessary to protect passive sensitive instruments from the electromagnetic environment in the platform. This is the case of many satellites using precise magnetometers to sense the Earth's magnetic field, such as Swarm, or the GOES 3rd generation. Sometimes booms are also used to isolate TT&C equipment from the rest of the spacecraft like in the case of Landsast-4 or EOS/Terra.

In addition, EMC issues are extremely hard to model and thus to predict. Consequently, extensive EMC testing is required to ensure that instruments will not interfere with each other during flight. Naturally, the number and cost of the tests that are necessary in the case of multi-instrument platforms is much higher than in the case of dedicated satellites.

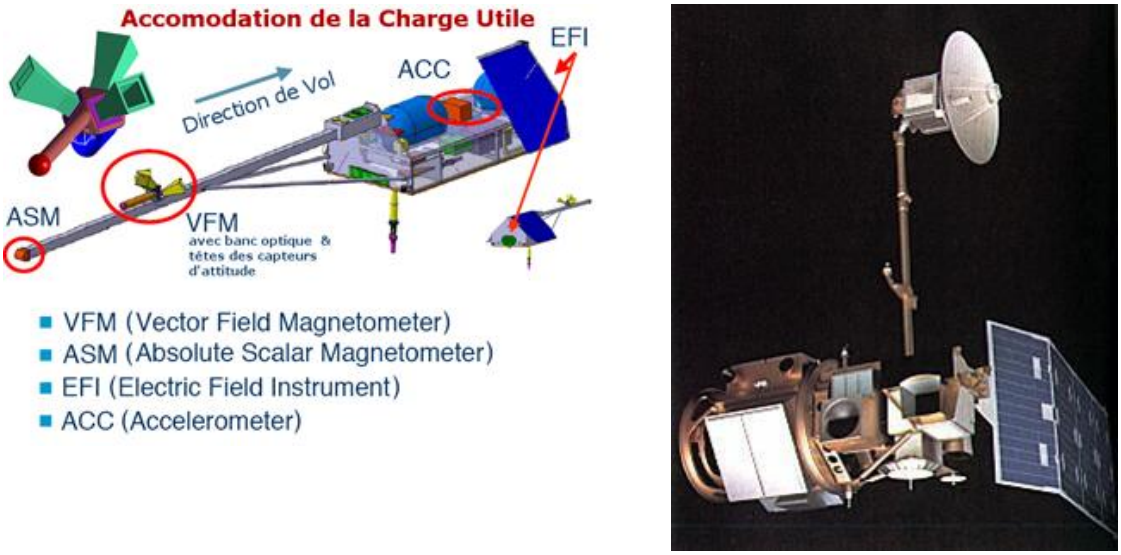


Figure 29: Examples of long booms in the EE/Swarm (left) and Landsat-4 (right) satellites

A rule capturing the EMC interaction between instruments is provided in Code 94 in the appendix.

Optical interactions: The purpose of most instruments on Earth observation satellites is by definition to observe the Earth. This implies that they all require a view of the Earth which means that in most multi-instrument platforms most instruments will be in the nadir looking face. As a consequence, the available surface for instruments on the satellite is far from being the whole satellite surface. Furthermore, instruments have very different viewing concepts and the configuration needs to be so that none of them interfere with each other.

Finally, a last complexity penalty is applied to spacecraft for which the total data rate requirement is large. Indeed, although one can sometimes trade complexity amongst subsystems, there are limitations that prevail due to the state of technology for certain components. Data rate is a good example. State-of-the-art downlink data rates are in X-band (8-12 GHz) which provides around 400-600Mbps in LEO with a reasonable ground station using an antenna on the satellite of reasonable size and emitting a reasonably small power. In order to achieve higher data rates, complex Ka-band or even optical links are being studied. This penalty is captured in the rule shown in Code 95 in the Appendix.

Once these penalties have been computed, rules are used to decide which subsystem-to-payload mass ratios to use depending on these penalties, and mass penalties are applied depending on the penalties. Most of the penalties are applied to the structures subsystem. This is so because it is assumed that all the work related to the spacecraft configuration is assigned to the structures team.

```
(define-rule MASS-BUDGET::design-structure-subsystem
"Computes structure subsystem mass using rules of thumb"

IF there is a Mission fact whose (structure-mass# nil) is not computed and whose
penalties are computed

=> (THEN)

COMPUTE initial structure mass as 0.5462 times the payload mass
ADD 5% mass penalty if mechanisms penalty
ADD 5% mass penalty if EMC penalty
ADD 5% mass penalty if thermal penalty
ADD 5% mass penalty if scanning penalty
MODIFY  Mision fact with (structure-mass# mass with penalties))
```

Code 16: Rule to predict the structure mass in complexity-corrected mass budgets

The payload-to-subsystem-mass fractions are taken from the appendix of (Larson & Wertz, 1999b), and are summarized in Table 10. The averages and standard deviations are over a very small sample of 4 remote sensing satellites provided in this reference.

Table 10: Subsystem-to-payload mass ratios used in the RBES

Subsystem	Subsystem mass/ Payload mass avg	Subsystem mass/ Payload mass Avg-1std	Subsystem mass/ Payload mass Avg+1std
Structure	0.5462	0.4653	0.6276
Power	0.7023	0.5780	0.8266
ADCS	0.1301	0.0809	0.1792
TT&C	0.0983	0.0578	0.1387
Thermal	0.0607	0.0289	0.0925
Propulsion	0.1763	0.0954	0.2572

The numbers in Table 10 are used in the rules as follows:

- For the structure subsystem, the average subsystem to payload mass fraction is taken, and penalties are added on top of this initial mass as shown in Code 16.
- For the power subsystem, the numbers in Table 10 are not used at all as the sizing is done independently using the power budget rules previously presented.
- For the TT&C, ADCS, and thermal subsystems, the high and low values (average plus and minus one standard deviation) are used. The high values are used when the corresponding complexity penalties are active, and the low values are used when they are inactive.
- A complexity factor of 2.0 is applied to the TT&C subsystem for missions for which the data rate is so high that the data produced in one orbit cannot be downloaded to the NASA ground stations.
- A complexity factor of 3.0 is applied to the ADCS subsystem for missions flying at 400km in order to take into account that the drag force is 10 times stronger than at 600km.
- For the propulsion subsystem, the average value is taken, and a propellant mass calculated with a very rough deltaV budget is added. This deltaV budget is simply computed as a function of orbit altitude, estimated satellite dry mass, and lifetime. The rule that computes this deltaV budget is provided in the Appendix, Code 101.

The rules that compute the subsystem masses follow these assumptions. An example for the communications subsystem, which takes into account the data rate penalty, is shown in Code 17.

```
(define-rule design-comm-subsystem
"Computes comm subsystem mass using rules of thumb"
IF there is a Mission fact whose(comm-OBDH-mass# nil) has not been computed, and the
payload mass, and datarate penalty are both available

    => (THEN)

IF (datarate-penalty 1) THEN ASSIGN subsystem-to-payload mass ?comm-mass-coeff 0.1387
```

```
ELSE ASSIGN subsystem-to-payload mass ?comm-mass-coeff 0.0578

COMPUTE comm-mass = payload-mass * ?comm-mass-coeff
MODIFY mission fact (comm-OBDH-mass# ?comm-mass)
)
```

Code 17: Rule to predict the communications subsystem mass in complexity-corrected mass budgets

The rules for the others subsystems are very similar and are provided in the Appendix. The complete complexity-corrected satellite mass budget is thus given by the sum of all the subsystem masses.

The performance of the mass model from the statistical point of view is provided below. Plotting true spacecraft mass versus model spacecraft mass leads to an $R^2 = 0.92$.

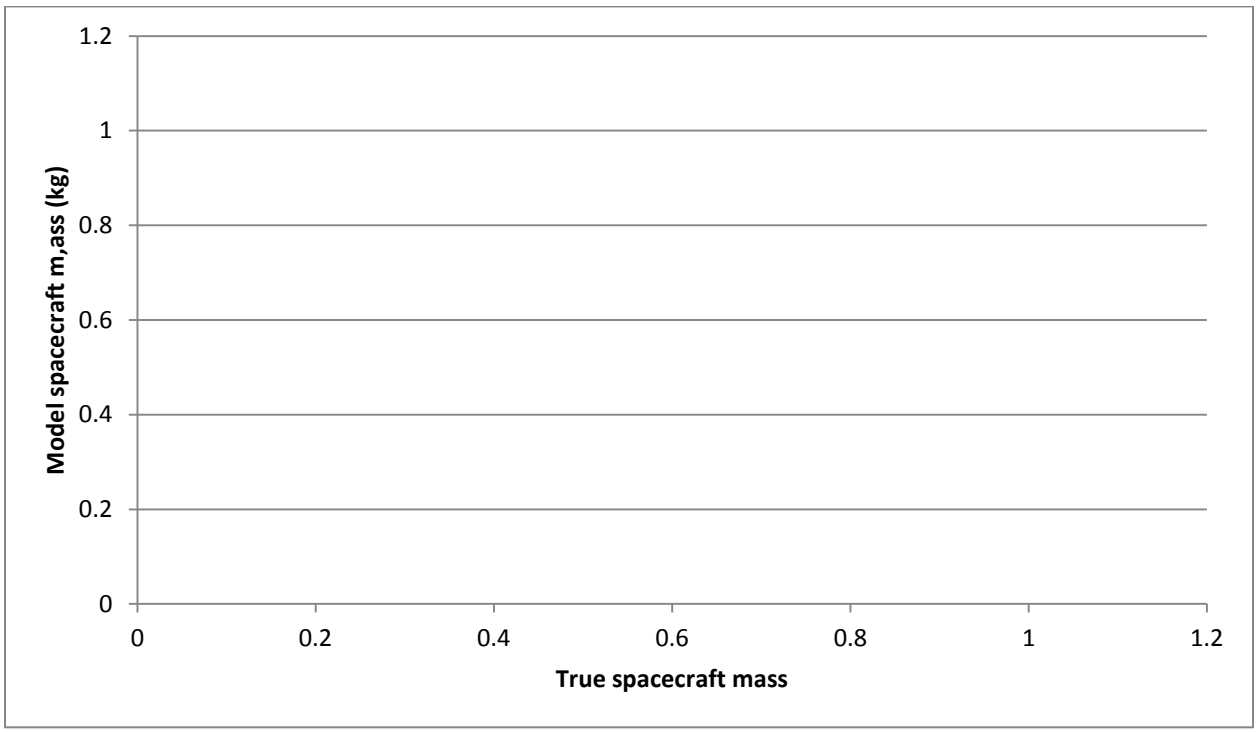


Figure 30: Statistical performance of the mass budget model

4.2.2.4 Rules for standard bus selection

Many Earth observing satellites use commercially available standard buses instead of dedicated buses in order to reduce development and fabrication costs. For example, Thales-Alcatel's Proteus bus has been used for several CNES and ESA missions, including Jason, PICASSO-CENA, SMOS, and MEGHA-TROPIQUES, as well as several NASA missions, including for example CALIPSO (Dechezelles & Huttin, 2001).

The selection of a standard bus is done by choosing the lowest cost compatible standard bus available. Compatibility between the payload and the standard bus includes constraints on payload mass, power, data rate, dimensions, data storage, attitude determination and control accuracy, and viewing geometry amongst others.

Using a standard bus will very likely yield cost savings, if there is a good match between the payload and the standard bus. However, in most cases, **some adaptation** of the standard bus to the payload is required. Depending on the magnitude of this adaptation, the cost savings will be more or less important.

The RBES has incorporated an option to choose a standard bus from a database based on payload requirements. The current database only contains three buses: the **EOS T-330** (Frazier & Engineer, 1998), the **BCP-2000** (A. A. Barnes, 1998), and the **Pegastar** (Lindberg, Lyon, Meurer, & Coxon, 1992). For each standard bus, the rules encoding the maximum payload mass, power, and data rate supported are provided in the Appendix 9.2.5. These values are taken from (Matossian, 1995).

The database as it is today only allows classification of a payload into three classes of standard buses. This may lead to results that are too sensitive to the costs of these buses, for which there is much uncertainty. Therefore, in order for this database to be truly useful, it would need to be augmented with several other standard buses that ensure a smoother transition between payload requirements. One natural example for the database is to use the RSDO-3 catalog X.

4.2.2.5 Rules for launch vehicle selection

An important fraction of mission lifecycle cost for most Earth observing missions is taken by launch cost. Moreover, differences in packaging architectures are sometimes driven by differences in launch cost. Therefore, obtaining a reliable estimate of the launch vehicle of a spacecraft is important.

Table 11 presents the characteristics of a few commercially available launch vehicles. This set of launch vehicles is not complete, but it provides a continuous enough progression of performance and cost, so that the sensitivity to launch costs is reasonable. A much smaller set of launch vehicles would result in a situation where adding a few more kg of payload could make a satellite switch between a small and a large launch vehicle, with differences in cost greater than any other effect, which is undesirable. The data from Table 11 is taken from user guides of the corresponding launch vehicles. Some cost data was also adapted from (Weigel & Hastings, 2004a).

Table 11: Cost, performance, and dimensions of a few launch vehicles

	LEO_POL_400	SSO_400	SSO_600	SSO_800	GTO	diameter(m)	length(m)	cost ($M)
Atlas-5	20000	20000	15000	10000	10000	4.57	7.63	110
Delta-7920	3642	3600	3400	3200	500	2.69	7.53	65
Delta-7420	2269	2257	2123	1989	300	2.69	7.16	55
Delta-7320	1982	1860	1740	1620	250	2.51	6.82	45
Minotaur-IV	1225	1180	1110	1050	0	2	5.44	35
Taurus XL	1015	1053	961.5	870	0	1.98	5.71	30
Taurus	1015	1053	961.5	870	0	1.4	2.67	20
Pegasus XL	300	280	240	190	0	1.18	2.13	15

Given the data in Table 11, and the mass and dimensions of the spacecraft, and the desired orbit, the RBES computes which launch vehicles are compatible with the mission, and assign the less costly option. The following assumptions are used to check whether a launch vehicle is compatible with a spacecraft:

- A performance margin of 10% over spacecraft wet mass is required
- 80% of the maximum dimension of the spacecraft needs to fit in the fairing height.
    - This value of 80% represents the ability of the spacecraft to be folded in launch configuration.
- The nadir area of the spacecraft needs to fit in the launch vehicle are computed as fairing height times diameter
- The maximum dimension of the spacecraft is driven by the maximum dimension of the instrument
- The nadir area of the spacecraft is computed as the sum of the instrument areas

The code for the launch vehicle selection rules can be found in the Appendix 9.2.4.

4.3 Application of the VASSAR methodology to EOSS

The most important metric utilized to evaluate any EOSS is its scientific and societal benefit, and in fact many performance requirements should be traceable to an increase in societal or scientific benefit. The VASSAR methodology to assess architectural value for system architectures using rules was presented in detail in Chapter 3. In this section, the methodology is applied to EOSS.

The section starts with a succinct literature review about alternative ways of assessing scientific value is provided for reference. Second, an overview of the VASSAR framework as applied to EOSS is provided.

Third, the implementation of the different groups of rules used in VASSAR is described. Finally, the methodology is summarized and discussed.

4.3.1 A language for architecting EOSS

In RBES, rules match patterns on facts in working memory. Therefore, rules and facts need to be specified in a common "language", namely a set of templates or data structures with slots or attributes.

In order to be able to express requirements for all the disciplines of the Earth Sciences, it is necessary to define a set of templates that has enough modeling breadth and depth. The goal is to maximize the expressivity of the language by adding as many attributes as needed, keeping in mind that the number of slots in the templates do affect the amount of memory required by the algorithm to work.

The major fact types and corresponding templates used in the RBES are: measurements, instruments (considered instruments and manifested instruments), missions, orbits (for a mission analysis database), subobjective satisfaction and objective satisfaction. A few slots of the templates for missions, instruments, and measurements, are provided in Table 12. Overall, 86 different slots or attributes are defined in the measurement template, 116 attributes are defined in the instrument template, and 52 attributes are defined in the mission template. The complete templates for missions, instruments, and measurements, can be found on http://web.mit.edu/dselva/www/RBES/EOS/. These numbers and Table 12 give an idea of the richness and complexity of the rules that can be expressed with such abstractions. Some of these attributes are only relevant to a particular scientific discipline (e.g. sensitivity-in-low-troposphere-and-PBL for atmospheric chemistry), or to a particular technology (e.g. number-of-looks# for radar). Sometimes, as a result of an expert interview, a new slot was added to a template. This process can be done in a matter of seconds without affecting the rest of the code.

Table 12: Templates for the 3 main types of facts used in the RBES: missions, instruments, and measurements

Template	Examples of slots
Mission	launch-vehicle, lifetime, mechanisms-penalty, mission-architecture, mission-cost#, Name, num-of-planes#, num-of-sats-per-plane#, operations-cost#, orbit-altitude#, orbit-anomaly#, orbit-inclination.
Instrument	Concept, Illumination source, num-of-SWIR-channels, num-of-TIR-channels, num-of-UV-channels, num-of-VNIR-channels, Penetration, Pointing-capability, Radiometric-accuracy#, scanning, sensitivity-in-cirrus, sensitivity-in-low-troposphere-PBL
Measurement Data Product	avg-revisit-time-cold-regions#, avg-revisit-time-global#, avg-revisit-time-US#, Horizontal-Spatial-Resolution-Along-track#, Horizontal-Spatial-Resolution-Cross-track, signal-to-noise-ratio#, Spectral-resolution, Vertical-Spatial-Resolution#, Radiometric-accuracy#, sensitivity-in-cirrus, sensitivity-in-low-troposphere-PBL, NEP-NEDT

4.3.2 Attribute Inheritance Rules

Mission, instrument, and measurement attributes, are populated at different moments in the evaluation processes, and have different origins: the user, other evaluation models (rule-based or not), an optimization algorithm, a database (e.g. a mission coverage attributes are retrieved from a mission analysis database), a parent system element (e.g. an instrument attribute may be inherited from its mission), or a combination of attributes from parent system elements (e.g. a measurement attribute may be computed from a mission attribute and an instrument attribute).

Attribute inheritance rules describe all these relationships between system attributes. A closer look at Table 12 will show that instruments and measurements share a few attributes such as radiometric accuracy. An attribute inheritance rule exists for each of these cases that dictates that the measurement attributes be inherited from the parent instrument. These rules are trivial and are automatically created from a spreadsheet containing the attributes of each template. More sophisticated attribute inheritance rule combine information from two different hierarchical levels in the architecture to populate a low-level attribute.

The structure and a generic example for attribute inheritance rules were provided in the previous chapter in Code 11. An example of non-trivial attribute inheritance rule as applied to EOSS is illustrated in Code 18. This rule computes the horizontal spatial resolution of a nadir-looking instrument from orbit characteristics (mission attribute) and angular resolution (instrument attribute).

```
(Define-rule compute-horizontal-spatial-resolution-cross-scanner
"Compute horizontal spatial resolution from instrument angular
resolution and orbit altitude for a nadir-looking, cross-scanning
instrument"

IF there is an Instrument (with Geometry nadir-looking) (with angular-
resolution# ?dtheta) (flying at an orbit-altitude# ?h)
(with Horizontal-Spatial-Resolution-Cross-track# unknown)

=>(THEN)

COMPUTE ?hsr = 2*?h*tan(?dheta/2)
MODIFY the fact concerning the Instrument (with Horizontal-Spatial-
Resolution-Cross-track# ?hsr)
)
```

Code 18: Attribute inheritance rule that computes horizontal spatial resolution from orbit altitude and instrument angular resolution

The rest of non-trivial attribute inheritance rules is provided in the Appendix. Note that some control over the flow of rule execution is required, since mission->instrument inheritance needs to occur before instrument->measurement inheritance, and inheritance from database needs to occur before both of them.

Therefore, attribute inheritance rules are divided in a three groups with different rule priority (salience property in Jess (E. Friedman-Hill, 2003)).

4.3.3 Instrument capability rules

The main relationships between functional elements (measurements) and formal elements (instruments) are encoded in instrument capability rules.

In the actual RBES, in order to increase flexibility and usability, instrument-specific instrument capability rules are automatically generated in a pre-processing step from a spreadsheet containing the user-input measurements - and their attributes that are independent of the mission attributes - that each instrument can take.

The structure of generic capability rules was presented in the previous chapter in Code 10. When a new instrument is asserted, the corresponding measurement facts are asserted.

With an ideal knowledge of the state-of-the-art of every discipline of remote sensing, instrument capability rules could be completely independent of the EOSS at hand. Just by providing the full set of characteristics of the instrument, we should be capable of retrieving its capabilities. However, this would require an extremely large body of knowledge, for a potentially modest return in terms of fidelity and traceability. Thus, the framework is flexible enough to allow by-passing this feature for cases where knowledge is incomplete, or simply the user wants to increase computational speed. An example of instrument-specific capability rule is shown in Code 19 for the AIRS instrument.

```
General structure (in natural language)
(Define-rule capabilitites-of-instrument-X
“Description of the rule: defines capabilities of instrument X”

IF there is a (manifested instrument (of type T) (named X) (with attribute1 x1)
(with attribute2 x2))
AND (Attribute1 x1 is val1 or better))
AND (Attribute2 x2 is val2))

=>(THEN)

ASSERT a (Measurement fact (of parameter X1) (with “high” accuracy)) (taken by X))
ASSERT a (Measurement fact (of parameter X2) (with “medium” accuracy)) (taken by X))
)

Example (in the CLIPS language):

(defrule AIRS-measurements
   "Define measurement capabilities of AIRS instrument "

 (Manifested-instrument (Name AIRS))
```

```
=>
   (assert (Measurement (Parameter "2.5.1 Surface temperature -land-") (Accuracy
High) (flies-in ?miss) (Id AIRS1) (Instrument AIRS)))
   (assert (Measurement (Parameter "1.2.1 Atmospheric temperature fields) (Accuracy
High) (flies-in ?miss) (Id AIRS2) (Instrument AIRS)))
)
```

Code 19: Structure and example of instrument capability rules

The alternative to defining this rule would be to infer its capabilities from its characteristics, but that would probably require hundreds if not thousands of additional rules, just to find that AIRS can indeed perform the measurements given in Code 19[11].

In addition to the rules that assert the measurements corresponding to each instrument, there are other capability rules that affect the science output of instruments based on: a) orbit selection; b) limitations of resources aboard the spacecraft.

Several rules were presented in Section 4.2.2.1 that assign an orbit to a multi-instrument spacecraft based on how each orbit would affect the science output of each instrument, as well as the cost of the spacecraft. However, these rules do not explicitly affect the science and cost metrics. Instead, the satellite sizing rules (power budget rules, mass budget rule, and cost estimation rules) affect the cost metric, and a combination of requirement rules and capability rules affect the science output.

Requirement rules link the science output of an instrument to its orbit by expressing explicit orbit requirements for the measurements taken by the instrument. For example, the requirement rules related to soil moisture objectives can require that the measurements be taken from a dawn-dusk SSO. Furthermore, degraded requirement rules can explicitly express the fraction of the value that is lost if the local time is AM/PM instead of dawn-dusk.

Several capability rules also link orbit to science output. They are essentially rules adapted from the set of orbit selection rules, which set the output of a certain instrument to zero in case that instrument is put in an orbit in which it cannot function. Since these rules are very important in packaging problems, they are described below one by one:

[11] The set of measurements for the AIRS instrument presented in Code 19 contains only a sample of the measurements. The complete instrument capabilities for AIRS and all the other instruments can be found on http://web.mit.edu/dselva/www/RBES/EOS/EOS Instrument Capability Rules.xlsx

- **Passive optical instruments cannot take their measurements in dawn-dusk (DD) orbits**: Passive optical instruments require the presence of sunlight to work. Thus, they cannot function correctly in dawn-dusk orbits. If an instrument of this type is flown on a DD orbit, its science output is set to zero.
- **Passive VNIR land imagers have very low effective duty cycles on PM orbits:** Passive imagers in the visible or NIR require the absence of clouds in order to take images of the land. In many regions, especially in the tropics, cloudiness increases in the early afternoon. For this reason, most of these instruments fly on AM orbits. Hence, if such instrument is flown on a PM orbit, its science output is also set to zero.
- **Side-looking imagers suffer from image distortion at low altitudes**: Microwave imagers are typically side-looking in order for example to resolve right-left ambiguities. This off-nadir angle increases coverage, but also results in image distortion in the edge of the image due to $1/cos(\theta)$ factor. Furthermore, given an off-nadir angle, the incidence angle on ground will increase with decreasing altitude. Larger angles result in larger image distortions. Hence, in practice, side-looking microwave imagers do not fly at low altitudes to limit image distortion.
- **Two lidars on the same platform working at the same wavelength may interfere with each other**: If two lidars that work at the same wavelength are put on the same platform, the returns from one lidar may interfere with the other one. In order to avoid this problem, in such situation, the model sets the science output of the two instruments to zero.

In addition to being affected by the orbit, science output of instruments can also be limited by availability of resources (namely power and data rate) aboard the spacecraft. The following rules were added in order to take into account these effects:

- **The effective duty cycle of an instrument may be limited by the power available on the bus**: In large satellites carrying multiple high-energy instruments, it is very typical to decrease the duty cycle of the instruments in order to keep the average power below reasonable limits. This obviously affects the science output of these instruments, which could be ON for longer if the instrument was flown on a dedicated spacecraft. In order to model this situation, the RBES computes a max duty cycle due to power limitations as 10kW divided by the satellite BOL power. If this duty cycle is less than 100%, the science output of all instruments aboard this spacecraft is multiplied by this duty cycle. The value of 10kW was chosen because it is representative of the state-of-the-art in satellite power subsystems.

- **The effective duty cycle of an instrument may be limited by the data downlink capability of the bus and the ground stations**: If a large satellite carries multiple high data rate instruments, it can happen that the data produced by the instruments in one orbit at 100% duty cycle cannot all be downloaded to the ground stations. Thus, it may be required to decrease the duty cycle of these instruments.

  In order to model this situation, the RBES assumes that each satellite can have one 7-minute access to a ground station per orbit, at an effective data rate of 500Mbps, which results in 25.6GB/orbit. It computes a duty cycle due to downlink limitations as 25.6GB/orbit divided by the total GB of data produced by the payload in one orbit, and if the duty cycle is less than 100%, the science output of all instruments aboard this spacecraft is multiplied by this duty cycle. Note that this rule is conservative in the sense that 500Mbps is much higher than what most current NASA satellites have available in their downlinks.
- **Minimum duty cycle**: Two different max duty cycles are computed due to power and downlink limitations. In practice, the most constraining (i.e. the minimum) of these two duty cycles will be used to scale the science output of all instruments in the spacecraft.

The CLIPS implementation of these rules is provided in the Appendix.

4.3.4 Synergy rules

The emergence rules presented in the previous chapter for the generic framework specialize into synergy rules in the case of EOSS. Indeed, instrument capability and attribute inheritance rules are not enough to describe some emergent behavior that appears at the level of measurements and data products.

For example, when solving the inverse problems, i.e. the retrieval of some property of interest from the magnitude measured by the instrument, scientists can sometimes apply data processing algorithms such as assimilation algorithms, or disaggregation schemes, which will produce new data products with properties that none of the initial data products had (N. N. Das, Entekhabi, & Njoku, 2011). Since these emergent data products can satisfy requirements that the initial data products cannot, it is very important to be able to model them. This is the role of the synergy rules.

An example of a synergy rule is provided in Code 20. This particular rule inspired by the SMAP mission will apply a disaggregation scheme that will combine a high resolution, low accuracy soil moisture product with a low resolution, high accuracy product to yield a medium resolution high accuracy product. Similar disaggregation schemes exist that for example, combine frequent low resolution datasets with sparser, higher resolution datasets.

The rest of synergy rules is provided in the Appendix 9.2.6. Note that even though there are only 64 synergy rules, some of them, especially the ones that apply to all parameters, are extremely powerful in generating new measurement facts.

```
(define-rule SMAP-spatial-disaggregation
"A high accuracy coarse spatial resolution measurement can be combined with a lower
accuracy high spatial resolution measurement to produce a high accuracy medium
spatial resolution measurement"

IF there is a (Measurement (of Parameter "2.3.2 soil moisture") (with Horizontal-
Spatial-Resolution# ?hsr1) (and Accuracy# ?a1) (Id ?id1))
AND another (Measurement (of Parameter "2.3.2 soil moisture") (with Horizontal-
Spatial-Resolution# ?hsr2&:(> ?hsr2 ?hsr1)) (and Accuracy# ?a2&:(< ?a2 ?a1)) (Id
?id2))
AND there is a fact indicating that the 2 measurements are (cross-registered
(measurements ?id1 ?id2)))

=> (THEN)

ASSERT another (Measurement (of Parameter "2.3.2 soil moisture") (with Horizontal-
Spatial-Resolution# (given by the geometric-average of ?hsr1 ?hsr2))) (and Accuracy#
(the max of ?a1 and ?a2))))
)
```

Code 20: A synergy rule modeling a generic spatial disaggregation scheme such as the one proposed in (N. N. Das et al., 2011)

For example, a rule exists that allows trading temporal resolution against variable measurement error for any measurement by simply averaging out in time. Synergy rules are effectively the way the RBES framework models systems emergence. Synergy rules are EOSS-independent, i.e. they do not depend on the EOSS being considered, but they may be measurement dependent, i.e. some rules modeling particular data processing algorithms may only apply to a certain measurement (e.g. soil moisture).

4.3.5 Requirement rules

Requirement satisfaction rules express stakeholder needs and goals in the form of measurement requirements. Thus, when the requirement rules are executed, the RBES performs the attribute-level comparison between the measurements performed by the EOSS (the instrument capability, attribute inheritance, and synergy rules have all already been applied), and the measurement requirements. There are two types of requirement satisfaction rules: full satisfaction rules, and partial satisfaction rules.

Full satisfaction rules express the level of data quality and quantity (i.e. measurement attributes) required for full satisfaction of a particular stakeholder objective. Partial satisfaction rules cover many degraded cases in which one or more attributes are not at the required level for full satisfaction, resulting in a partial loss of benefit. The general structure and an example of a requirement rule are provided in Code 21. Note that most requirement satisfaction rules have the same structure: they make reference to a specific measurement parameter, and have a set of requirements of the form attribute-required value. Thus, as it was the case for instrument capability rules, requirement satisfaction rules are imported from a spreadsheet containing the numerical data for each subobjective. Requirement satisfaction rules are obviously EOSS-specific.

```
General structure (in natural language)
(Define-rule full-satisfaction-of-subobjective-X
“Description of the rule: conditions for full satisfaction of subobjective
X”

IF there is a Measurement (of a certain parameter) (taken by a certain
combination of instruments who1) (with a spatial resolution hsr1) (a
temporal resolution tr1) (+ requirements concerning other attributes)
AND (Spatial resolution tr1 is Medium-100m-1km or better))
AND (Temporal resolution hsr1 is Medium-1day-3days or better))

=>(THEN)

ASSERT a fact indicating full satisfaction of subobjective X (taken by who1)
)

Example (in the CLIPS language) for the case of EOSS
(defrule REQUIREMENTS::subobjective-WAE8-3-nominal
   "Conditions for full satisfaction of subjective WAE8-3"

(Measurement (Parameter "2.4.2 vegetation state") (Region-of-interest
?x1&~nil) (Coverage-of-region-of-interest Global) (Horizontal-Spatial-
Resolution ?x4&~nil) (Spectral-sampling ?x3&~nil) (taken-by ?who) (Temporal-
resolution ?x2&~nil))
 (test (ContainsRegion ?x1 Global))
 (test (>= (SameOrBetter Temporal-resolution ?x2 Medium-1day-3days) 0))
 (test (>= (SameOrBetter Spectral-sampling ?x3 Multispectral-10-100-
channels) 0))
 (test (>= (SameOrBetter Horizontal-Spatial-Resolution ?x4 Medium-100m-1km)
0))

   =>

   (assert (REASONING::fully-satisfied (subobjective WAE8-3) (objective
"Vegetation") (parameter "2.4.2 vegetation state") (taken-by ?who))))
```

Code 21: Structure and example of requirement satisfaction rules for EOSS

4.3.6 Value aggregation rules

Value aggregation rules in the general framework were presented in Code 9. The current implementation of the RBES for EOSS allows for four different hierarchical levels of value decomposition: 1) overall EOSS value: a single number representing the aggregated scientific and societal benefit of the EOSS architecture; 2) panel value: a set of numbers representing the scientific and societal benefit of the EOSS architecture to the major stakeholder groups; 3) objective satisfaction: a set of numbers representing the degree of satisfaction of individual stakeholder objectives; 4) subobjective satisfaction: a set of numbers representing the degree of satisfaction of the different subobjectives of which an objective consists.

The comparison between the achieved and required measurement sets is done at the subobjective level, i.e. three levels below the overall EOSS value in the hierarchy.

We propose that this removes some of the subjectivity in the knowledge elicitation process, since experts are less reluctant to make an assessment at this higher level of fidelity, and they will probably do it more accurately.

Value aggregation rules combine individual subobjective requirement satisfaction into objective satisfaction, objective satisfaction into panel satisfaction, and panel satisfaction into EOSS value. Value aggregation rules are also EOSS-specific.

As explained in Section 3.3, value aggregation rules can for example use weighted averages of subobjective satisfaction to infer objective satisfaction, weighted averages of objective satisfaction to infer panel satisfaction, and a weighted average of panel satisfaction as metric for overall EOSS value.

More sophisticated value aggregation rules, in addition to the ones presented in Section 3.3, may include non-linear terms such as Boolean expressions (and, or, not), maximum and minimum values for certain metrics, or non-linear utility functions (e.g. a logarithmic utility curve to account for risk aversion) amongst others.

4.3.7 Fuzzy attribute rules

Traditionally, RBES have been capable of dealing with inexact reasoning and uncertain statements (Giarratano & Riley, n.d.). While our RBES does not have a full fuzzy reasoning capability in the sense of Zadeh's fuzzy sets, some rules were created in order to deal with the simultaneous presence of both quantitative and semi-quantitative information.

For example, a scientist may express a requirement for horizontal spatial resolution using a numerical value (e.g. 250m), an interval (e.g. anything between 100 and 400m), or a fuzzy or ambiguous value (e.g. very high). Therefore, the RBES needs to be able of going back and forth from the quantitative and semi-quantitative world.

In order to do that, fuzzy attribute rules are defined so that a mapping can be specified by the user to make the link between fuzzy attributes and numerical attributes.

An example of fuzzy attribute rule is provided in Code 22. The rest of fuzzy attribute rules are provided in the Appendix. Note that these mappings can be in some cases completely dependent on the application. For example, high spatial resolution can be 100m for hydrology, or 1m for disaster monitoring.

```
 (Define-rule numerical-to-fuzzy-Horizontal-Spatial-Resolution
"Description of the rule: "Transforms numerical horizontal spatial resolution
value into a fuzzy value"

IF there is a Measurement (with fuzzy Horizontal-Spatial-Resolution unknown)
(and numerical Horizontal-Spatial-Resolution# ?num))

=>(THEN)

IF ?num < 1m THEN ASSIGN ?val = "Highest-1m-orless"
ELSE IF ?num < 10m THEN ASSIGN ?val = "Very-high-1-10m"
ELSE IF ?num < 100m THEN ASSIGN ?val = "High-10-100m"
ELSE IF ?num < 1km THEN ASSIGN ?val = "Medium-100m-1km"
ELSE IF ?num < 10km THEN ASSIGN ?val = "Low-1km-10km"
ELSE IF ?num < 100km THEN ASSIGN ?val = "Very-low-10-100km"
ELSE ASSIGN ?val = "Lowest-100km-or-greater"
MODIFY Measurement (with a fuzzy Horizontal-Spatial-Resolution ?val)
)
```

Code 22: General structure and example of a fuzzy attribute rule

4.3.8 Fact databases

In addition to all the rule sets presented in this section, there are two fact databases that are added into working memory at the initialization of the RBES: a) mission analysis database; b) instrument characteristics database. The mission analysis database is used by the attribute inheritance rules in order to compute the temporal resolution of a satellite or constellation. It contains coverage figures of merit, calculated off-line with dedicated simulation software (AGI's Satellite Tool Kit ® a.k.a. STK) for the most common orbits for satellites and constellations in EOSS. This includes sun-synchronous orbits, true polar orbits, and low inclination orbits between 265km and 1000km of altitude, all of them circular. For each orbit, revisit time was computed using STK for several different coverage grids (global, US, tropical regions, cold regions), and then then the average over time, and worst-case on latitude-longitude is reported on the database.

During execution of the attribute inheritance rules in the RBES, the revisit time information corresponding to the relevant satellite or constellation is retrieved. If the information is not available in the database, an STK session can be opened to calculate the necessary data and save it in the database for later execution. The instrument characteristics database is also used by the attribute inheritance rules that copy attribute values from the instrument database to the actual manifested instrument facts. When an instrument is manifested, its characteristics are copied from the database. Note that the instrument characteristics database may be specific to a given case study, or it can also be used as a way to store information about past, present, and future instruments from several agencies and organizations.

4.3.9 Explanation rules

Explanation rules for the VASSAR framework were described in Section 3.8. The explanation facility developed for EOSS includes several high level functions that build on these explanation rules and provide more sophisticated explanations about science and cost metrics. These functions include amongst others a function to compare two instruments or missions. This function evaluates both missions and provides detailed explanations for both science and cost.

In terms of science, it explains: a) which combination of datasets and data processing algorithms enables satisfaction of subobjectives that are partially or fully satisfied; b) the reason or reasons why the mission misses partially satisfied subobjectives, and subobjectives that could potentially be satisfied but are completely missed. Subobjectives that could potentially be satisfied are identified whenever there is a subobjective that is completely missed, and there is a measurement fact whose parameter corresponds to that subobjective.

In terms of cost, the function provides a detailed mass budget showing which engineering penalties are active, a detailed cost budget, the launch vehicle selected for each mission, and the standard bus selected for each mission if appropriate.

Another useful function in the explanation facility provides a comparison of two selection, packaging, or scheduling architectures. This function shows the differences between the two architectures both in the objective space and the space of architectural variables. For packaging architectures, it also shows which synergies are captured and lost in one particular architecture, and which interferences penalize that particular architecture, so that the user may get an idea of how to improve it.

Many functions in the explanation facility are text-based, and work by querying the working memory of the RBES. Other tools are available that support graphical reasoning about the trade space, for example by providing the information about one particular architecture on a plot on demand, or highlighting architectures on the plot that share one particular characteristic in the architectural variable domain.

The explanation facility plays an extremely important role in the framework, as it ensures the traceability of the results to the model assumptions and underlying knowledge base. Furthermore, it improves the efficiency of the tradespace exploration process by providing the means to the human operator of designing better heuristics for the next iteration.

4.3.10 Summary

Figure 31 shows the flow of execution of the RBES. From the EOSS architecture, the corresponding mission and instrument facts are asserted. Instrument capability rules assert the measurement facts associated to each of the manifested instruments. Attribute inheritance rules regulate how attributes are inherited between missions, instruments, and measurements: instrument attributes are inherited either from their mission or from the corresponding instrument in the database; measurement attributes are inherited from their instrument, their mission, or they are derived from combinations of other mission and instrument attributes. The initial set of measurement capabilities is completed with new and modified measurements through the synergy rules. This new set of measurement capabilities is compared against measurement requirements defined in the requirement rules, which assert subobjective satisfaction facts (full or partial satisfaction). Subobjective satisfaction facts are logically combined to produce objective satisfaction facts, and objective satisfaction facts are combined to produce panel satisfaction metrics. These two steps occur through the value aggregation rules. Finally, panel satisfaction metrics are weighted to produce an overall EOSS score.

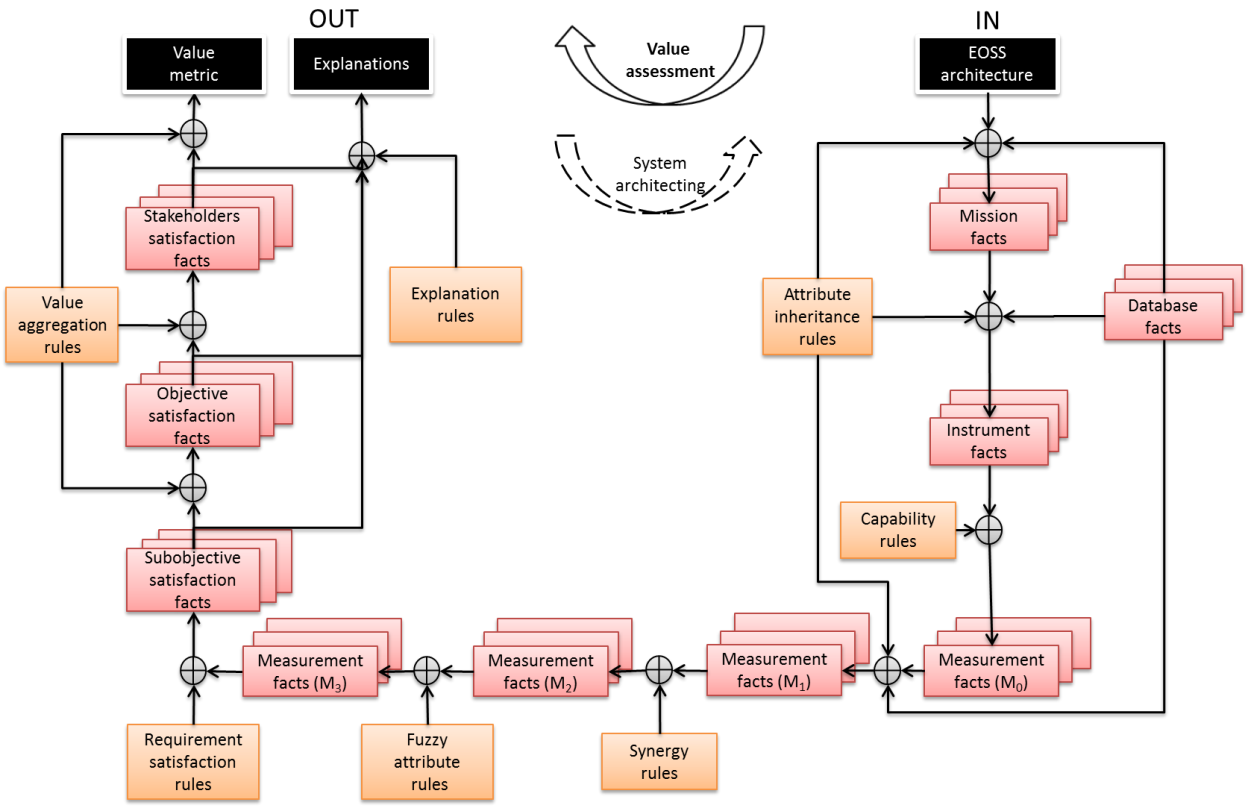


Figure 31: Flow of execution in the Rule-based Expert System for assessing the scientific value of EOSS

4.4 Figures of Merit for EOSS

Before going into the description of the figures of merit developed for this tool, we briefly discuss the two main methodologies used for identifying figures of merit: the first one is the preferred one for system architecting, and it is based on an exhaustive decomposition of all stakeholder needs, whereas the second one is a more classical program management approach, focusing on programmatic success.

The goal of the system architecting process is to create a system architecture that ensures sustained value delivery to stakeholders. Hence, a good approach for identifying figures of merit is to do a stakeholder analysis, and to perform a top-down decomposition of stakeholder needs into objectives. A thorough NASA/NOAA-centered stakeholder analysis was conducted by Timothy Sutherland for the US Earth observation program (Sutherland, 2009), and there is no need to replicate this effort in the context of this thesis. Sutherland created a stakeholder network with 13 major stakeholders, and identified the most important stakeholders and value flows in the network. Some of the qualitative arguments and quantitative parameters of my tool are based on the finding of this thesis.

For example, for the Decadal Survey case study, the relative weights of the six scientific panels (Climate, Weather, Land, Water, Health, and Solid Earth) come from value flow calculations in Sutherland's stakeholder network. This approach leads to a more holistic view of the system.

A more classical approach to identifying figures of merit, is to start off from the "augmented" iron triangle. The "classical" iron triangle in project management graphically illustrates the tension that exists between cost or resources, time or schedule, and performance or scope. It is typically said that from these three, at the very best one can optimize one, and control a second one (as in hold it under an interval), but necessarily let the other free. Other versions of the same principle are less optimistic and observe that you can control two, and let the third free (no optimization possible). Thus, one can achieve the desired performance for a fix schedule, but then cost overruns may occur (e.g. Apollo program); or one can meet cost and schedule goals, but that may require descoping or performance degradation in general (e.g. many "successful" current space programs). Note that this is an upper bound of what can be done, and systems exist that fail to achieve cost, schedule, and performance goals simultaneously. More recently, the "classical" iron triangle has been "augmented" with a fourth aspect, namely risk, to acknowledge that in some cases, the tension between the three elements can be partially resolved by increasing risk. For example, in the NASA Faster-Better-Cheaper approach, cost, schedule, and realistic performance targets were sometimes simultaneously met, but that was at the expense of increased risk.

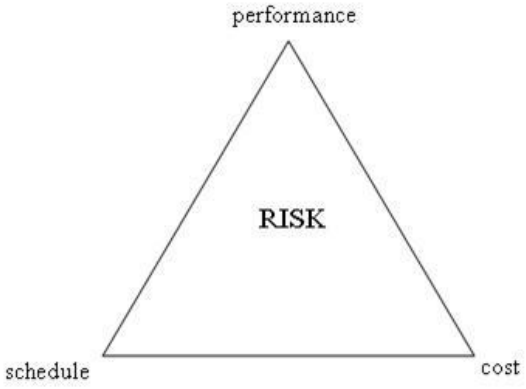


Figure 32: Augmented iron triangle

In this second approach for identifying figures of merit, these four categories (cost, schedule, risk, performance) are decomposed instead of stakeholder needs, leading to a set an alternative set of figures of merit. This alternative set is typically biased towards programmatic stakeholders, because from the four main categories, three deal with programmatic aspects and only one with performance.

The two approaches should yield the same results, as both of them should include in principle cost, schedule, risk, and performance considerations. However, the first approach tends to lead to more detailed decomposition of performance metrics, while the second approach tends to lead to more detailed decomposition of programmatic metrics. Therefore, in the context of this thesis, we used both approaches with the purpose of obtaining a well-balanced set of figures of merit for Earth observing programs.

The metrics used to solve the different SAP about EOSS are described in the rest of this section. These metrics build upon the domain- specific rules presented in Section 4.2, as well as on the rules resulting from the application of the VASSAR framework as described in Section 4.3.

4.4.1 Scientific and societal benefit

The primary objective of an EOSS is to take exhaustive high-quality measurement sets of the Earth's land, oceans, and atmosphere, from space, in such a way that user (scientists, government agencies) requirements in terms of spatial, temporal, and spectral resolution amongst others are met.

As discussed in the previous chapter, there are a few different ways of estimating the scientific and societal value of satellite measurements, which range from very expensive Observing System Simulation Experiments (OSSEs) to simple Science Value Matrices (SVMs). In this thesis, we use a rule-based approach to assess scientific and societal benefit, as described in Section 4.3.

We note that the scientific benefit metric is used both in the instrument selection and instrument packaging problems. However, the range of variation of the metric across architectures is much wider in the instrument selection problem than in the instrument packaging problem, because in the packaging problem the instrument set is fixed.

Variations in scientific benefit in the packaging problem essentially come from: a) synergies between instruments that are captured or lost in the decision of putting instruments together on the same platform or not; b) orbit selection compromises in multi-instrument missions that affect the science output of all instruments; c) finite resources on multi-instrument spacecraft that limit the scientific output of instruments. These three sources of variation of value make the dependence of science value with number of satellites not trivial at all. Fully distributed architectures may lose some synergies between instruments, but some of these synergies can still be captured by flying instruments on train configurations.

Furthermore, in fully distributed architectures, each satellite is tailored to a specific payload and therefore loss of science value due to competing instrument orbit requirements or finite resources aboard the spacecraft completely disappear. On the other hand, an architecture based on fewer larger satellites may be able to capture a few more synergies, but there may also be some loss due to competing instrument orbit requirements and finite resources aboard the spacecraft.

Thus, scientific benefit ranges from 0% to 100% by definition in any instrument selection problem, while in the packaging problem for NASA's EOS (our first case study), scientific benefit ranged from ~60% to 80%. Note that when a packaging architecture is defined, the maximum scientific score may be less than one due to insuperable competing requirements.

Scientific benefit is obviously not directly used in the mission scheduling problem, as all scheduling architectures have the same scientific benefit when we integrate over the entire campaign. Instead, time discounted benefit is used, where the benefit of a mission in the campaign decreases as it is pushed toward the end of the campaign. This metric will be formally defined in Section 4.4.5.

While it is clear that scientific benefit metrics will utilize the VASSAR framework, there are still questions about how to apply the framework. For example, how do we compute the value of an instrument for the selection problem? One possibility is to compute the value of a single mission carrying that instrument. Scores calculated using this approach are called **scores in isolation**, because the instrument is considered in the absence of the other instruments in and outside the program. However, this approach does not capture interactions between instruments. More precisely, let V_i be the individual instrument score of instrument i, and let $V_{\{i,j\}}$ be the score of a mission consisting of the two instruments i and j. Then, it is clear that $V_{\{i,j\}} = V_i + V_j - R_{\{i,j\}} + S_{\{i,j\}} \neq V_i + V_j$. This concept is illustrated in Figure 33.

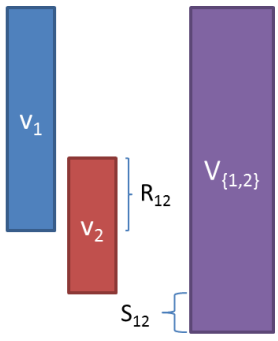


Figure 33: Pictorial representation of the concept of synergies and redundancies between instruments

The blue and red rectangles on Figure 33 represent the notional sets of subobjectives satisfied by two generic instruments 1 and 2. Their relative positions in the vertical axis with some overlap indicate that part of the measurements taken by 1 are also taken by 2, which leads to the definition of the mutual redundancy between instruments 1 and 2 (R_{12}). The purple rectangle represents the capabilities of the combination of the two instruments. This rectangle is positioned in the vertical axis in such a way that it illustrates how synergies between instruments 1 and 2 are capable of satisfying more subobjectives than their simple superposition. The actual value of the synergistic combination is thus the sum of the individual values, minus their mutual redundancies, plus their synergies:

$$V_{\{i,j\}} = V_i + V_j - R_{\{i,j\}} + S_{\{i,j\}}:$$

Hence, another type of score can be defined that considers the interactions between the instruments and a predefined reference architecture. This leads to two definitions:

- **marginal descoping scores** $V_i^{'} = V_{\{ref\}} - V_{\{ref \setminus i\}}$;
- **marginal upgrading scores**: $V_i^{'} = V_{\{ref \cup i\}} - V_{\{ref\}}$.

In other words, this approach considers that the value of an instrument is equal to: a) the value that is lost when the instrument is deleted from a reference architecture; b) the value that is gained when the instrument is added to a reference architecture. Both of these definitions capture all synergies between this instrument and the reference.

The problem in general with this approach of marginal scores is that the definition of a reference architecture may not be trivial, and it is in any case "static in nature", as this reference may change over time. If the instrument selection problem was to be formulated in a recursive way (instruments are chosen one after the other), then the reference architecture could be defined as the cumulative subset of instruments selected at each stage in the recursion. Note that this recursive scheme would require exponential space to be formulated using dynamic programming, as there are *N* instruments, and each instrument can have 2^{N-1} scores corresponding to the number of subsets that can be constructed without that instrument. Such formulation could then be solved using any discrete-event approach such as Markov Decision Processes or Time-expanded Decision Networks (Silver & de Weck, 2007).

4.4.2 Lifecycle cost

Lifecycle cost, or affordability, is an extremely important figure of merit for EOSS and almost any other system. The metric we use in this framework is an assessment of the entire lifecycle cost of the system, as opposed to just a measurement of its development or fabrication cost. This is consistent with the major principle of system architecture of having a holistic, lifecycle approach to architecting.

Cost estimation methodologies are classically divided in three types: 1) top-down estimations, a.k.a. parametric cost models, or Cost Estimating Relationships (CERs); 2) bottom-up estimations; 3) analogy-based estimations (Apgar, Bearden, & Wong, 1999; Cost Analysis Division - NASA Headquarters, 2008; Shishko, 2004). These methodologies are often used in combination.

In order to develop a lifecycle cost metric for EOSS, we started with a bottom-up decomposition of total lifecycle cost into several tasks that loosely match mission lifecycle phases: 1) instrument development and fabrication; 2) bus development and fabrication; 3) integration, assembly and testing; 4) launch; 5) program overhead; 6) operations.

For each of these contributions, a top-down model is used typically based on mass. We tried to limit as much as possible the number of different sources for these CERs, since cost estimates from different CERs may produce inconsistent results. Also, we had access to very limited publicly available cost data due to nationality issues. Most CERs we used are directly taken from chapter 20 in the Space Mission Analysis and Design book (Apgar et al., 1999). The accuracy of the cost model could be increased if more modern CERs such as the ones in the NASA/Air Force Cost Model (NAFCOM) were used (NASA & US Air Force, 2002).

Finally, it is important to note that the goal of cost models for early system architecture tools is not to provide accurate absolute cost estimates, but rather to provide accurate relative cost assessments while showing enough sensitivity in cost to the major architectural trade-offs. In other words, we want figures of merit to be architecturally distinguishing, even at the expense of absolute accuracy. Hence for example, for the instrument packaging problem, complexity penalties were added that potentially exaggerate the mass and cost of complex spacecraft, in order to emphasize the trade-off between small satellites and large satellites.

The rules utilized for the different contributions to the overall lifecycle cost metric are discussed in the following paragraphs.

Instrument cost: The cost of developing and fabricating a remote sensing instrument depends on many parameters including mass, power, data rate, dimensions, number of channels, spectral resolution, and spatial resolution amongst others. Furthermore, the parameters that capture most of the variance in cost depend much on the type of technology. In fact, actual analyses of variance have shown a large dependence on instrument type for the optimal choice of these attributes (Agahi, Ball, & Fox, 2009).

That being said, when all instrument types are included, the three attributes best correlated to development and fabrication cost are mass, power, and data rate, in this order. The publicly available online version of the NASA Instrument Cost Model consists of a single CER that provides instrument cost as a function of these three parameters (Agahi et al., 2009):

$$\text{cost(FY04\$k)} = 25{,}600 \cdot \left(\frac{m(kg)}{53.8}\right)^{0.26} \cdot \left(\frac{P(W)}{61.5}\right)^{0.32} \cdot \left(\frac{R_b(kbps)}{40.4}\right)^{0.11}$$

This CER has an $R^2 = 0.77$ and is based on a database containing 37 instruments. Since this rule provides total instrument cost (recurring and non-recurring), it is assumed that 80% of the cost is non-recurring, and 20% recurring. In other words, the assumption implies that building a second copy of the same instrument would only cost 25% of the cost of developing and building the first unit of the instrument.

Note that the rule only computes the cost of the instrument if it has been developed domestically. International instruments receive a cost of zero in order to enable basic trade-offs in terms of international cooperation.

Bus cost: A bottom-up decomposition of both development and implementation cost of the bus is done in terms of subsystems: structure; electrical power; attitude determination and control; on-board data handing, tracking, telemetry and communications; propulsion; thermal control. For each subsystem, a different mass-based CER is used to estimate subsystem development/implementation cost as a function of mass, except for the electrical power subsystem, for which a simple power budget is computed. Hence, the power budget and mass budget rules presented earlier are required to produce the inputs of cost estimation rules. The different cost estimation rules are described below. These CERs are all taken from (Apgar et al., 1999). Note that different CERs are applied for recurring and non-recurring cost.

Table 13: Cost Estimating Relationships for spacecraft subsystem costs, taken from [120], pages 795-796

Subsystem	Recurring/ Non-recurring	Cost Estimating Relationship $cost_{sub,R/NR}(\$FY00k) = a \cdot m_{sub}(kg)^b$	Range of applicability (on $m_{sub}(kg)$)	Standard Error
Structure	Non-recurring	$a = 157;\ b = 0.83$	54-392	38%
Power	Non-recurring	$a = 62.7;\ b = 1.00$	31-491	57%
Thermal	Non-recurring	$a = 394;\ b = 0.635$	3-48	45%
Communications	Non-recurring	$a = 545;\ b = 0.761$	12-65	57%
ADCS	Non-recurring	$a = 464;\ b = 0.867$	20-160	48%
Propulsion	Non-recurring	$a = 17.8;\ b = 0.75$	81-966	N/A
Structure	Recurring	$a = 13.1;\ b = 1.00$	54-560	39%
Power	Recurring	$a = 112;\ b = 0.763$	31-573	44%
Thermal	Recurring	$a = 50.6;\ b = 0.707$	3-87	61%
Communications	Recurring	$a = 635;\ b = 0.568$	13-79	41%
ADCS	Recurring	$a = 293;\ b = 0.777$	20-192	34%
Propulsion	Recurring	$a = 4.97;\ b = 0.823$	81-966	20%

Launch cost: The launch cost model is based on the assumption that each spacecraft is launched individually, and that it falls into one of the launch vehicle categories defined in Table 11. The assignment of each spacecraft to one of these categories is done using some rules-of-thumb based on spacecraft mass and dimensions, orbit requirements, and cost. These rules have been described in subsection 4.2.2.4.

Integration, assembly and testing cost (IA&T): Integration, assembly and testing non-recurring cost is assumed to be a function of total spacecraft development cost (i.e., non-recurring bus + payload cost), as suggested in (Apgar et al., 1999). The specific CER used for non-recurring IA&T cost is provided below.

$$cost_{IA\&T,NR}(\$FY00k) = 989 + 0.215 \cdot cost_{sat,NR}(\$FY00k) \ ;$$

$$SE = 46\%\ cost_{sat,NR}(\$FY00k) \in [2{,}703; 395{,}529]$$

In terms of recurring cost, (Apgar et al., 1999) provides a CER as a function of spacecraft total mass including payload:

$$cost_{IA\&T,R}(\$FY00k) = 10.4 \cdot m_{sat}(kg);$$

$$SE = 44\%, m_{sat}(kg) \in [155; 1390]$$

Operations cost: Operations cost is far from being negligible when compared to the total lifecycle cost of most Earth observing missions. Rasmussen argues that approximately 8% of lifecycle cost is due to operations in "small missions", which is approximately equivalent to 20% of the bus cost (Rasmussen & Sun-synchronous, 1998). According to Rasmussen the same operations for a large spacecraft are only about 25% more expensive, which relative to the large bus cost represents a mere 7% as opposed to 20%. Hence the cost of operating a BCP-2000 class satellite is approximately 20% of \$40M = \$8M while the cost of operating a T-330 class satellite is 25% more i.e. \$10M. In this example, it the relative operations cost per instrument is much lower in the case of a large platform.

For the purpose of this model, we used the NASA NOCM as a baseline for operations cost. Such model is publicly available in the following NASA website: http://cost.jsc.nasa.gov/MOCM.html. Other models exist such as the NASA Spacecraft Operations Cost Model (SOCM), which is also available on-line[12]. The NASA MOCM provides a rough order of magnitude (ROM) estimate of the yearly cost of orbital operations for a spacecraft of a certain type and development cost. The CER used in this model for Earth observation satellites is provided below:

$$cost_{ops}(FY00\$) = 0.035308 \cdot \left(cost_{S/C}(FY00\$)\right)^{0.928} \cdot lifetime(yrs)$$

The standard error and range of applicability of this CER are unknown.

Program/overhead cost: Management cost, overhead cost, government oversight cost or program cost are all different names to refer to the cost of a program that is not directly associated to activities of design, engineering, fabrication, testing or operations. The CERs used for overhead cost are the ones provided in (Apgar et al., 1999). Program overhead non-recurring cost is a function of non-recurring spacecraft cost (including payload) as suggested by the following CER:

$$cost_{pro,NR}(\$FY00k) = 1.963 \cdot cost_{sat,NR}(\$FY00k)^{0.841}$$

This CER has a standard error of 36% over the range $cost_{sat,NR}(\$FY00k) \in [4{,}607; 523{,}757]$.

As for the recurring cost, it is a function of spacecraft recurring cost (also including payload):

$$cost_{pro,R}(\$FY00k) = 0.341 \cdot cost_{sat,R}(\$FY00k)$$

This CER has a standard error of 39% over the range $cost_{sat,R}(\$FY00k) \in [15{,}929; 1{,}148{,}084]$.

[12] http://cost.jsc.nasa.gov/SOCM/SOCM.html

One of the advantages claimed by supporters of small satellites is that they allow the application of streamlined management techniques which can severely reduce program management cost (see references (Sarsfield, 1998) and (Committee on Earth Studies, Space Studies Board, 2000)). Several arguments can be provided that intuitively explain why this is so:

- Meetings with a larger number of people are less effective
- Logistics in larger missions are more difficult than in smaller missions. Mathematically, if there are N instruments in a mission, the number of logistical problems that can appear is not proportional to N, but rather to all the possible combinations of N elements taken by pairs which is N*(N-1)/2).

Overhead cost is extremely difficult to measure and even more to estimate. In (Rasmussen & Sun-synchronous, 1998), Rasmussen estimates that management cost is about 8.9% of bus cost for small missions and 9.3% of bus cost for large missions. Although these figures are not conclusive, they tend to support the idea that overhead cost tends to be growing worse than linearly for larger missions. This suggests that the overhead cost model presented above could be modified to add an organizational penalty that takes into account these facts. Another potential feature to be added to the model would be an organizational penalty that would take into account whether different organizations are involved in a given mission. Multi-organizational missions may incur larger overhead costs due for example to duplication of some positions, and more resource-consuming meetings between organizations. The study of this and other penalties is left for future work.

Cost overrun: An expected cost overrun is added to the total initial lifecycle cost because it is noted that the probability of schedule slippage and cost overrun is larger in larger missions. Some studies have suggested that the probability of schedule slippage is correlated with the maturity of the technology used in the mission, as described by the initial mission TRL (Dubos, Saleh, & Braun, 2008). The expected cost overrun is computed in two steps:

1. Given the initial TRLs of all the instruments in a given spacecraft, the expected schedule slippage in % of total development time is computed as a function of the lowest instrument TRL in the spacecraft, using the findings in [126]:

$$RSS\ (\%) = 829 \cdot e^{-0.56 \cdot min_{TRL}}$$

2. This expected schedule slippage is transformed into a cost overrun by applying the relationship found in (Weigel & Hastings, 2004b):

$$\text{cost overrun}\ (\%) = \ 0.24 \cdot RSS(\%) + 1.7$$

While some recent articles have discussed the validity of using TRL as a quantitative metric for this purpose (Conrow, 2011), the idea behind the study, namely that less mature technologies are more likely to suffer from schedule slippage, remains a fair assumption.

Summary: The lifecycle cost model is summarized in the table below, that lists all the contributions to the metric, and a short explanation of how they are quantified. Since contributions come from different sources, all contributions are corrected for inflation and translated into $FY00M.

As noted earlier, the model allows sharing part of these costs with international partners. Total lifecycle cost is thus the addition of all these costs, with an expected cost overrun.

Table 14: Summary of main assumptions in lifecycle cost model

Lifecycle cost contribution	**Major models and assumptions**
Instrument development	80 % NICM cost (Agahi et al., 2009)
Instrument TFU	20 % NICM cost (Agahi et al., 2009)
Bus development	SMAD CERs (Apgar et al., 1999)
Bus TFU	SMAD CERs (Apgar et al., 1999)
Integration, Assembly & Testing	SMAD CERs (Apgar et al., 1999)
Launch	7 launch vehicle classes: Atlas-5, Delta-7920, 7420, 7320, Minotaur-IV, Taurus XL, Taurus, Pegasus XL. Costs given in Table 11.
Program overhead	SMAD CERS (Apgar et al., 1999)
Operations	NOCM[13]
Cost overrun	Schedule slippage as a function of minimum instrument TRL [126] Cost overrun directly proportional to schedule slippage(Weigel & Hastings, 2004b)

Concerning the possibility of having international partners, this option is taken into account through the use of the partnership mission attribute. The value of this attribute acts as a mask for the different contributions to the lifecycle cost, so that some of the contributions to the total lifecycle cost can be made zero depending on the type of agreement. Hence for example, if a domestic instrument is flown on an international spacecraft launched by the international partner, the user may choose to not count for the bus and launch cost, and possibly other parts of the lifecycle cost (e.g., operations).

[13] http://cost.jsc.nasa.gov/MOCM.html

Similarly, if an international instrument is flown on a domestic mission, the cost of developing and building this instrument will not be taken into account for the overall lifecycle cost. This allows for a more realistic estimation of the costs incurred from the point of view of a particular agency.

4.4.3 Programmatic risk

In the context of this thesis, programmatic risk refers to risk of schedule slippage and cost overrun. More precisely, we are interested in the components of this programmatic risk that are architecturally distinguishing. Hence, causes of programmatic risk that do not vary across architectures are not considered.

The programmatic risk metric is used both in the instrument selection and instrument packaging problems, but not in the mission scheduling problem. The formulation of the programmatic risk metric in the instrument selection is different from the formulation in the instrument packaging problem, in order for the metrics to be architecturally distinguishing in each SAP.

In the instrument selection problem, an architecture with high programmatic risk is one in which several "high risk" instruments are selected. A "high risk" instrument is defined as an instrument for which substantial investment still needs to be made in order for the instrument to be ready to be flown. While the amount of investment that needs to be done on an instrument in order to achieve flight qualification is hard to predict, it seems intuitive that the initial TRL of the instrument will be somewhat correlated. However, as mentioned before, attempts to use TRL as a cardinal metric have been rather unsuccessful. In order to avoid this issue, we define the programmatic risk metric in the instrument selection problem as the fraction of instruments that have initial TRL < 5.

$$risk_{prog,SEL} = sum_i(f_i);$$

$$f_i = \begin{cases} 1 \; if \; TRL_i < 5 \\ 0 \; otherwise \end{cases}$$

A TRL of 5 implies the validation of the system at the component level in a relevant environment as opposed to a laboratory environment. Many senior systems engineers agree that the step from TRL 4 to TRL 5 is the hardest to achieve and the one that requires the most significant investment.

In the instrument packaging problem, the previous definition of programmatic risk is not architecturally distinguishing, as all packaging architectures would have the same score under this metric. Instead, the following definition of the programmatic risk metric for the instrument packaging problem is architecturally distinguishing:

$$risk_{prog,PACK} = avg_s(\underset{i \in sat,s}{\text{avg}} TRL_i - \underset{i \in sat,s}{\min} TRL_i)$$

The idea behind the packaging programmatic risk metric is to avoid situations in which a high risk instrument could delay the deployment of important mature instruments. A high value of $risk_{prog,PACK}$ indicates that there is a general imbalance in the TRLs of the instruments, that could be improved if instruments with similar levels of maturity were binned together.

4.4.4 Launch risk

During the qualitative discussion of the instrument packaging problem in section 4.1.2.2, the issue of launch risk was raised. Distributed architectures are perceived as more desirable than monolithic architectures in terms of launch risk, because they are more robust to a single launch failure. Although real launch risk measured as the average number of instruments successfully put into orbit is independent of the packaging architecture assuming identical launch vehicle reliabilities, perceived launch risk varies across architectures due to risk aversion.

In order to model this risk aversion, I explored two possibilities: 1) incorporating a concave risk aversion curve to the number of instruments successfully put into orbit and computing the average after applying risk aversion; 2) using the concept of entropy from information theory.

In the first approach, the risk metric is simply computed as the average utility of the number of instruments successfully put into orbit, where this utility function is for example a logarithmic curve. Let *NI* be the number of instruments in a program, and *N* be a random variable representing the number of instruments successfully put into orbit by an architecture. Then, the launch risk metric is defined as:

$$risk_{launch} = \sum_{n=0}^{NI} Prob\{N = n\}u(n) = \sum_{n=0}^{NI} Prob\{N = n\}\frac{log(1+n)}{\log(1+NI)}$$

Considering a launch as a Bernoulli trial with probability of success equal to the reliability of the launch vehicle R_{LV}, $Prob\{N = n\}$ is related to the probability of having NL successful launches, which can be computed using a binomial distribution:

$$Prob\{NL = nl\} = \binom{NL}{nl} R_{LV}^{nl}(1 - R_{LV})^{NL-nl}$$

The risk metric can thus naively be calculated for each architecture by computing the probability $Prob\{NL = nl\}$ and the utility of the corresponding number of instruments successfully put into orbit for each of the 2^{NS} cases (each of the NS launches can either be a success or a failure, which explains the 2^{NS}).

In the second approach, the launch risk metric is computed as follows. Let p be an array containing the number of instruments in each satellite, and $\hat{p}$ be the convex normalization of p (i.e. $\widehat{p_i} = \frac{p_i}{\sum_i p_i}$). $\widehat{p_i}$ can be interpreted as the probability of a random instrument to belong to satellite i. The risk metric can now be defined as one minus the entropy of this probability distribution:

$$risk'_{launch} = 1 - h(\widehat{p}) = 1 + \sum_{i=0}^{NS} \widehat{p_i} \log_2(\widehat{p_i})$$

The entropy of a probability distribution of length NS is a real value between 0 and $\log_2 NS$. The value of 0 is achieved by a delta probability distribution with all its weight in a single value. This is the equivalent of the monolithic architecture with only one satellite in the instrument packaging problem. The value of $\log_2 NS$ is achieved by the uniform probability distribution that assigns identical weight to all possible values. This is the equivalent of a completely distributed architecture with as many instruments as satellites. Note that in this case $\log_2 NS = \log_2 NI$.

4.4.5 Discounted benefit

As mentioned before, (undiscounted) scientific benefit cannot be used as a metric of the mission scheduling problem because all scheduling architectures have the same undiscounted scientific benefit. An architecturally distinguishing version of scientific benefit for the mission scheduling problem is discounted scientific benefit, or simply discounted benefit.

The main idea behind discounted benefit is that science today is better than science tomorrow. For example, reducing the uncertainty about climate change tomorrow rather than in ten years has a certain value, which can be computed for example using a value of information approach (Nordhaus, 1998). Therefore, it makes sense to discount scientific benefit in a similar way to what is done with cash flows in finance.

In finance and project management, the net present value of a project can be computed as the sum of the yearly profits (revenues minus costs) discounted in time in order to capture the opportunity cost of investing money on the project at hand instead of on an alternative project.

$$NPV = \sum_t \frac{R_t - C_t}{(1+r)^t}$$

The rate *r* at which cash flows are discounted is chosen to reflect the interest rate that the investment could bring on an alternative project, and it typically ranges from 0% to 20%, with lower values for public endeavors (usually between 3% and 10%).

In the context of this thesis, different discount rates are allowed for different panels, and therefore scientific benefit is defined as the sum over the panels of the total discounted benefit to each panel:

$$DB(A) = \sum_p \sum_t \frac{B_{p,t}(A)}{\left(1+r_p\right)^t}$$

where $B_{p,t}(A)$ is the benefit provided by architecture A to panel *p* at time period *t*, and r_p is the discount rate for panel *p*. The discount rate for each panel intuitively reflects the opportunity cost of investing on a mission that benefits a certain panel instead of another one.

where $B_{p,t}(A)$ is the benefit provided by architecture A to panel *p* at time period *t*, and r_p is the discount rate for panel *p*. The discount rate for each panel intuitively reflects the opportunity cost of investing on a mission that benefits a certain panel instead of another one.

In finance, discounting makes cash flows in the present or near future more important than cash flows in the far future. The same is true for discounted benefit, since for reasonable discount rates, it depends on the benefits of the first few missions.

The discounted benefit metrics has been successfully applied in the past in the context of three master thesis (Colson, 2008; Seher, 2009; Suarez, 2011).

4.4.6 Fairness

The notion of fairness across panels or scientific disciplines appears for the mission scheduling problem in the three aforementioned thesis (Colson, 2008; Seher, 2009; Suarez, 2011).

In Colson's OPN formulation, the scheduling problem is a recursive process where a decision about the next mission to fly is made at each time period. In this context, the notion of fairness was introduced through a constraint that restricted the set of missions available to choose to the subset of available missions that provided value to the two least satisfied panels. Hence, if at time the two least satisfied panels are panel 1 and panel 2, any mission not giving any value to at least one of these panels cannot be chosen in the next stage.

Seher and Suarez captured fairness as an a posteriori soft constraint based on the separation between the $B_{p,t}$ curves. Intuitively, architectures that lead to large deviations between the $B_{p,t}$ curves got lower fairness scores.

In addition to the mission scheduling problem, the notion of fairness is also applicable to the instrument selection problem, as a certain subset of instruments may be biased towards a given discipline. In this case, this notion can be simply captured as a hard constraint on the minimum over the panels of the benefit of the architecture to any panel:

$$f(A) \equiv \min_{p} \sum_{i} B_{i,p}(A) \geq f_{min}$$

4.4.7 Data continuity

Finally, an extremely important metric in EOSS is the ability of a certain architecture to cover potential data gaps in the time series of measurements with long records such as ocean color (CZCS, SEAWIFS, MODIS/MERIS), ocean altimetry (TOPEX-POSEIDON, Jason, OSTM), or aerosol optical depth (AVHRR, MODIS).

Let us consider a 2D observations matrix O where the columns indicate time and the rows indicate measurements that require continuity. Each element of this matrix contains the number of satellites that perform measurement m at time t. Hence, $O(m,t) = 0$ if no satellites perform measurement m at time t (yielding a data gap), and $O(m,t) > 0$ otherwise.

Under this scenario, a data continuity metric can be computed by computing the delta in the observations matrix with and without the architecture, and summing the resulting delta matrix $D(i,j) = \frac{\left(O_{with\ A}(i,j) - O_{w/o\ A}(i,j)\right)}{O_{w/o\ A}(i,j)}$ with certain weights that represent: 1) the relative importance of the potential data gaps; 2) the relative importance of gaps in the future with respect to gaps now. Let $W(i,j)$ be the separable matrix containing such weights, $W(i,j) = w_m(i) \cdot w_t(j)$. Then, the data continuity metric can be computed as:

$$DC \equiv \sum_i \sum_j D(i,j) \cdot W(i,j) = \sum_i \sum_j \frac{\left(O_{with\ A}(i,j) - O_{w/o\ A}(i,j)\right)}{O_{w/o\ A}(i,j)} \cdot \left(w_m(i) \cdot w_t(j)\right)$$

However, the more positive $O_{w/o\ A}(i,j)$ is, the lower the marginal value of an additional observation of that measurement should be. This effect can be taken into account by slightly changing the definition of $D(i,j)$:

$$D(i,j) \equiv \sum_{k=1}^{\left(O_{with\ A}(i,j) - O_{w/o\ A}(i,j)\right)} \frac{1}{O_{w/o\ A}(i,j) + k}$$

Hence for example, if $O_{w/o\ A}(i,j) = 1$ and $O_{with\ A}(i,j) = 2$, $\sum_{k=1}^{\left(O_{with\ A}(i,j) - O_{w/o\ A}(i,j)\right)} \frac{1}{O_{w/o\ A}(i,j)+k} = \frac{1}{2}$, and if $O_{w/o\ A}(i,j) = 5$ and $O_{with\ A}(i,j) = 8$, then $\sum_{k=1}^{\left(O_{with\ A}(i,j) - O_{w/o\ A}(i,j)\right)} \frac{1}{O_{w/o\ A}(i,j)+k} = \frac{1}{6} + \frac{1}{7} + \frac{1}{8}$

4.4.8 Discussion

In this section, the FOMs that are used in the three EOSS SAPs have been described. A list of these metrics with a succinct description and a mapping to the SAPs that utilize them is provided in Table 15.

In addition to these metrics, and according to what has been described when down-selection rules were introduced in section 2.3.6, the three SAPs utilize Pareto ranking to discard "highly dominated" architectures, and multi-attribute utility to discard "unbalanced" architectures. Pareto rankings only include lifecycle cost and science, or data continuity and discounted benefit in the case of mission scheduling. The other metrics are typically less architecturally distinguishing, i.e., the number of different values in the image domain is smaller, and therefore they are used as hard constraints to discard poor architectures, rather than using them for computing Pareto rankings.

Table 15: Summary of the metrics utilized to architect EOSS

Metric	Description	Selection	Packaging	Scheduling
Science	Uses VASSAR (capability, emergence, requirement rules)	X	X	
Lifecycle cost	Uses VASSAR (orbit selection, power, mass, volume budgets, LV selection, sat sizing with p/l interferences, NICM, CERs)	X	X	
Programmatic risk (v1)	% of instruments in program that require significant development (TRL < 4)	X		
Programmatic risk (v2)	Penalizes architectures where high risk instruments may delay more mature instruments $\frac{}{(avg_{sat\ i}(TRL) - min_{sat\ i}(TRL))}$		X	
Launch risk	Fewer sats ➔ higher perceived risk. Metric is an average of the probability distribution of putting N=#instr put into orbit given #launches and reliabilities weighted by risk aversion		X	
Fairness (v1)	Min panel score is a soft indicator of fairness	X		
Fairness (v2)	A more precise indicator is the deviation between curves of value delivery to panels over time			X
Data continuity	Favor architectures which close important potential data gaps			X
Discounted value	Favor sequences which fly valuable missions first			X

4.5 Case studies and methodology applied to the case studies

Three case studies are analyzed in this thesis: 1) The NASA Earth Observing System; 2) The NRC Decadal Survey; 3) The Iridium GEOScan program.

The NASA Earth Observing System (EOS) was a multi-billion Earth Science program started at NASA in the 80's. Originally intended to consist of two extra-large polar platforms carrying about 40 different instruments, EOS ended up consisting of several medium and small satellites carrying a subset of the original instruments. The NASA EOS case study was chosen as a retrospective case study with the aim of providing partial validation of the methodology and tool.

The second case study concerns the Earth Science Decadal Survey (DS). The DS is a current multi-billion program that intends to fulfill the requirements of all the disciplines of the Earth sciences for the next decade.

A reference architecture exists for the DS that was laid out by an NRC committee of experts. This reference architecture consists of 17 missions, from which only two are past conceptual studies (namely SMAP and ICESAT-II, with launch dates in 2015). The goal of this case study is to assess the reference architecture, and potentially to identify interesting alternative architectures.

The third and final case study concerns the Iridium Next Generation Hosted Payload program, and more particularly, APL/Draper's GEOScan program. Iridium LLC is developing their new generation constellation of communication satellites in LEO, and they are offering a hosted payload of about 50kg and 50W for Earth observation in each of their 66 satellites (plus spares). APL with the collaboration of Draper Laboratory has created the GEOScan program, in which the instruments to be put in this hosted payload opportunities are analyzed from a global perspective. The GEOScan architecture is now almost defined, with a system sensors suite of six instruments that will be flown on each of the 66 spacecraft, plus some slots on each spacecraft for other hosted payloads from universities or other organizations. The goal of this case study is to identify the set of optimal system sensors to be flown, as well as the optimal number of each type of sensor.

These three case studies are different in nature, and thus stretch the capabilities of the tools and methods in different ways. The NASA EOS case study is better than the two others in terms of data availability, because being a past program there is a huge literature available both on scientific requirements and instrument capabilities. On the other hand, it is extraordinarily complex from the systems engineering perspective due to the four consecutive descoping processes it suffered in the late 80's and early 90's. It is rich in budget pressure, international cooperation, and other policy elements that are hard to include in the model. The DS case study is hard because there is a lot of uncertainty in the capabilities of the instruments (which for the most part do not yet exist), and also some ambiguity in terms of the scientific priorities. Furthermore, computationally it is a harder problem because there is a larger number of missions and instruments to consider. Finally, the difficulty of the GEOScan case study is two-fold: first, it is very important to be able to model the differences between typical remote sensing instruments and Cubesat-class instruments; second, it is also important to quantify the effect of increasing the number of satellites on the scientific value of provided to different stakeholder groups.

The methodology applied to the three case studies is illustrated in Figure 34.

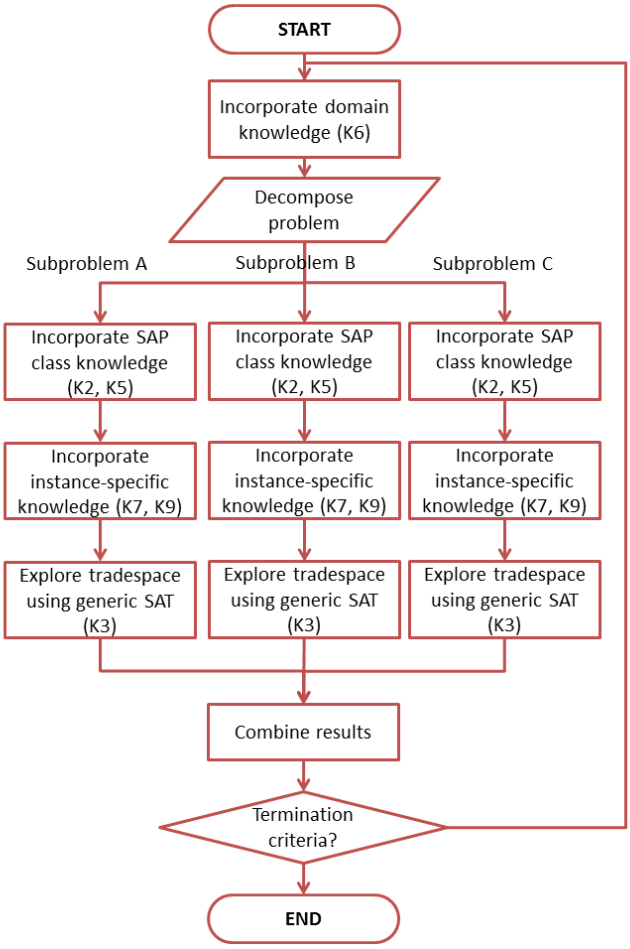


Figure 34: Methodology used to solve the three EOSS case studies

Domain-specific knowledge that is common to the three case studies, such as orbit selection rules, and complexity-corrected mass budgets, is available in the RBES(K6). Each case study is decomposed into three sub-problems: the instrument selection problem, the instrument packaging problem, and the mission scheduling problem.

The instrument selection problem is formulated as a selecting problem. The class-specific rules corresponding to the DsP are used (K2, K5). Case study specific rules including requirement rules, aggregation rules, and capability rules are then incorporated (K7, K9). The RBES is then used to explore the architectural tradespace corresponding to this particular view.

The same procedure is applied to the instrument packaging problem, which is formulated as a partitioning problem. The mission scheduling problem is formulated as a permuting problem. Once the three SAPs have been individually solved, their results are combined to produce a subset of preferred complete architectures. At this point, it may be necessary to refine the model by adding or removing rules, and iterate.

The results for the three case studies are presented in the next three chapters. The structure of these chapters is the following:

- First, the historical context and goals of the case study are discussed.
- Second, the EOSS-specific knowledge that is common to the three problems is introduced: aggregation rules, requirement rules, and capability rules.
- Third, a few intermediate results are given that are used as inputs in the three problems: instruments and mission scores (marginal scores and scores in isolation), S-DSM and E-DSM, and data continuity.
- Fourth, the results concerning the instrument selection problem are introduced and discussed.
- Fifth, the results concerning the instrument packaging problem are introduced and discussed.
- Sixth, the results concerning the mission scheduling problem are introduced and discussed.
- Finally, the chapter is concluded with a discussion about the results of the theirs problems.

5 Case Study 1: The NASA Earth Observing System (EOS)

5.1 Context and goals for the case study

In the early 1980's, a growing effort appeared at NASA to create a large-scale Earth Observing program. Eventually, in 1990, Congress authorized $132 million in new appropriations for NASA to begin the development of its Earth Observing System (EOS) (Leshner, 2007).

At that time, EOS was a $17B program over ten years, planning to launch six Shuttle-class satellites over a 15-year period. Five years later, after going through four major restructuring and descoping processes, the planned budget for EOS fell below $8B. Fifteen instruments initially preselected were cancelled. The use of international cooperation as a means to share costs was also intensified in order to share costs. The sequence of descoping processes from the initial EOS presented to the Office of Management Budget in 1989 to the finally implemented EOS is illustrated in Figure 35. Red ovals in Figure 35 represent the descoping processes, and blue rectangles represent the architectures before and after each descoping process.

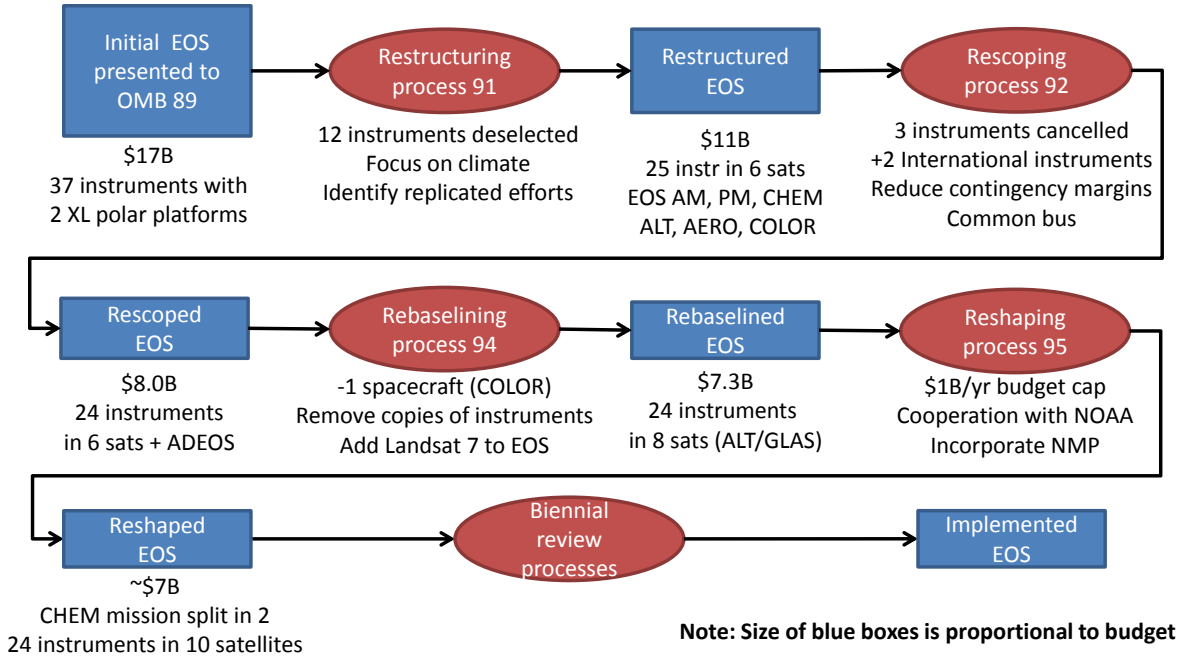


Figure 35: Summary of descoping processes for the NASA EOS between 1989 and 1995

During these descoping processes, the architecture of EOS went from being based on two extra-large Shuttle-class platforms to a more diverse architecture with three mid-size satellites (Terra, Aqua, Aura) and several smaller satellites. This evolution is illustrated in Figure 36. In Figure 36, rectangles represent spacecraft, and text inside rectangles specifies spacecraft name, mass, and launch vehicles in black text on top followed by instruments on red (cancelled), green (added), or white (maintained). Instruments on cursive come from international partners.

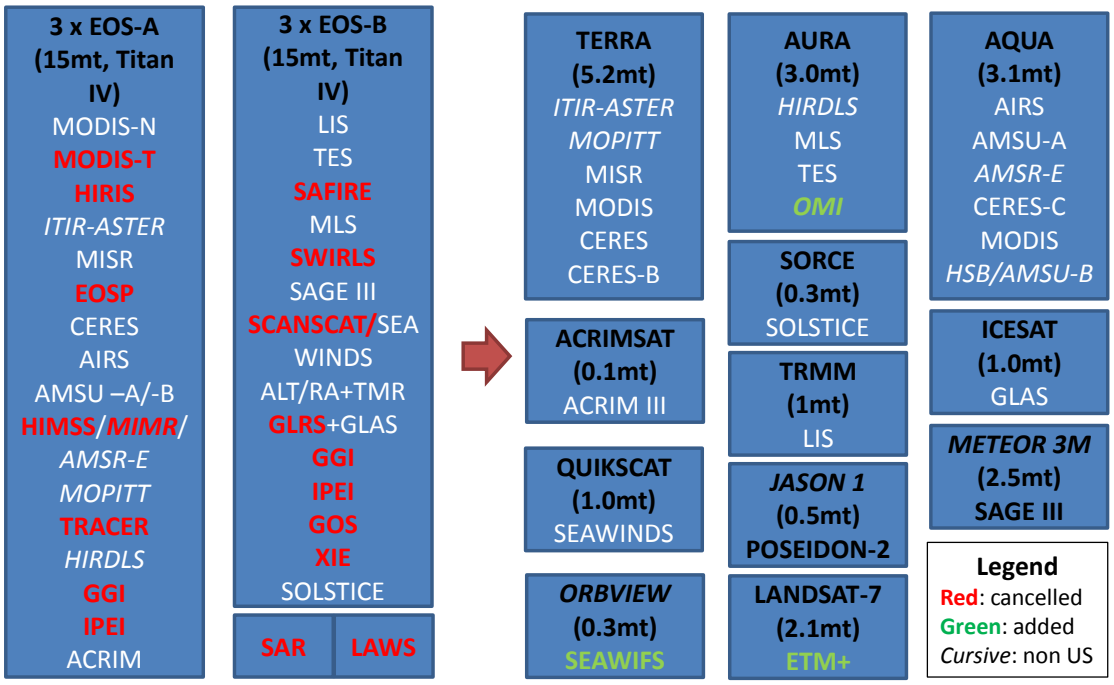


Figure 36: Overall changes in selection and packaging architectures for the EOS program between 1989 and 1995

This case study is retrospective in nature. Substantial documentation exists concerning the scientific requirements of the EOS program (Asrar & Dozier, 1994; Butler et al., 1984; King & Greenstone, 1999; Tuyahov, 1986), the capabilities of the different instruments considered for the EOS program (Abshire, Smith, & Schutz, 1998; Aumann & Miller, 1995; R. A. Barnes & Holmes, 1993; Fu et al., 1994; Gille et al., 2003; Goetz & Herring, 1989; Levelt et al., 2006; Long, Freilich, & Leotta, 1990; Schoeberl et al., 2006; Waters et al., 2006; Wielicki, Barkstrom, & Harrison, 1996; Willson, 2001), and the details of the different restructuring processes (Leshner, 2007), (Moore III et al., 1991), (Scolese & Bordi, 1990). Many of the scientists and engineers in charge of making those decisions are also available to provide testimony on the rationale behind their decisions.

The goal of this first case study is to take advantage of its retrospective nature, and of the quantity and quality of the available information, in order to assess the validity of the tool. This is done by applying the methodology presented in the previous chapters to the information available at the time of the development of the EOS program, and attempting to replicate the decisions that were made. While this does not formally validate the tool, it does give us some confidence before applying it to the two other case studies, which are predictive in nature.

5.2 Case study specific rules

5.2.1 Aggregation rules

5.2.1.1 1st level of decomposition of stakeholder needs: panels

Measurement requirements were categorized in seven panels for the EOS case study, according to the EOS science plan (King, 1990): water and energy cycles (WAE), ocean circulation and productivity (OCE), tropospheric chemistry and greenhouse gases (GHG), land ecosystems and hydrology (ECO), cryospheric systems, ozone and stratospheric chemistry (ICE), and volcanoes and climate effects of aerosols (SOL).

These seven panels were all equally weighted in importance except for the ozone panel, which was given a weight half of the other panels[14], as shown in Table 16. The lower weight for the ozone panel is justified by the existence of a large mission outside of the EOS program, entirely devoted to ozone and stratospheric chemistry, namely the UARS mission (Upper Atmosphere Research Satellite) (Reber, Trevathan, & McNeal, 1993). UARS was launched in 1991 and successfully carried out its scientific mission for more than 14 years until its operation was ceased in 2005 due to budget cuts. UARS carried several instruments that took valuable stratospheric measurements, including a microwave limb sounder, several other chemistry instruments, and copies of the SOLSTICE and ACRIM instruments for radiation budget studies.

Table 16: 1st level of decomposition of stakeholder needs for the EOS case study

Panel	Id	Description	Weight
Clouds and radiation	WAE	Cloud formation, dissipation, and radiative properties, which influence response of the atmosphere to greenhouse forcing	2/13=15.4%
Oceans	OCE	Exchange of energy, water, and chemicals between the ocean and atmosphere, and between the upper layers of the ocean and deep ocean	2/13=15.4%
Greenhouse	GHG	Chemistry of the troposphere and lower stratosphere	2/13=15.4%

[14] Personal conversation with Dr Christopher Scolese, director of NASA GSFC

Gases			
Land & Ecosystems	ECO	Land hydrology and ecosystem processes	2/13=15.4%
Glaciers and Polar Ice Sheets	ICE	Glaciers, Sea Ice, and Ice Sheets	2/13=15.4%
Ozone and Stratospheric Chemistry	OZO	Ozone and Chemistry of the upper stratosphere	1/13=7.7%
Solid Earth	SOL	Volcanoes and Climate Effects of Aerosols	2/13=15.4%
			100%

5.2.1.2 2nd level of decomposition of stakeholder needs: panel objectives

Scientific objectives were identified and ranked in importance for each of the seven panels, based on the information available in (Asrar & Dozier, 1994; Butler et al., 1984; King & Greenstone, 1999; Tuyahov, 1986), and (King, 1990) amongst others. This information was contrasted with several experts that were involved in the early development of the EOS program. For example, eight objectives were identified for the WAE panel. They are listed in Table 17.

Table 17: 1st level of decomposition of stakeholder needs for the EOS case study: WAE panel objectives

Objective	Id	Description	Weight
1	WAE1	Atmospheric circulation	13%
2	WAE2	Cloud radiative feedback	19%
3	WAE3	Precipitation patterns	13%
4	WAE4	Water vapor	13%
5	WAE5	Aerosols	13%
6	WAE6	Radiation budget	19%
7	WAE7	Ice and snow	6%
8	WAE8	Land Surface Water	6%
			100%

These eight objectives were ranked in three groups of importance. Cloud radiative feedback and radiation budget were assigned higher priority (3/16) because of their very close relationship to key climate variables. Ice and snow and land surface water were assigned the lowest priority (1/16) because these objectives are primary objectives of other panels, namely the cryosphere and ecosystem panels. The other objectives were assigned medium priority (2/16).

Details about the 2nd level of decomposition of stakeholder needs for the other six panels are provided in the Appendix 9.3.1. A total of 43 objectives were identified.

5.2.1.3 3rd level of decomposition of stakeholder needs: "subobjectives"

In the second and last level of decomposition of stakeholder needs, each of the 43 objectives was decomposed into as many subobjectives as required in order that one subobjective corresponds to exactly one measurement. 121 subobjectives were thus identified. An example of this decomposition is shown in Table 18 for the radiation budget objective of the WAE panel.

Table 18: 2nd level of decomposition of stakeholder needs for the EOS case study: subobjectives for radiation budget

Objective 6	**WAE6**	**Radiation budget**	
Subobjective	**Id**	**Description**	**Weight**
1	WAE6-1	Total solar irradiance	15%
2	WAE6-2	Short-wave radiation (solar reflected)	35%
3	WAE6-3	Long-wave radiation (thermal emission)	35%
4	WAE6-4	Albedo and reflectance	15%
			100%

In this case, short-wave and long-wave radiation measurements are assigned a higher priority than the total solar irradiance and albedo measurements (35% to 15%) because they contain more information. Indeed, total solar irradiance can be inferred from the SW+LW outgoing flux assuming equilibrium, but the opposite is not true, SW and LW outgoing fluxes cannot be inferred from total solar irradiance.

More information about the 3rd level decomposition of the rest of EOS objectives is available in the Appendix 9.3.1.

5.2.2 Requirement satisfaction rules

The scientific and societal objectives of the EOS program necessarily changed during the descoping and restructuring processes. A good description of the initial requirements can be found in the final report of the science and mission requirements working group (SMRWG), released by NASA in 1984 (Butler et al., 1984). Table 19 was taken from this report, and summarizes some of the major measurement requirements for EOS at that time. Note for example that the approach proposed to cover several of the measurement requirements is the use of SAR, but this instrument was subsequently deleted.

For each of the 121 subobjectives, rules were created that express the measurement requirements for full satisfaction, and several cases of degraded satisfaction. A sample of such rules is provided in Table 20 for the case of subobjective WAE1-1 concerning atmospheric temperature fields. Table 20 shows the full satisfaction rule for subobjective WAE1-1, as well as several partial satisfaction rules. Note that this table only includes a subset of the attributes used in the rule. The actual rules include additional requirements in terms of cloud mask, spectral sampling, and accuracy amongst others. Additional information about the EOS requirements rules is provided in the Appendix 9.3.2. The complete set of requirement satisfaction rules for the EOS case study can be found in http://web.mit.edu/dselva/www/RBES/EOS/.

Table 19: Some initial measurement requirement for the EOS program from the SMRWG report (1984)

PARAMETER	APPLICATION	ACCURACY DESIRED	ACCURACY REQUIRED	APPROACH	SPATIAL RES	OBSERVATION FREQUENCY
Soil Features						
• Moisture	Hydrologic and geochemical cycles	5 moisture levels	5 moisture levels			
• Surface		5%	10%	Microwave Radiometer	1-10 km	2 day
• Root Zone		5%	10%	Model	30-1000 m	1 week
• Types-Areal Extent (peat, wet lands)	Geochemical cycles Agricultural & Forestry	10%	10%	Visible/SAR	30 m	annual
• Texture-Color	Agriculture & Forestry	10%	10%	Visible/SAR	30 m	annual
• Erosion	Geochemical cycles	10%	10%	Visible/SAR	30 m	annual
• Elemental storage	Geochemical cycles					
• Carbon		10%	10%	Visible/SAR	30 m	monthly
• Nitrogen		10%	10%	Visible/SAR	30 m	monthly
• Permafrost	Geochemical	10%	10%	Visible/SAR	30 m	annual
Surface Temperature						
• Land	Primary production, soil moisture and respiration	0.5°C	±1°C	Thermal IR	1 km ± 0.5 km	12 hours
• Inland Waters	Mass/Energy Flux	0.1°C	0.5°C	Thermal IR	30 m	12 hours
• Ocean	Mass/Energy Flux	0.1°C	0.5°C	Thermal IR, Microwave	4 km (open ocean)	12 hours
					1 km (coastal ocean)	12 hours
• Ice	Mass/Energy Flux	0.5°C	1°C	Microwave, Thermal IR	1 km	1 day
Vegetation						
• Identification	Hydrologic cycle, biomass distributions and change, primary production, plant productivity, respiration, nutrient cycling, trace gas, source sinks, vegetation-climate interaction, microclimate	1%	5%	Visible, Near IR, Thermal IR	1 km	7 day
• Areal Extent		1%	10%	Visible, Near IR, Thermal IR	30 m	30 day
• Condition (stress, morphology, phytomass)		10%	15%	Visible, Near IR Thermal IR, SAR	30 m	3 day
• Leaf area index canopy structure and density		10%	20%	Visible, Near IR, Thermal IR, SAR	30 m	3 days

Table 20: Requirement satisfaction rule for atmospheric temperature fields for the EOS case study (partial view)

rule	value	description	Measurement	Attribute-value1	Attribute-value2
nominal	100%	"Conditions for full satisf	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	All-weather yes
degraded-1	50%	"Insufficient temporal re	1 Atmospheric temperature fi	**SameOrBetter Temporal-resolution Medium-1day**	All-weather yes
degraded-2	50%	"Insufficient vertical spat	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	All-weather yes
degraded-3	75%	"no hyperspectral (but gc	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	All-weather yes
degraded-4	66%	"no all weather capabilit	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	**All-weather no**
degraded-5	50%	"insufficient accuracy"	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	All-weather yes
degraded-6	50%	"insufficient sensitivity in	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	All-weather yes
degraded-7	66%	"missing cloud mask"	1 Atmospheric temperature fi	SameOrBetter Temporal-resolution High-12h-24h	All-weather yes

5.2.3 Instrument capability rules

After describing stakeholder needs, the second important group of case study specific rules concerns instrument capabilities. 43 instruments were considered in this case study, and several characteristics necessary to assess requirement satisfaction and to perform lifecycle cost estimations were identified for each instrument.

Table 21 provides a snapshot of a few characteristics for a subset of the EOS instruments.

Table 21: Snapshot of the table containing EOS instrument characteristics

Instrument	Attribute-value1	Attribute-value2	Attribute-value3	Attribute-value4	Attribute-value5	Attribute-value6	Attribute-value7
ACRIM	All-weather no	Angular-resolution-azimuth# nil	Angular-resolution-elevation# nil	average-data-rate# 0.001	average-power# 10	Concept "A sun-looking self-calibrating active cavity radiometer for direct measurement of total solar irradiance"	Day-Night Day-only
AIRS	All-weather no	Angular-resolution-azimuth# 1.1	Angular-resolution-elevation# 1.1	average-data-rate# 1.27	average-power# 180	Concept "A high spectral resolution grating spectrometer containing 2378 infrared channels for obtaining atmospheric temperature profiles -sounding-"	Day-Night Day-and-night
ALT-SSALT	All-weather yes	Angular-resolution-azimuth# nil	Angular-resolution-elevation# nil	average-data-rate# 0.1	average-power# 214	Concept "A dual-frequency radar altimeter for oceanography"	Day-Night Day-and-night
AMSR-E	All-weather yes	Angular-resolution-azimuth# nil	Angular-resolution-elevation# nil	average-data-rate# 0.125	average-power# 350	Concept "A 12-channel microwave radiometer designed to monitor a broad range of hydrologic variables, including precipitation, cloud water, water vapor, sea surface winds, sea surface temperature, sea ice, snow, and soil moisture."	Day-Night Day-and-night

Furthermore, for each instrument, a rule was created that asserts the measurements that the instrument can take. An example of such rule for the TES instrument is provided in Code 23.

```
(define-rule CAPABILITIES::TES-measurements
   "Define measurement capabilities of instrument TES"
IF there is a (Manifested-instrument (with Id ?id) (and Name TES))
   =>
ASSERT Measurement (Parameter "1.8.2 O3") (Id TES1))
ASSERT Measurement (Parameter "1.8.1 H2O") (Id TES2)))
ASSERT Measurement (Parameter "1.8.4 CH4") (Id TES3)))
ASSERT Measurement (Parameter "1.8.5 CO") (Id TES4)))
ASSERT Measurement (Parameter "2.5.1 Surface temperature -land-") (Id TES5)))
ASSERT Measurement (Parameter "1.2.1 Atmospheric temperature fields") (Id TES6)))
```

```
ASSERT Measurement (Parameter "1.8.7 NOx-NO, NO2-, N2O5, HNO3") (Id TES7)))
ASSERT cross-registration of all measurements taken by the instrument
(degree-of-cross-registration instrument) (platform ?id)))
)
```

Code 23: Instrument capability rule for the EOS TES instrument

This rule asserts the seven measurements TES can take when it detects that a copy of this instrument has been manifested. The measurements are assigned ID#s from TES1 to TES7. These ID#s play a role in the explanation facility. In addition to asserting these seven measurements, this rule also asserts that these seven measurements are cross-registered because they are taken by the same instrument. This is also important, as some synergy rules only apply to cross-registered measurements. Finally, note that this rule simply asserts the measurement facts without filling in measurement attributes (e.g., spatial resolution). Indeed, attribute inheritance from instruments and missions to measurements is carried out by the attribute inheritance rules, which are executed a step later in the VASSAR methodology.

Additional information concerning the EOS instrument capability rules is provided in the appendix. The complete set of requirement satisfaction rules for the EOS case study can be found in http://web.mit.edu/dselva/www/RBES/EOS/. This information comes from individual instrument or mission publications (Abshire et al., 1998; Aumann et al., 2003; Aumann & Miller, 1995; R. A. Barnes & Holmes, 1993; Beer, Glavich, & Rider, 2001; Bianchini, Cortesi, Palchetti, & Pascale, 2004; Cheney, Miller, Agreen, Doyle, & Lillibridge, 1994; Diner et al., 1989; Franz, Werdell, & Meister, 2005; Gille et al., 2003; Hughes Space Communications, 1990; Imhof et al., 1995; Kerr, 1991; B. H. Lambrigtsen & Calheiros, 2003; Levelt et al., 2006; Long et al., 1990; McCormick, Chu, Zawodny, Mauldin III, & McMaster, 1991; Monaldo, Thompson, Pichel, & Clemente-Colón, 2004; Njoku, Jackson, & Lakshmi, 2003; Rider & McCleese, 1991; Rottman, 2000; Ruf, Keihm, & Janssen, 1995; Schoeberl et al., 2006; Schutz & Suite, 2001; Travis, 1747; Tsai, Spencer, Wu, Winn, & Kellogg, 2000; Way & Smith, 1991; Wielicki et al., 1996; Willson, 2001; T. N. Woods, 2000; Yamaguchi, Kahle, Tsu, Kawakami, & Pniel, 1998), and was partially verified by a pool of experts.

5.3 Results

5.3.1 Preliminaries

Before presenting the results of the three sub-problems (i.e., instrument selection, instrument packaging, and mission scheduling), intermediate results that are used in the three sub-problems are presented and discussed. In particular, single instrument scientific scores, and bilateral interactions between instruments both at the science and engineering levels (i.e. S-DSM and E-DSM) are introduced, as well as mission scores and the data continuity metric for the mission scheduling problem.

5.3.1.1 Instrument scores

The importance of instrument benefit and instrument cost for the instrument selection and packaging problems is obvious. As explained in Section 4.4.1, there are two ways of computing individual instrument scores: a) consider the instruments in isolation; b) consider the instruments with respect to a non-empty reference architecture. The simplest approach is to consider single instrument missions and evaluate their cost and benefit using VASSAR. Instrument scientific scores computed this way for the EOS case study are provided in Figure 37. Figure 37 would suggest that the SAR, MODIS, MODIS-T, MLS, and OMI, are the highest scoring instruments in terms of scientific benefit. However, this approach does not capture interactions between instruments. For example, the altimeter (ALT-SSALT) and the radiation budget instruments (CERES) achieve very small scores because they require additional instruments in order to fully satisfy several subobjectives.

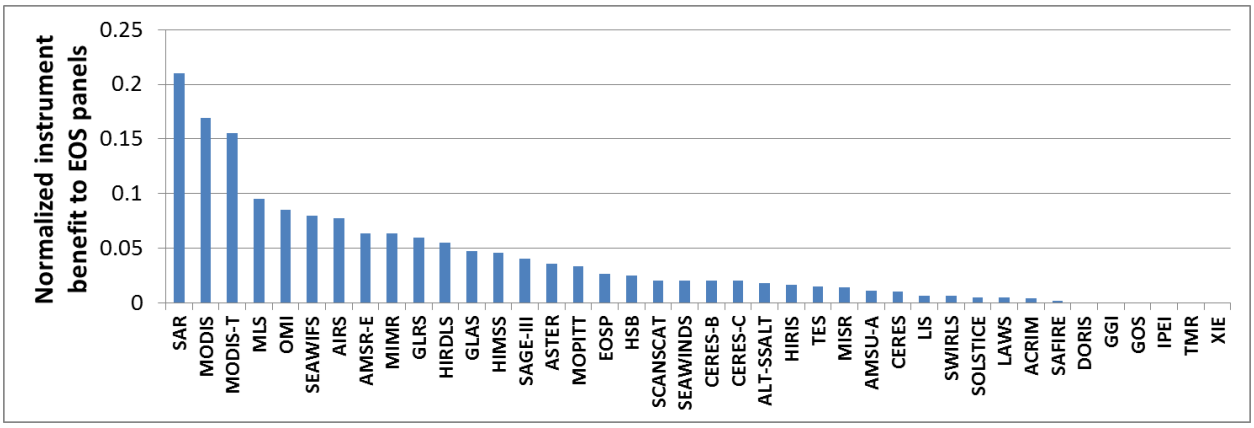


Figure 37: EOS instruments benefit scores in isolation

More precisely, the altimeter requires the TMR and at either GGI or DORIS for full satisfaction of the subobjectives concerning sea level height measurements. As for the CERES, several CERES with different scanning concepts on different orbits, and a cloud imager such as MODIS are required to fully satisfy Earth and cloud radiation subobjectives.

In this isolated approach, instrument costs are obtained by costing the single instrument missions. Note that this is a better option than just applying an instrument cost model, since: a) the cost of selecting an instrument is not only the cost of developing the instrument, but also the cost of developing the corresponding bus, integration, testing, launching, and operating the spacecraft; b) some instruments are international and therefore their development cost is zero; however, the cost of putting that instrument on a spacecraft is obviously non-zero.

Instrument cost-effectiveness in isolation can be computed by dividing the benefit scores provided in Figure 37 by the single instrument mission scores. The results of this calculation are shown in Figure 38. In a greedy approach to solve the instrument selection problem, one could start choosing instruments from Figure 38 in decreasing order of cost-effectiveness until there is no more money left.

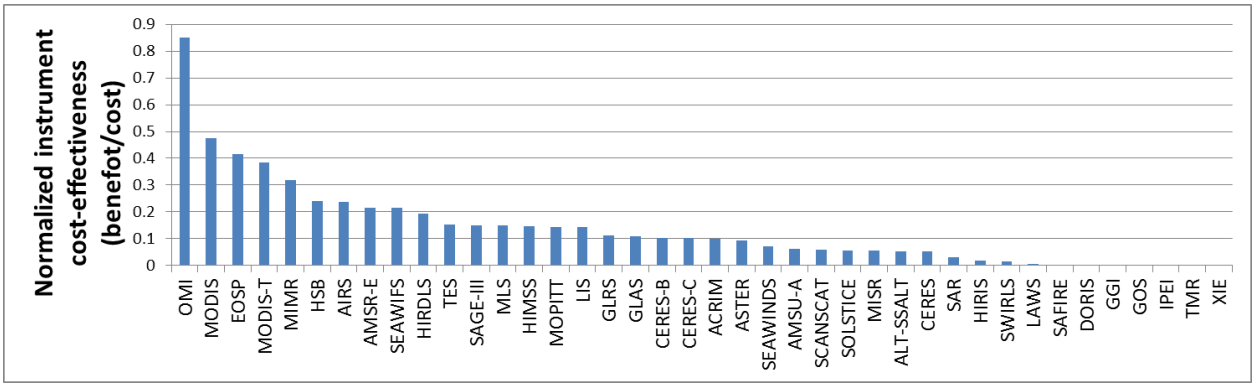


Figure 38: EOS instruments cost-effectiveness when considered in isolation

The second approach consists in the computation of marginal scores with respect to a reference architecture. In the EOS case study, a natural reference is the actual subset of instruments selected after all the descoping processes. This subset consists of 25 instruments. The marginal scores of these instruments with respect to the EOS reference architecture are provided on Figure 39.

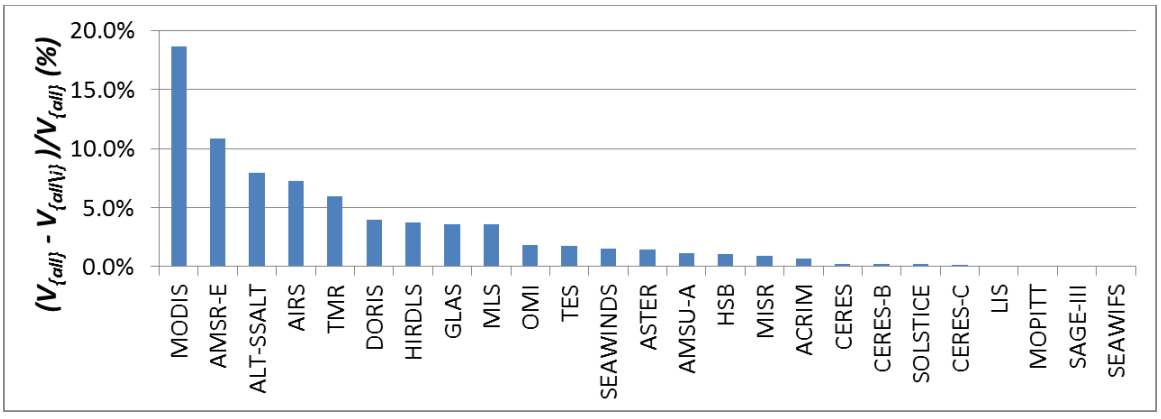


Figure 39: Marginal descoping scores for the EOS instruments w.r.t. the reference EOS architecture

5.3.1.2 S-DSM and E-DSM

After individual instrument scores, the second most important parameter for both the instrument selection and packaging problems are the bilateral interactions between the instruments. These interactions are captured by the B-DSM and the C-DSM defined in Section 9.1.2, which were relabeled as S-DSM and E-DSM for the case of EOSS.

A partial view of the S-DSM for the EOS instruments is shown in Table 22. In Table 22, each cell was computed as the difference of the score of a two-instrument mission minus the score obtained when the two single-instrument subobjectives are superimposed, i.e.:

$$\text{S-DSM}_{\{i,j\}} = V_{\{i,j\}} - V_i \cup V_j$$

where the operator $\cup$ computes a score by aggregating the maxima of the individual instrument subobjective satisfaction scores, and thus takes into account redundancies between instruments.

Table 22: S-DSM for the EOS case study

	AIRS	AMSR-E	AMSU-A	ASTER	CERES	CERES-B	CERES-C	HIRDLS	HSB	MISR	MLS	MODIS	MODIS-B	MOPITT	OMI	TES
AIRS	0.000	0.019	0.029	-0.031	0.000	0.000	0.000	0.000	0.010	0.011	0.018	0.012	0.012	0.000	0.023	0.019
AMSR-E	0.000	0.000	0.000	0.050	0.000	0.000	0.000	0.002	0.000	0.011	0.000	0.059	0.059	0.000	0.000	0.000
AMSU-A	0.000	0.000	0.000	-0.002	0.000	0.000	0.000	0.002	0.000	0.004	0.000	0.002	0.002	0.000	0.000	0.018
ASTER	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.020	0.000	0.000	-0.042	0.011	0.011	-0.026	-0.005	-0.015
CERES	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CERES-B	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CERES-C	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
HIRDLS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.033	0.033	0.009	0.014	0.040
HSB	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
MISR	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.020	0.020	0.000	0.013	0.000
MLS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.017	0.017	0.000	0.017	0.051
MODIS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000
MODIS-B	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000
MOPITT	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
OMI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.018
TES	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Therefore, positive values of S-DSM (green cells in Table 22) indicate positive synergies between instruments, and negative values (red cells in Table 22) indicate negative scientific synergies. Negative scientific synergies are possible due to design compromises in the presence of multiple instruments that lead to suboptimal science performance by one of the instruments. For example, instruments taking measurements in which diurnal sampling plays an important role in the error budget (e.g., altimeters) may be affected by being put together with other instruments because that a compromise in design may lead to selecting a suboptimal orbit (namely, an SSO).

The pictorial representation of this adjacency matrix is shown in Figure 40. Only the instruments considered for the packaging problem were included in Figure 40. We observe that the two MODIS, MISR, and the chemistry instruments are the most synergistic instruments when only bilateral interactions are considered. The major clusters in Figure 40 are fairly good indicators of potential missions: we have the sounders in the top (AIRS, AMSU-A, and HSB on the top), and a chemistry cluster on the bottom with HIRDLS, MOPITT, MLS, TES, MODIS, with ASTER and MISR in the middle.

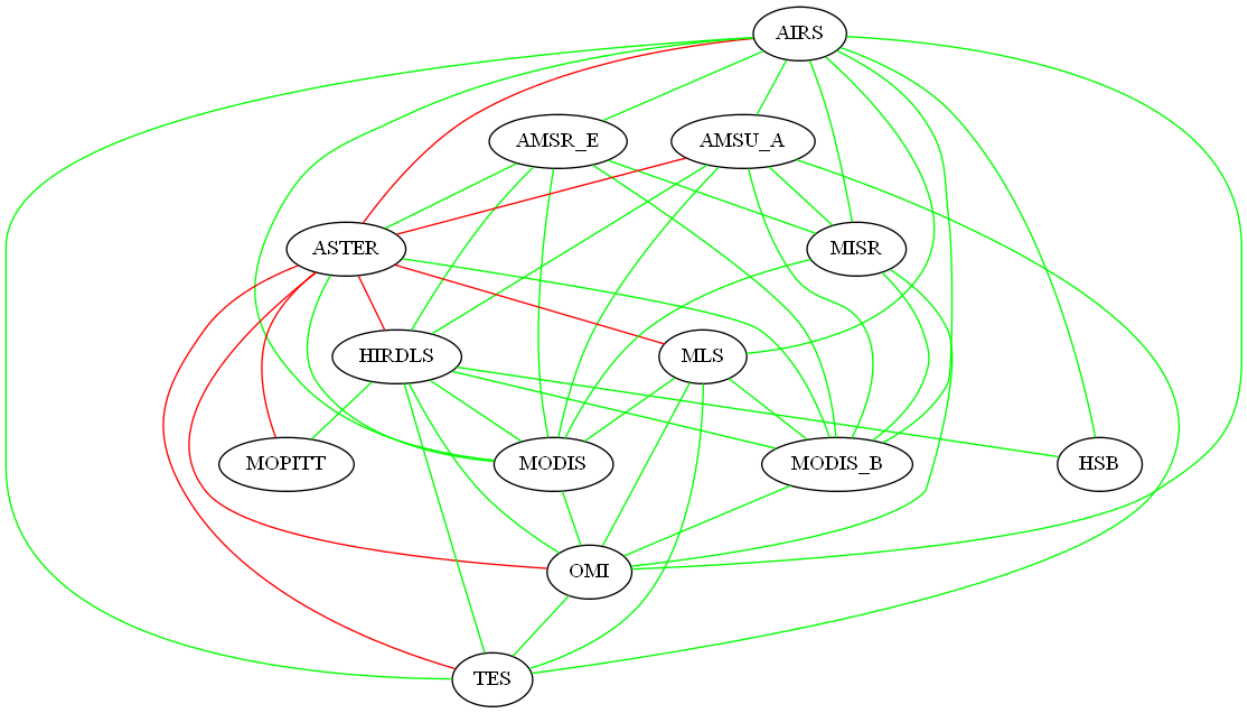


Figure 40: Pictorial representation of the S-DSM for the EOS case study

On the engineering side, a partial view of the E-DSM is shown in Table 23. In this case, each element of this matrix is computed as the difference between the cost of the two-instrument missions minus the sum of the costs of the two single-instrument missions (values are in FY00$M).

Table 23: E-DSM for the EOS case study (partial view)

	AIRS	AMSR-E	AMSU-A	ASTER	CERES	CERES-B	CERES-C	HIRDLS	HSB	MISR	MLS	MODIS	MODIS-B	MOPITT	OMI	TES
AIRS	0	-21	-32	-13	-10	-10	-10	10	-24	-8	-12	-8	-8	-3	-23	-26
AMSR-E	0	0	-24	-21	-32	-32	-32	13	-18	-29	-10	-28	-28	-16	-17	-19
AMSU-A	0	0	0	-22	-35	-35	-35	5	-24	-34	-28	-7	-7	-2	-23	-25
ASTER	0	0	0	0	-20	-20	-20	16	-15	-17	0	-14	-14	-3	-14	-17
CERES	0	0	0	0	0	-39	-39	1	-27	-38	-26	-13	-13	-7	-26	-28
CERES-B	0	0	0	0	0	0	-39	1	-27	-38	-26	-13	-13	-7	-26	-28
CERES-C	0	0	0	0	0	0	0	1	-27	-38	-26	-13	-13	-7	-26	-28
HIRDLS	0	0	0	0	0	0	0	0	-21	6	23	13	13	13	-20	-20
HSB	0	0	0	0	0	0	0	0	0	-25	-22	-26	-26	-23	-17	-18
MISR	0	0	0	0	0	0	0	0	0	0	-23	-12	-12	-5	-25	-26
MLS	0	0	0	0	0	0	0	0	0	0	0	-20	-20	11	-21	-24
MODIS	0	0	0	0	0	0	0	0	0	0	0	0	-14	-3	-25	-27
MODIS-B	0	0	0	0	0	0	0	0	0	0	0	0	0	-3	-25	-27
MOPITT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-22	-24
OMI	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-18
TES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hence, positive entries of this matrix represent negative interactions between instruments, while positive entries of this matrix represent potential savings by putting the instruments on a shared platform. Again, this matrix only captures bilateral interactions. The graphical representation of this adjacency matrix is provided in Figure 41.

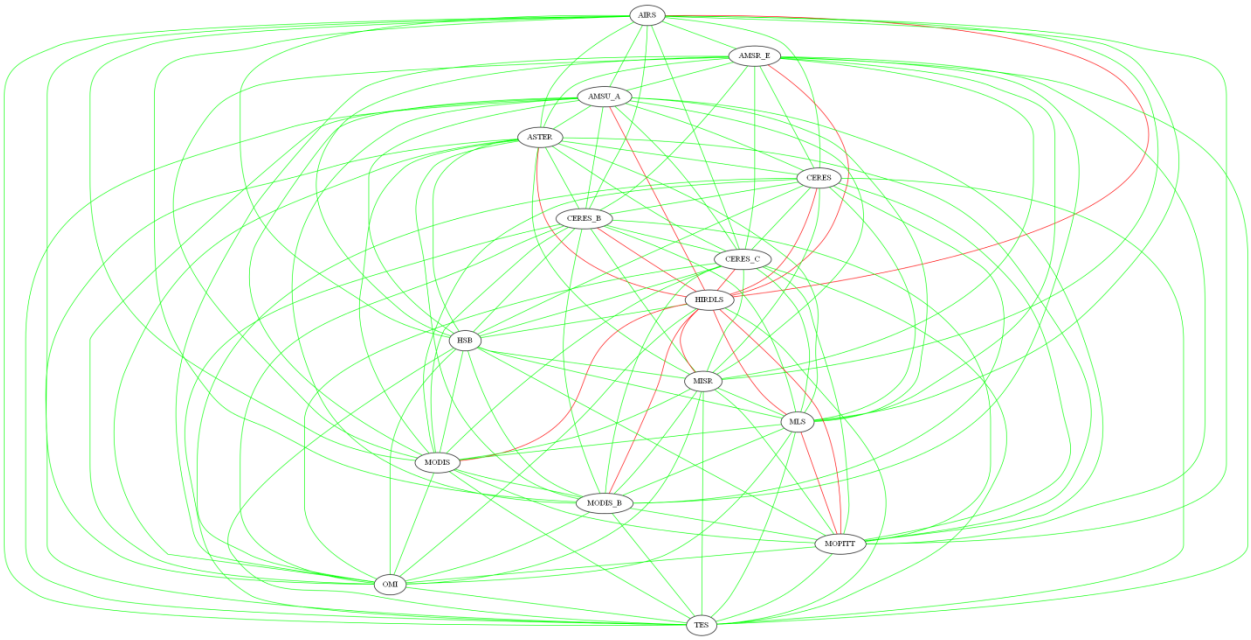


Figure 41: Pictorial representation of the E-DSM in the EOS case study (partial view).

It is noticeable from Figure 41 that most cost interactions between EOS instruments are positive. This means most of these instruments are good candidates to share a common bus. This makes sense because: a) they are all passive instruments with similar orbit requirements; b) most of them are relatively big instruments, which precludes the use of Pegastar-class buses and calls for the use of a larger multi-instrument platform.

5.3.1.3 Data continuity matrix

If bilateral interactions between instruments both in benefit and cost play major roles in the instrument selection and packaging problems, the data continuity matrix plays a role of equivalent importance in the mission scheduling problem. Introduced in Section 4.4.7, the data continuity matrix DCM is a multi-domain matrix where $DCM(m,t)$ represents the number of instruments that take measurement m during time interval t. The data continuity matrix for the EOS case study was computed by considering **all domestic and ESA missions (obviously except for EOS missions) between the years 1997 and 2014**. Only a reduced subset of measurements considered critical in terms of data continuity were considered. The matrix is shown in Figure 42. The color of each cell is proportional to the number of instruments taking that measurement at that moment in time (white = 0 instruments, black = 20 instruments).

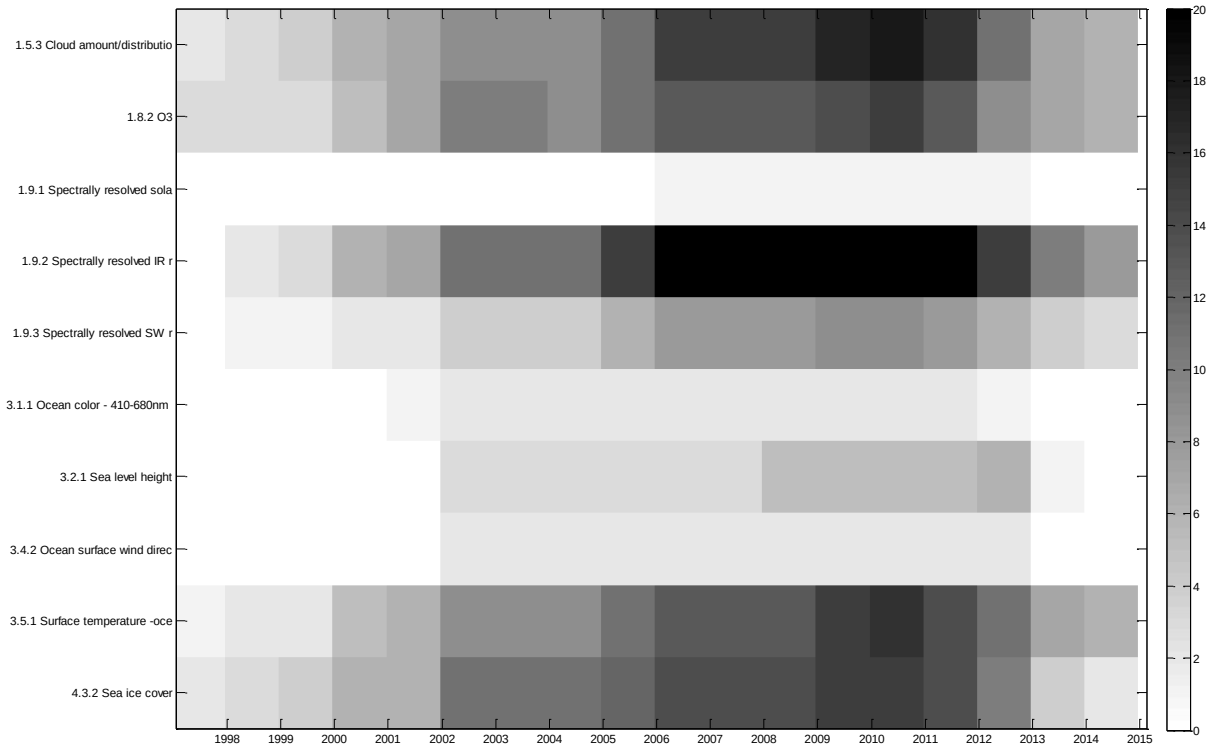


Figure 42: Data continuity matrix for the EOS case study

As shown in Figure 42, the major potential data gaps that EOS had to face were the following:

- Earth radiation budget (measurements 1.9.1 to 1.9.3 on Figure 42)
- Altimetry (measurement 3.2.1 on Figure 42)
- Scatterometry (measurement 3.4.2 on Figure 42)
- Ocean color (measurement 3.1.1on Figure 42)

Hence, the data continuity metric from the mission scheduling problem will favor architectures covering at least partially these data gaps. It may also be considered to add hard constraints concerning the launch dates of certain missions in order to close some of these data gaps. This will be discussed in more detail in section 5.3.4.

5.3.1.4 Mission scores and costs

In the scheduling problem, individual mission scores and costs are precomputed; they do not need to be recalculated at every architecture evaluation, because mission scores don't change, as nor the selection neither the packaging of the instruments change. An important assumption implicit in this statement is that the scheduling mode does not consider synergies across missions, or in other words, it considers that synergies across missions are negligible compared to synergies within missions. This is clearly an area for potential future improvement: the mission scheduling problem should take into account synergies across missions.

The missions considered for the EOS scheduling problem are described in Table 24.

Table 24: Lifetimes, payloads, and cost estimates for the missions in the EOS scheduling problem.

Mission	Lifetime (yrs)	Payload	Cost $M
ACRIMSAT	8	ACRIM	44
AQUA	6	AIRS AMSR-E AMSU-A CERES-C HSB MODIS	1,114
AURA	6	HIRDLS MLS OMI TES	968
ICESAT	3	GLAS	358
JASON-1	6	ALT-SSALT TMR GGI DORIS	264
METEOR-3M	4	SAGE-III	125
ORBVIEW	6	SEAWIFS	192
QUIKSCAT	8	SEAWINDS	285
SORCE	6	SOLSTICE	86
TERRA	6	ASTER CERES CERES-B MISR MODIS-B MOPITT	1,252

The cost estimates shown in Table 24 were computed using the model and do not represent actual mission costs. These costs take into account the particularities of the partners with international organization in each case, such as instruments developed, buses bought or spacecraft launched by international partners. Table 24 shows that TERRA, AQUA, and AURA are the most costly missions, which is no surprise.

These missions were evaluated in a preprocessing step and the mission scores shown in Figure 43 were obtained. Figure 43 ranks the missions in terms of scientific return, while Figure 44 ranks the mission in terms of cost-effectiveness.

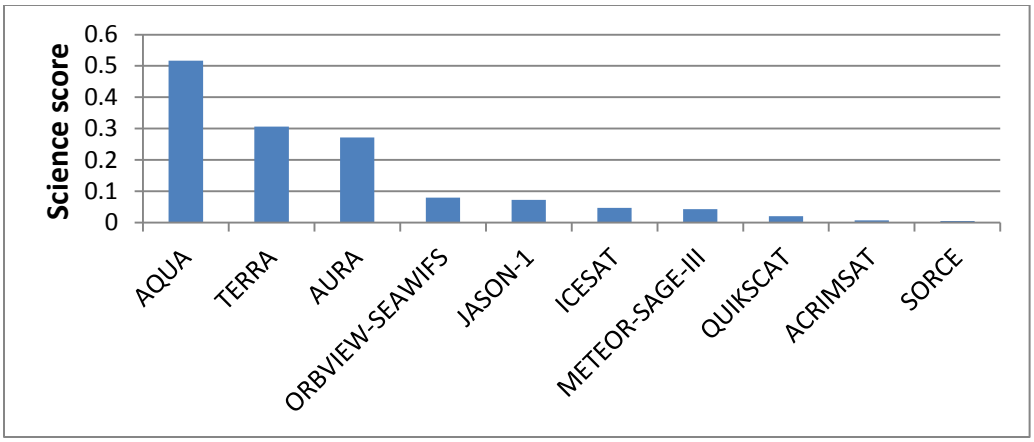


Figure 43: EOS missions for scheduling problem ranked by science scores

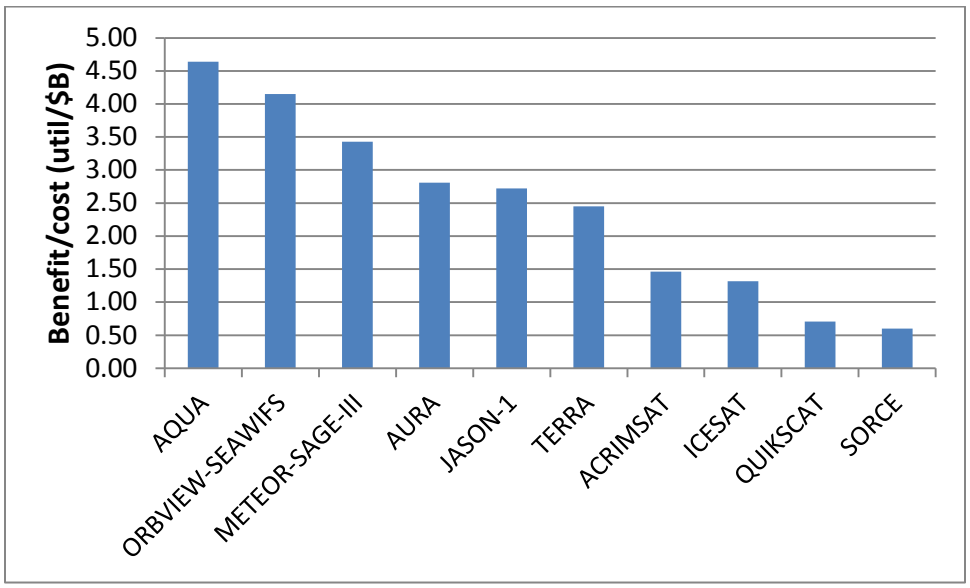


Figure 44: EOS missions for scheduling problems ranked by cost-effectiveness

The three top missions in science score coincide with the three top missions in cost, although AQUA achieves a higher score and a lower cost than TERRA. Concerning the cost-effectiveness ranking, AQUA is still the top mission, but the next two missions are SEAWIFS and SAGE-III, which are partnerships in which NASA provided the instrument and the partner provided the spacecraft, launch, and operations.

This type of partnership is very advantageous for NASA from the perspective of the model because it gets most of the value of the data products without having to pay for the full cost of the spacecraft development, fabrication, launch, and operations. Furthermore, in both cases, the instruments were actually continuity instruments that were very high TRL, and therefore the cost of the instrument was essentially fabrication cost.

5.3.2 Instrument selection

5.3.2.1 Configuration management: summary of rules used

The EOS instrument selection problem considers the 43 instruments shown in Figure 37. Therefore the size of the tradespace is $8.8 \cdot 10^{12}$ architectures. The reference architecture consists of the 25 instruments shown in Figure 39.

Before presenting and discussing the results of the EOS instrument selection case study, the different types of rules that were used to obtain these results are summarized. As explained in the theory chapters, domain-independent and SAP class-independent rules are provided with the generic algorithm. SAP class-specific rules are taken from the library of classes of SAPs, in particular from down-selecting problems. Domain-specific rules are added as needed. Some of the domain-specific rules are common to all EOSS (e.g., attribute inheritance rules and some emergence rules) and therefore are available in the expert system, some are specific to EOS, but used by the three SAPs, while some are specific to the EOS instrument selection problem. The results presented in this section were obtained with the set of rules provided in Table 25.

In addition to these rules, the enumeration constraints shown in Table 26 were added to the EOS instrument selection case study. XOR rules require that exactly one of the instruments is selected. GROUP constraints require that either none or all of the instruments are selected. These rules are not strictly necessary, as the algorithm is intelligent enough to locate these conflicts. However, they do accelerate the convergence of the search process.

Table 25: Summary of rules utilized in the EOS instrument selection case study

Class of rule	Type of knowledge	Source of rule
Grammar and enumeration rules	Domain-independent, common to all down-selecting problems	Library-DsPs
Search heuristics	Domain-independent, common to all down-selecting problems	Library-DsPs
VASSAR-aggregation rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-requirement satisfaction rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-instrument capability rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all down-selecting problems	See section 2.3.6

Table 26: Additional enumeration constraints added to the EOS instrument selection case study.

Type of constraint	Instruments concerned	Justification
XOR	SCANSCAT-SEAWINDS	These two instruments were essentially redundant and competing for selection
XOR	MLS-SAFIRE	These two instruments were essentially redundant and competing for selection
XOR	GLAS-GLRS	These two instruments were essentially redundant and competing for selection
GROUP	ALT-SSALT TMR	It doesn't make sense to consider the altimeter without the microwave radiometer and vice versa.

5.3.2.2 Results from the architectural tradespace exploration

The generic search algorithm complemented with the rules enumerated inTable 26 was initialized with a random generation containing the reference architecture, and the algorithm was run for 30 generations. The last generation obtained before down-selection is plotted in Figure 45. In Figure 45, each diamond represents one EOS selection architecture (i.e., a subset of instruments) in the cost-science space. Keep in mind that these are only two of the four dimensions of the architectural objective space in instrument selection problems, as programmatic risk and fairness are also relevant metrics. The color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process.

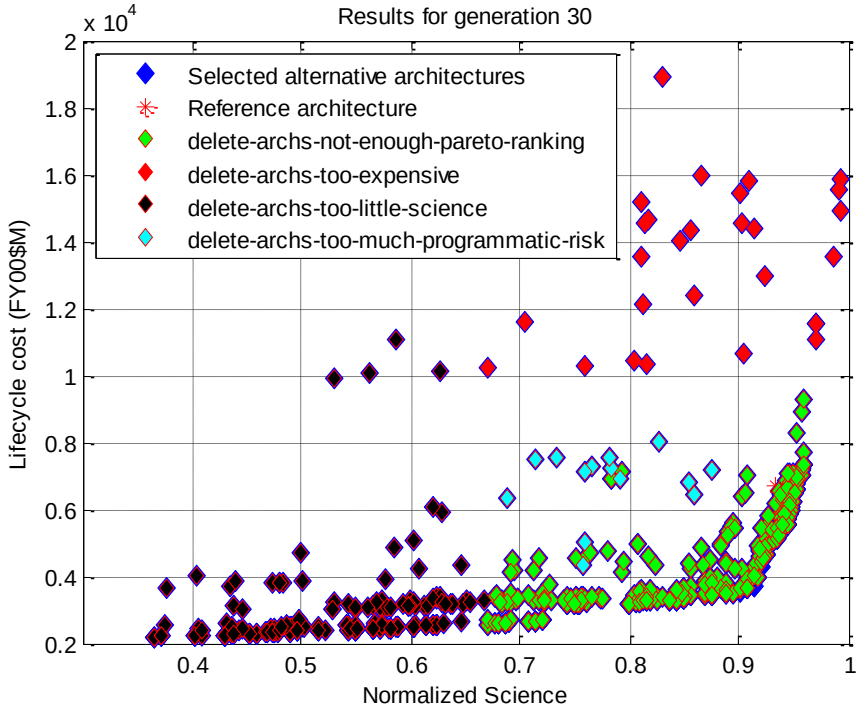


Figure 45: Population of EOS selection architectures after 30 generations in the science-cost space.

The down-selection rules that originated the elimination of the architectures as shown in Figure 45 are the following:

- Normalized Science (computed as described in Section 4.4.1) > 0.67
- Lifecycle cost (computed as described in Section 4.4.2) < 4 FY00$B
- Normalized programmatic risk (computed as described in Section 4.4.3) < 0.1
- Utility (computed as described in Section 2.3.6.2 with weights 40% science, 40% cost, 20% risk, 0% fairness) > 0.5
- Pareto ranking (computed as described in Section 2.3.6.1) < 4

Note that some architectures are eliminated because they have too high cost or too low science, a handful because of too high programmatic risk, and most architectures are eliminated because they are highly dominated (i.e., they have a high Pareto ranking, where higher ranking is worse). The remaining architectures after application of these down-selection rules are shown in Figure 46, together with the reference architecture and the utopia point. Four architectures are highlighted on the fuzzy Pareto frontier, and described in more detail in Table 27.

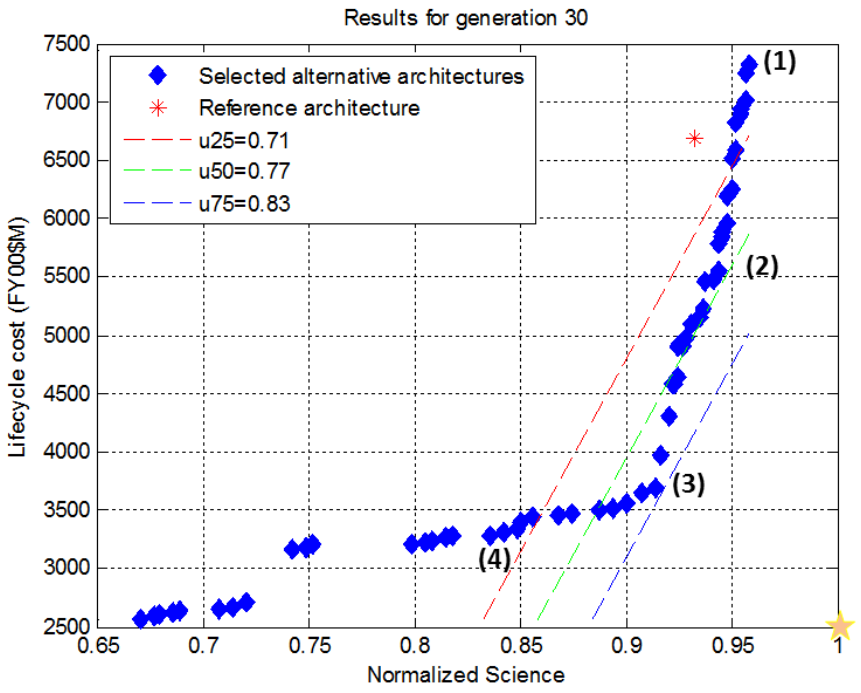


Figure 46: Fuzzy Pareto frontier of EOS selection architecture after 30 generations

While the reference architecture is close to the Pareto frontier, it is dominated, and several architectures obtain the same science for lower cost, or higher science at the same cost. Table 27 shows that **none of the selected alternative architectures carry the SAR**. A simple query of the expert system shows that architectures carrying the SAR were eliminated due to multiple reasons, but the root cause is an unfavorable cost-effectiveness at high risk. Hence, the model correctly deletes the SAR. The model also correctly selects MLS over SAFIRE, SEAWINDS over SCANSCAT, and GLAS over GLRS on most of the top architectures. However, there are some differences between the reference architecture and the top architectures on Table 27. In particular:

Table 27: Difference between the EOS reference selection architecture and the top alternative architectures

Arch. id#	Instruments added w.r.t. ref	Instruments deletedw.r.t. ref
1	MIMR + EOSP+LAWS + SWIRLS+ HIRIS	LIS + SEAWIFS + MOPITT + SAGE-III + AMSR-E
2	MIMR + EOSP+ LAWS + SWIRLS	LIS + SEAWIFS + MOPITT + SAGE-III + AMSR-E+ 2 x CERES
3	MIMR + EOSP	LIS + SEAWIFS + MOPITT + SAGE-III + AMSR-E+ 2 x CERES + ASTER + MISR + OMI
4	MIMR	LIS + SEAWIFS + MOPITT + SAGE-III + AMSR-E+ 2 x CERES + ASTER + MISR + OMI + CERES -C + HIRDLS + SOLSTICE + TES

- LIS is deleted because no objectives were associated to lightning. This problem was solved by incorporating a new objective in the weather panel that requires lightning[15].
- MOPITT is deleted because it is considered redundant with TES. Both instruments measure CO and CH4 and have good sensitivity in the lower troposphere, although they are very different instruments: MOPITT is a gas filter correlation radiometer and thus it has better coverage than TES but only looks at a very narrow band around the spectral features of interest, while TES is a Fourier transform spectrometer and therefore captures a much broader spectral region around the features of interest. In reality, other factors may have played an important role in the decision to select MOPITT, such as the fact that it was required to have a collaboration with the Canadian Space Agency.
- SEAWIFS is also deleted because, in this "static view" of the instrument selection problem, the instrument is considered redundant and inferior to MODIS. In reality, it will become clear when the scheduling problem is introduced that a second ocean color instrument was needed to be launched immediately in order to cover a potential data gap before the arrival of MODIS.

[15] The charts on Section 5.3.1.1 already take this objective into account, as it can be seen from the positive score obtained by LIS.

- SAGE-III is also deleted, in this case because it is considered redundant with HIRDLS and MLS. This is a similar case to SEAWIFS, although not identical. SAGE-III was selected due to couplings with the scheduling problem as well, but not only related to covering a potential data gap in aerosol data products. In this case, there was a requirement for the known heritage SAGE instrument to overlap with the new aerosol instrument (namely MODIS) in order to provide cross-calibration. This constraint appears explicitly in the scheduling problem.
- As a consequence of these instruments being deleted, the model has some money left with respect to the reference architecture that it chooses to invest primarily on the EOS polarimeter, and to a lesser degree also on other higher risk instruments such as the Doppler wind lidar LAWS and the other wind instrument SWIRLS.
- MIMR is selected over AMSR-E, because the two instruments offer comparable capabilities while MIMR appears to be less massive than AMSR-E. The model suggests that, again, factors that go beyond purely technical reasons may have affected this decision.

The results that have been presented are the "unconstrained" version of the results, where the model is left free to find a set of preferred architectures given the inputs. The differences between the reference architecture and these preferred architectures have been discussed. However, since the purpose of this case study is to validate that the tool is capable to produce useful results, **a second set of results was obtained with a few additional rules that try to capture the factors mentioned in the discussion above**. These rules are given in Table 28. FIX constraints require that the instruments are all selected.

Table 28: Additional rules added to the EOS instrument selection case study in order to model non-technical factors discussed above.

Type of constraint	Instruments concerned	Justification
FIX	ASTER HSB MOPITT AMSR-E	The selection of international instruments was driven by policy considerations and therefore needs to be forced as an externality. This rule was needed for AMSR-E and MOPITT, but ASTER and HSB were added for consistency.
FIX	SEAWIFS SAGE-III	These two instruments are continuity instruments required to cover potential data gaps and to provide cross-calibration with other similar instruments in the set.

With these rules incorporated, the model was run for 70 more generations. The results after these 70 extra generations are shown in Figure 47, and the remaining architectures after application of down-selection rules are shown in Figure 48, together with the reference architecture and the utopia point.

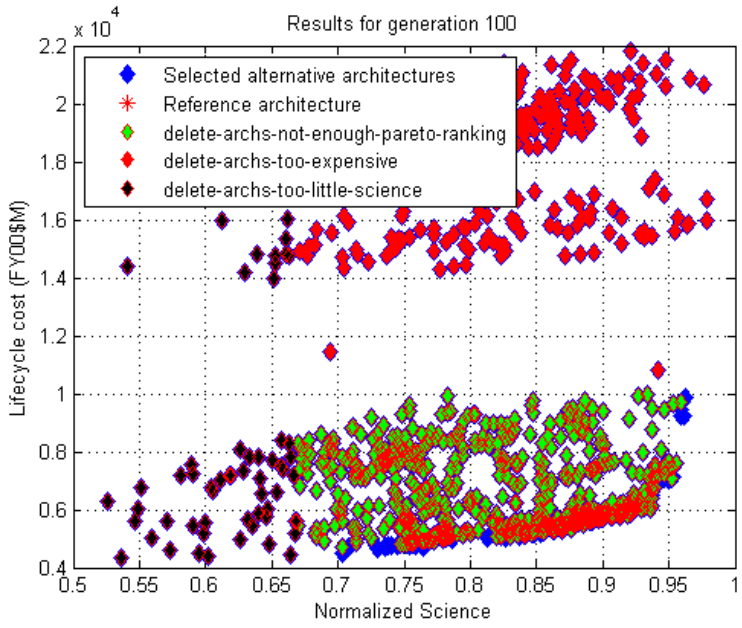


Figure 47: Population of EOS selection architectures after 100 generations in the science-cost space.

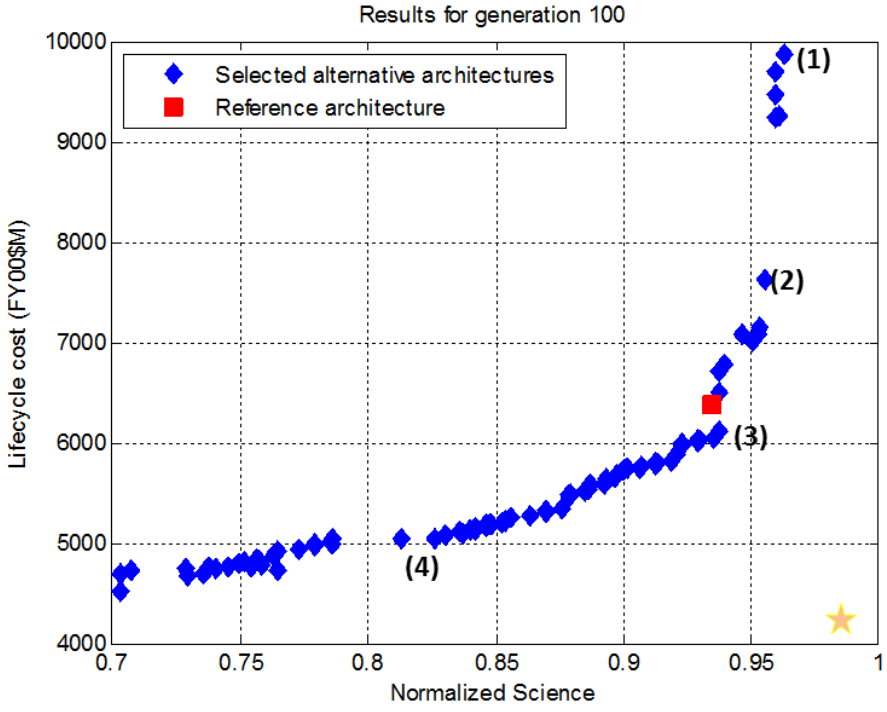


Figure 48: Fuzzy Pareto frontier of EOS selection architecture after 100 generations

It is apparent on Figure 48 that **the reference architecture not only lays on the fuzzy Pareto front, but it is also very close to the knee of the curve**, i.e., the point where there is a pronounced change in the marginal returns of investment. Alternative architecture #1 flies five more instruments than the reference architecture and flies the laser altimeter mission with the ranging option, but at a cost about 50% higher. Architecture #2 flies four more instruments than the reference architecture that enable it to satisfy valuable measurement requirements concerning atmospheric winds, cloud and aerosol properties, and ocean color, for a gain of about 3% in total science at a price of about 1.2$B.Four architectures are highlighted on the fuzzy Pareto frontier, and described in more detail Table 29.

Table 29: Difference between the EOS reference selection architecture and the top alternative architectures, with the additional rules defined in Table 28

Arch. id#	Instruments added w.r.t. ref	Instruments deletedw.r.t. ref
1	EOSP GLRS LAWS MIMR MODIS-T SWIRLS	GLAS
2	EOSP GLRS MIMR MODIS-T SWIRLS	GLAS
3	EOSP	2 x CERES
4	EOSP	ACRIM AMSU-A CERES HIRDLS LIS MISR SOLSTICE TES

Note that alternative architecture #3 on Table 29 is almost identical to the reference architecture. However, while Table 29 shows much fewer differences with respect to the reference architecture than the previous Table 27, there are still some differences that persist.

In particular, it is important to note that EOSP continues to appear on most non-dominated architectures. As a matter of fact, as budget is decreased from the $7B, architectures that delete other instruments while keeping EOSP tend to score better than architectures that delete EOSP. This is so because EOSP was a small, high performance instrument that was unique in its capabilities due to the multiple polarization measurements, which are very valuable to the cloud and aerosol communities, beyond the multi-angular measurements of MISR.

5.3.2.3 Discussion

At this point, it is necessary to remind that the goal of the EOS case study was to demonstrate the validity of the tool by trying to replicate the results of the real EOS case study. Overall, the results replicate what happened in reality in the EOS program, but only when the following rules were added into the analysis:

- **Force the selection of MOPITT**: MOPITT is a Canadian gas filter correlation radiometer, essentially redundant with TES in its capabilities of measuring CO and CH_4 with sensitivity in the lower troposphere. The motivations to fly MOPITT were driven by external needs (satisfaction of the needs of international partners). Hence, the selection of MOPITT was forced in this simulation in order to capture these policy considerations[16].
- **Force the selection of AMSR-E instead of MIMR:** AMSR-E and MIMR were multi-spectral MW imaging radiometers developed by the Japanese and ESA respectively, and essentially competing for the same spot in EOS. Interestingly, the algorithm systematically choses MIMR over AMSR-E, because they are essentially equivalent in terms of science and MIMR was, according to the reference available at the time, smaller and less massive and therefore considered as a less costly option. AMSR-E was selected instead of MIMR again because of non-technical considerations.
- **Force the selection of SAGE-III**: The algorithm detects that SAGE-III and other chemistry instruments are essentially redundant, with SAGE-III even being inferior in some aspects, mostly due to poor coverage because of the occultation measurement technique. SAGE-III was in reality a continuity instrument, whose goal was to provide cross-calibration to subsequent aerosol instruments. Thus, the selection of SAGE-III was forced as a hard constraint for this reason.
- **Force the selection of SEAWIFS**: SEAWIFS is an ocean color instrument that is redundant with, and inferior to MODIS in that respect. Again, the selection of SEAWIFS in the actual EOS obeyed data continuity considerations. This cannot be captured with soft constraints, and therefore the selection of SEAWIFS was forced in this simulation.

In summary, it is apparent that the tool successfully modeled the major trade-offs in the EOS instrument selection problem, once needs related to international partners, and couplings with the scheduling problem, were taken into account.

5.3.3 Instrument packaging

5.3.3.1 Configuration management: summary of rules used

The EOS packaging case study considers the 16 instruments that flew on the EOS/Terra, Aqua, and Aura spacecraft:

- TERRA: **ASTER, CERES, MISR, MODIS, CERES-B, MOPITT**
- AQUA: **AIRS, AMSU-A, HSB, AMSR-E, MODIS-B, CERES-C**

[16] Private conversation with Dr Christopher Scolese (NASA HQ)

- AURA: **HIRDLS, MLS, OMI, TES**

This implies a total of $Bell(16) = 10.4 \cdot 10^9$ architectures. The reference architecture is thus a 3-satellite architecture. This set of instruments is particularly challenging for several reasons:

- Many instruments are similar in characteristics (passive imagers) and orbit requirements (SSO, high altitude, AM or PM orbits)
- They are very synergistic instruments

For this case study, the following launch vehicles were considered.

- **Atlas 5** (EOS/Terra flew on an Atlas-5)
- **Delta-7920** (EOS/Aqua flew on a Delta-7920)
- **Delta-7420** (EOS/Calipso and EOS/Cloudsat flew on a Delta-7420)
- **Delta-7320** (EOS/Icesat flew on a Delta-7320)
- Taurus-3110 or **Taurus-XL** (available in 2004, EOS/OCO flew on a Taurus-3110)
- Taurus 2110 or **Taurus** (EOS/QUIKTOMS flew on aTaurus 2110)

The **Minotaur-IV is not considered** because it did not start service until 2010. European launchers were also discarded.

In terms of orbits, an initial run of the model for a few generations showed that only the **SSO-800km-AM** and **SSO-800km-PM** orbits need to be considered, as the model never assigned a different orbit to any spacecraft.

As it was done in the instrument selection case, before presenting and discussing the results of the EOS instrument packaging problem, the different types of rules used in the EOS instrument packaging problem that are not specific to this instance of SAP are reminded in Table 30.

In addition to these rules, the a few enumeration constraints were added to the EOS instrument packaging case study. Note that none of these rules are strictly necessary, as the algorithm is intelligent enough to locate these conflicts. The goal of these rules is rather to focus on the interesting regions of the tradespace, and thus accelerate the convergence of the search process. These additional enumeration constraints are shown in Table 31.

Table 30: Summary of rules utilized in the EOS instrument packaging case study

Class of rule	Type of knowledge	Source of rule
Grammar and enumeration rules	Domain-independent, common to all set	Library-PaPs

	partitioning problems	
Search heuristics	Domain-independent, common to all set partitioning problems	Library-PaPs
VASSAR-aggregation rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-requirement satisfaction rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-instrument capability rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Common to all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all partitioning problems	Library-PaPs

Table 31: Additional enumeration constraints added to the EOS instrument packaging case study

Type of constraint	Instruments concerned	Justification
APART	CERES CERES-C	It was a requirement of the radiation budget group that at least 2 CERES flew in different orbits in order to obtain a better diurnal sampling (Wielicki et al., 1996).
APART	MODIS MODIS-B	To achieve 12h temporal resolution in several key data products.
TOGETHER	AIRS AMSU-A HSB	The three sounders need to be together as opposed to in a train configuration in order to ensure high accuracy retrievals.

Also with the idea of steering the algorithm towards interesting regions of the tradespace, **architectures with more than six satellites, or more than ten instruments per satellite, were not considered**.

5.3.3.2 Results from the architectural tradespace exploration

The generic search algorithm complemented with the rules enumerated in Table 30 and Table 31 was initialized with a random generation containing the reference architecture, and the algorithm was run for 30 iterations. The last generation obtained before down-selection is plotted in Figure 49. In Figure 49, each diamond represents one EOS packaging architecture (i.e., a partition of instruments) in the cost-science space. The red square represents the reference architecture, and the yellow star represents the utopia point.

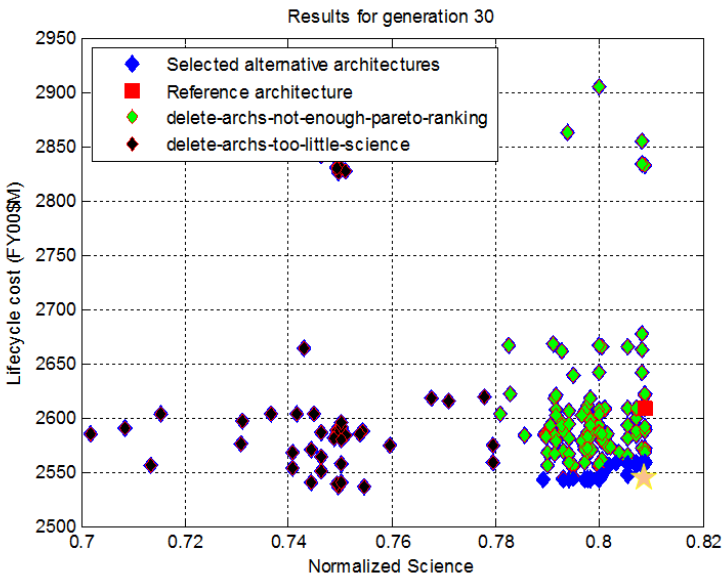


Figure 49: Population of EOS packaging architectures after 30 generations in the science-cost space

The color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process. The following down-selection rules were applied:

- Normalized Science > 0.78
- Lifecycle cost < 4 FY00$B
- Normalized programmatic risk < 0.4
- Normalized launch risk < 0.65
- Utility > 0.6
- Pareto ranking < 6

The remaining architectures after down-selection rules are applied are shown in Figure 50, together with the reference architecture and the utopia point.

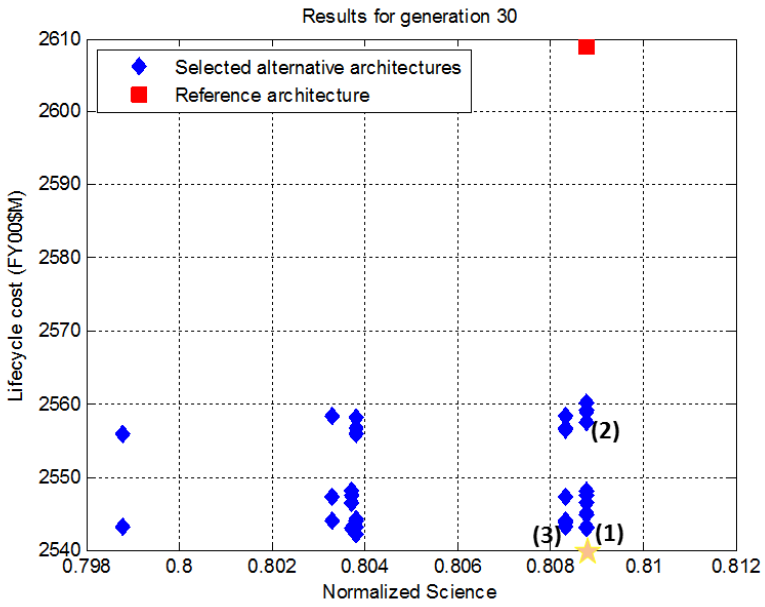


Figure 50: Fuzzy Pareto frontier of EOS packaging architecture after 30 generations

The first thing that we notice from Figure 50 is the appearance of vertical "clusters" of architectures. This structure appears in tradespace exploration when metrics have discrete codomains[17]. In this case, the codomain of the science metric is discrete mostly because: a) there is a finite number of synergies between instruments to be captured, and each synergy can either be captured or not; b) the function science(orbit) for each instrument typically only takes 2-4 values in the science codomain. The first reason is in part a result of a modeling assumption that makes it possible for trains of spacecraft to retain all of the scientific synergies. Hence for example, the model considers that in the reference architecture, all of the instruments from the AQUA and AURA spacecraft can be cross-registered to the level desired in to exploit synergies. Conversely, under this configuration, satellites flying on different orbits (e.g. AM vs PM orbit) do not allow cross-registration to a level that is enough to ensure all synergies.

The second cause is related to the fact that all architectures on the Pareto front consist of two types of orbits, both SSO at 800km: AM orbits, and PM orbits. As mentioned before, the model finds no motivation to fly any instruments at any orbit different than these two. This is really a particularity of this instrument set, and it is not a generality, as it will become clear for example in the Decadal survey case study. A broader variety of preferred instrument orbits would also potentially lead to the disappearance of the vertical clusters.

[17] The **codomain** of a function is the image of its domain, i.e. given $f: X \rightarrow Y$, X is the domain of f, and Y is its codomain.

One could argue that there is a third reason why this codomain is discrete, concerning the fact that there is a finite number of subobjectives that can be satisfied at a finite number of levels (full satisfaction, most value, some value, none). However, in practice, this fact does not justify the clusters because the number of subobjectives and options is too large for this effect to be visible.

Three alternative architectures were identified on Figure 50 at different points on the fuzzy Pareto frontier, and described in more detail in Table 32.

Arch	#sats	#instruments per satellite	Allocation
1	5	6-5-1-2-2	1 (PM): AIRS AMSR-E AMSU-A CERES-C HSB MODIS 2 (AM): ASTER CERES CERES-B MISR MODIS-B 3 (PM): OMI MOPITT 4 (PM): MLS TES 5 (PM): HIRDLS
2	4	7-5-2-2	1 (PM): AIRS AMSR-E AMSU-A CERES-C HSB MODIS OMI 2 (AM): ASTER CERES CERES-B MISR MODIS-B 3 (PM): HIRDLS MOPITT 4 (PM): MLS TES
3	5	6-6-2-2	1 (PM): AIRS AMSR-E AMSU-A HSB MODIS MOPITT 2 (AM): ASTER CERES CERES-B MISR MODIS-B OMI 4 (PM): MLS CERES-C 5 (PM): HIRDLS TES
ref	3	6-6-4	1 (PM): AIRS AMSR-E AMSU-A CERES-C HSB MODIS 2 (AM): ASTER CERES CERES-B MISR MODIS-B MOPITT 3 (PM): HIRDLS MLS OMI TES

Table 32: Difference between the EOS reference packaging architecture and the top alternative architectures

While the reference architecture is not on the science-cost Pareto front, Figure 49 and Figure 50 show that the reference architecture is close to the utopia point, since **it achieves the maximum scientific score** and **its cost is only about $65M (i.e., 2.0%) higher than the lowest cost.**

Note that this is only a projection of a 4D space on 2D, since launch and programmatic risk are other metrics of interest for packaging architectures. In fact, the reference architecture being a 3-satellite architecture, it scores poorly on these categories compared to more distributed architectures with four or five satellites, as it will be shown later in this section.

First of all, we use the explanation facility to understand why the reference architecture does not achieve the best score in terms of affordability. Explanation rules show that this is due to launch cost. This is better illustrated in Figure 51, which provides a pictorial representation of the reference architecture and the three alternative architectures. For each satellite, mass, orbit, lifecycle cost, and launch vehicle class are shown, in addition to the complexity penalties defined in 0. The legend for the complexity penalties is as follows: Green means inactive penalty, red means active penalty; M stands for mechanisms penalty, T stands for Thermal penalty, D stands for data rate penalty, A stands for ADCS penalty, S stands for scanning penalty, and E stands for EMC penalty.

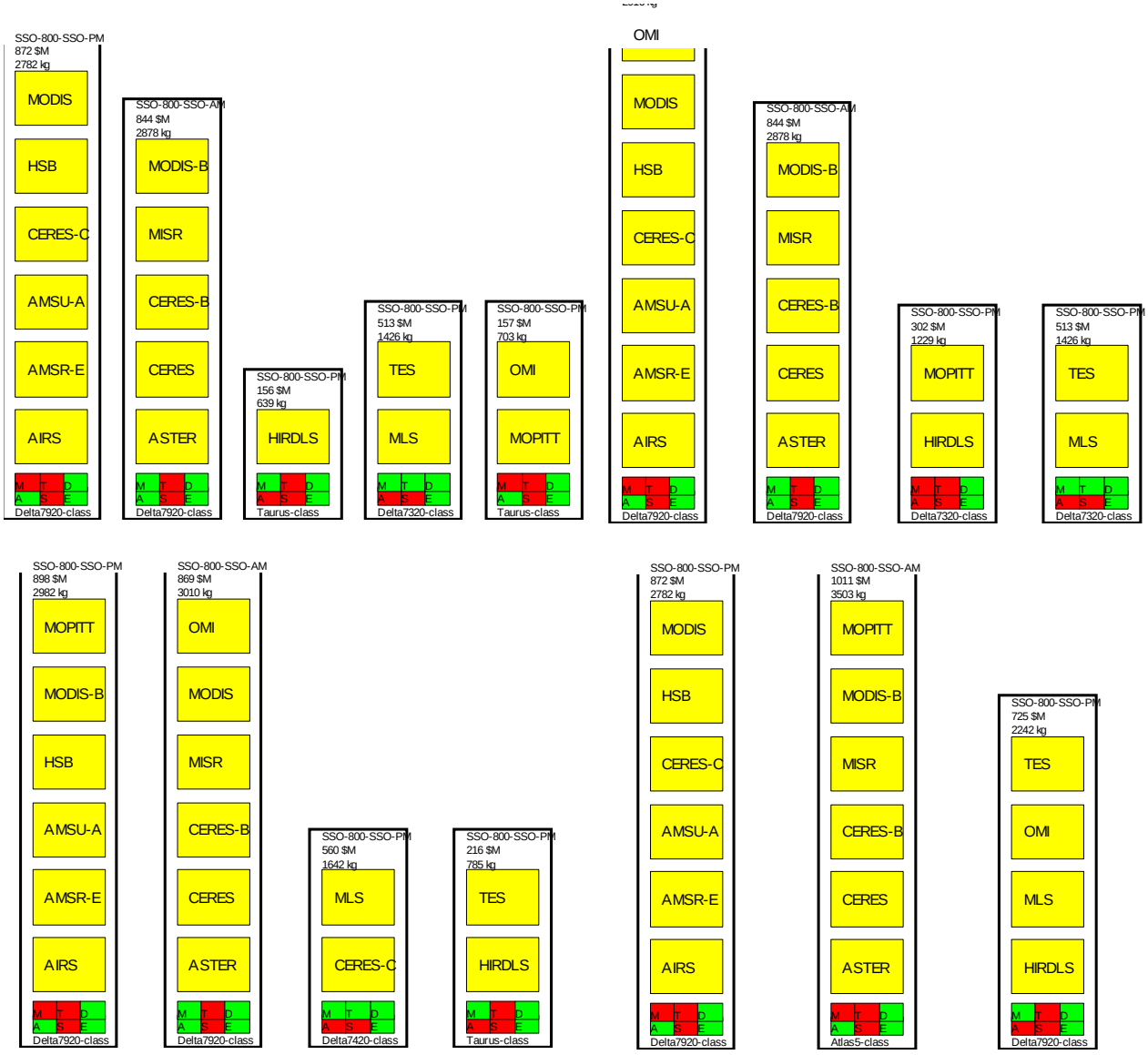


Figure 51: Detailed cost comparison between reference EOS packaging architecture (bottom right) and 3 alternative architectures

Figure 51 clearly shows how the **reference architecture requires an Atlas-5 to launch Terra, while no Atlas-5 launches are needed in any of the alternative architectures**. Launch costs for the reference architecture (1 Atlas-5 + 2 Delta-7920) are $240M. Launch costs for the alternative architectures are $215M (architecture 1), $220M (arch 2) and $205M (arch 3). The second most important cost difference resides in expected cost overrun. The model computes an expected cost overrun as a function of the expected schedule slippage, which in turn is a function of how instruments of different maturities are grouped together. All other components of the cost budget are very similar between all the architectures shown in Table 32, as it can be seen on Table 33.

Table 33: Cost comparison between EOS reference and alternative packaging architectures

Arch	Science	Payload cost	Bus cost	Launch cost	Program cost	IA&T cost	Ops cost	Expected overrun	Total cost
1	0.80878	484.8	739.0	215.0	362.7	280.8	252.6	208.2	2,543
2	0.80878	484.8	744.2	220.0	359.6	282.1	251.2	215.4	2,557
3	0.808348	484.8	748.8	205.0	359.3	283.1	251.5	210.7	2,543
ref	**0.80878**	484.8	757.8	240.0	353.5	286.6	249.0	**237.3**	**2,608**

It was mentioned earlier that the reference architecture is a 3-satellite architecture while alternative architectures have four or five satellites, which leads to improved cost, launch risk, and programmatic risk metrics. Distributing the 16 instruments over a larger number of satellites is more desirable for any risk-averse decision maker that wants to maximize the utility associated to the number of instruments successfully put into orbit. An interesting follow-up question is thus, how do more distributed architectures compare in terms of science and cost to the reference architecture? In order to answer this question, Figure 49 was redrawn with the colors of the diamonds now representing the number of satellites in the architecture. The resulting chart is provided in Figure 52.

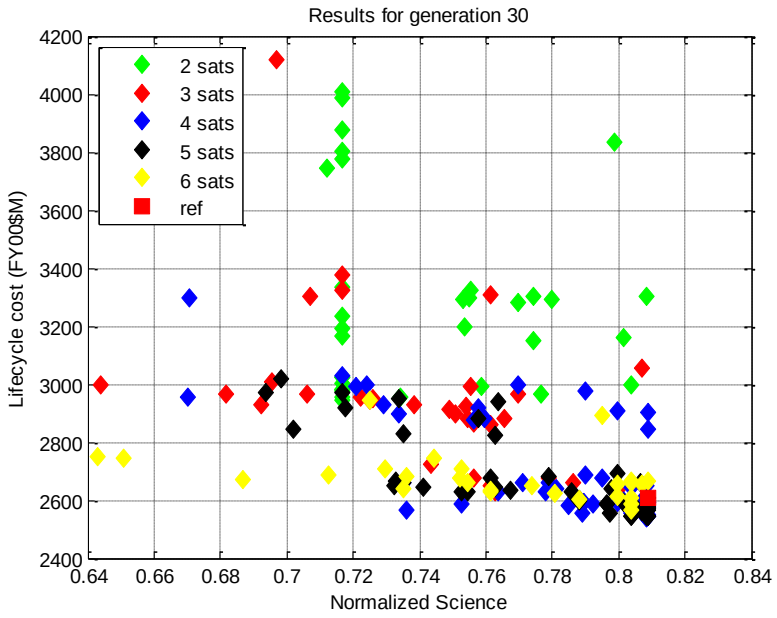


Figure 52: Packaging architectures in the cost-science space, colored according to number of satellites

Several comments can be drawn from Figure 52:

- **Architectures with 2, 3, 4, 4, and 6 satellites are capable of achieving the highest science score**. They do that by distributing the satellites in two trains: an AM train, and a PM train.
- **The least costly architectures are 4-, and 5- satellite architectures**. This is so because more distributed architectures can use the Taurus launch vehicle which is considerably cheaper than the Delta-2. This result is of course sensitive to the assumptions made in terms of launch vehicles and standard/dedicated buses.
- **The most costly architectures are generally 2-satellite architectures**. 2-satellite architectures absolutely require the presence of at least one Atlas-5.
- **The reference architecture is the least costly 3-satellite architecture**. Other 3 satellite architectures are not so compensated and achieve worse launch packaging factors.

The clustered structure of the trade space observed on Figure 49 suggests an isoperformance approach (O. L. de Weck & Jones, 2006), where all the architectures that achieve the maximum science score are identified, and then the cost-risk Pareto front is plot. This computation was done and the results are shown on Figure 53, where lifecycle cost and an average risk metric constructed as the average of programmatic risk and launch risk are shown, only for the subset of architectures that achieve the maximum science score.

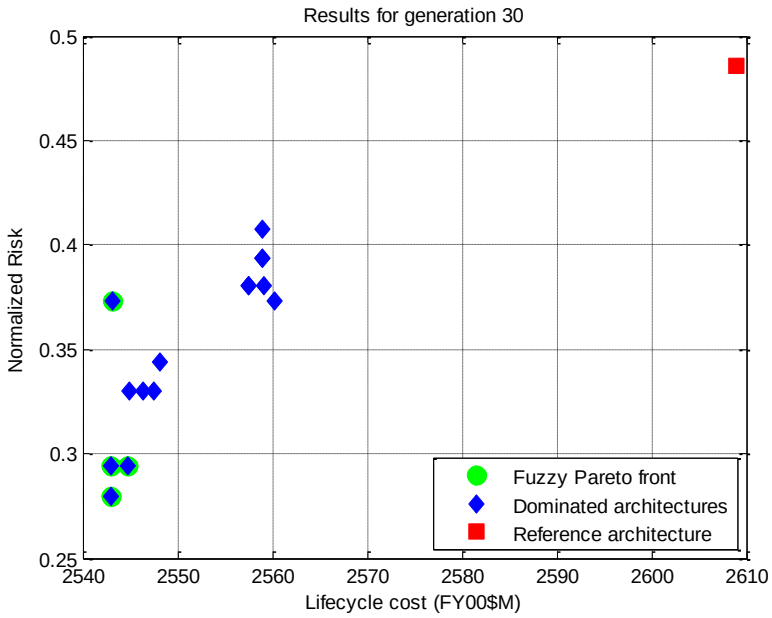


Figure 53: Cost-risk tradespace and Pareto frontier for architectures achieving the maximum science score

The best architecture on Figure 53 is also the best architecture on the science-cost space (alternative architecture #1 on Table 32). The other architectures on the fuzzy Pareto frontier of the cost-risk space are also 5-satellite architectures, where the instruments in the AURA spacecraft are distributed into several satellites.

All these results were obtained by assuming that a dedicated bus was developed for each satellite. As explained in Section 4.2.2.4, the RBES has the capability to allow for the use of standard buses instead of dedicated buses. Table 34 shows the difference in cost for the reference and alternative architectures when standard buses and dedicated buses are considered. This table only considers the T-330, the BCP-2000 and the Pegastar buses.

Table 34: Difference in lifecycle cost for reference and alternative EOS packaging architectures with standard buses vs dedicated buses

Arch#	Lifecycle cost (dedicated bus)	Lifecycle cost (standard bus)
1	2,543	2,285
2	2,557	2,285
3	2,543	2,019
ref	**2,608**	**2,003**

It is very interesting to see that **when standard buses are included, the reference architecture does not only achieve the highest science score, but it is also the most cost-effective way of doing so.** This is so because the bus packaging factors for the reference architecture are better than for the alternative architectures. The follow-up question is then whether there are other architectures that also obtain the same science and cost scores than the reference.

In order to answer this question, the tradespace search algorithm was run again, this time forcing the selection of standard buses for all spacecraft. The results of this new simulation are shown on Figure 54.

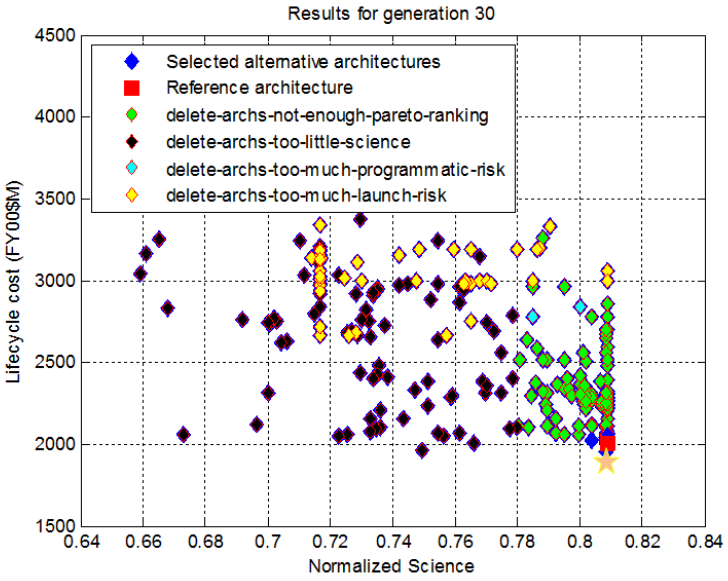


Figure 54: Population in last generation for the EOS packaging problem when standard buses are used

The set of down-selected architectures is shown on Figure 55. It is thus apparent that **the reference architecture is the best possible architecture on the science-cost space**. Other architectures achieve the same science and cost. These architectures are variations of the reference architecture where MOPITT is on one of the PM spacecraft.

Note that when standard buses are used the AURA spacecraft is not broken into smaller spacecraft because the MLS instrument by itself is 400kg and therefore requires a T-330 bus (the assumption is that the BCP-2000 can fit up to 300kg of payload). Hence, if this T-330 bus is necessary for MLS in any case, it is more interesting to fill it to its capacity (1300kg of payload) by adding the other instruments.

While science and cost are only 2 out of the 4 metrics considered (there are also launch and programmatic risk), and as explained before the reference architecture scores poorly in risk due to the lower number of satellites, any rational decision maker would agree to put science and cost as more important than launch and programmatic risk, given the definitions of these metrics described in Section 4.4.

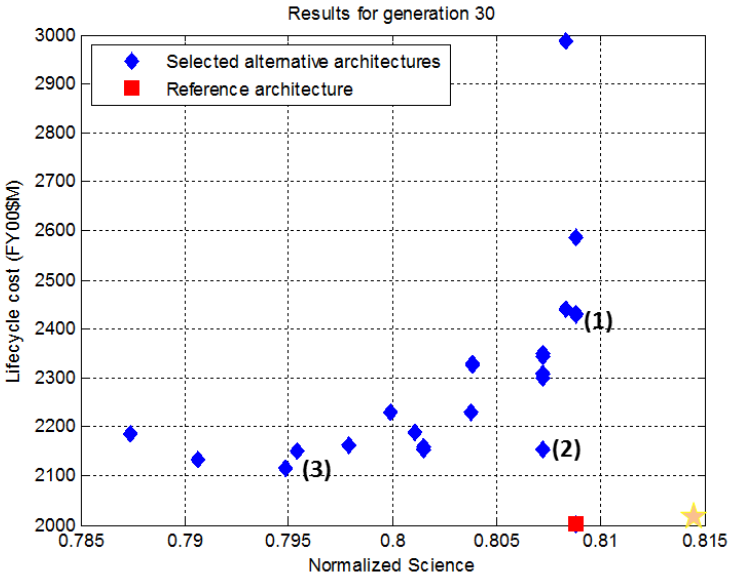


Figure 55: Down-selected architectures in the EOS packaging problem when standard buses are used

5.3.3.3 Discussion

It is apparent from the analysis that the instrument packaging problem is harder to solve than the instrument selection problem. This is so not only due to a larger tradespace size as shown in Section 2.2, but also because the level of modeling fidelity that is required is higher. In other words, it is a more knowledge-intensive problem than the instrument selection problem. Variations in scientific benefit across the whole spectrum of packaging architectures is one order of magnitude lower than the variation across the spectrum of selection architectures, because the "bulk" of the benefit is set with the instrument selection, and variations in the packaging problem come from synergies between instruments, and design compromises that affect scientific output (e.g., orbit selection). Therefore, a higher fidelity is required in order for the model to successfully capture the main architectural trade-offs.

The structure of the cost-science tradespace is also different to the one for the instrument selection problem. While the instrument selection trade space clearly showed a convex frontier on a wide range of science scores and costs, the trade space for the packaging problem is structured in clusters, where architectures within one cluster have the same science scores because they capture the same synergies.

This suggests several things: a) that the packaging problem is more a search problem than it is an optimization problem, in the classical meaning of the terms "search" and "optimization", the difference being on the difficulty of finding an acceptable architecture (as opposed to a good architecture), which is far greater in the packaging problem; b) that science and cost may not be competing metrics in the packaging problem. Indeed, it is not straightforward that architectures with high science are more costly, since science essentially is a matter of capturing all the right science synergies, and number of satellites cost is a poor indicator of lifecycle cost due to quantized launch and standard bus costs; c) that an iso-performance strategy can be utilized, where first all architectures that satisfy the maximum scientific score are identified, and then the optimization occurs in the cost-risk space.

In terms of validation, overall, the results of the EOS instrument packaging problem replicate what happened in reality in the EOS program, if:

- **The launch vehicles available and considered at the time are taken into account**: Since differences in cost across good packaging architectures are relatively small, quantized items such as launch cost play an important role. Launch cost is "quantized" because it can only take values from a small discrete set of possibilities. Hence, a small change in an architecture, such as moving an instrument from one satellite to another one, may lead to a big change in cost due to a change in launch vehicle.
- **The standard buses available and considered at the time are taken into account.** As described in Chapter 4, the tool allows two different approaches to satellite sizing: the design from scratch of a dedicated bus, tailored to the needs of its payload; b) the reuse of an available "standard" bus. The latter option is less costly in general whenever good packaging factors can be achieved, as development costs are very much reduced. A review of the literature and conversations with experts revealed that there were essentially three standard buses considered at the time of EOS: the T-330, a.k.a. as the EOS PM common spacecraft, for payloads about 1mt; the BCP-2000, for payloads around 300kg; and the Pegastar, for payloads below 70kg. Again, the choice of these standard buses is important and another quantized item in lifecycle cost, and thus adding other options would have led to different results. This is seen for example with MLS, which is too constraining to be flown on a BCP-2000, and thus requires a T-330. If an intermediate bus between the BCP-2000 and the T-330 was added, this result would certainly change.

- **Cross-registration between instruments in a train configuration is considered sufficient to capture all relevant synergies.** This assumption implies that a two-instrument satellite and a train of two single-instrument satellites are essentially identical from the science perspective. This assumption neglects the costs of cross-registering the data sets of the instruments, which could be comparably higher in the case of trains. This assumption is extremely important in determining scientific scores of most architectures, and in fact, results radically change if synergies between trains are not allowed. Simulations without intra-train synergies led to architecture classes driven by the placement of MODIS with either AQUA or AURA: architectures with a copy of MODIS in AQUA lose some synergies between MODIS and the chemistry instruments (e.g., aerosol and cloud properties), while architectures carrying that same copy on AURA instead miss important synergies between MODIS and the weather instruments (essentially cloud mask, cloud type, and aerosols).

While the reference architecture is the best architecture under these assumptions if only science and cost are considered, other good architectures identified by the model **do not fly MOPITT on the AM satellite, but on the PM train, with the other chemistry instruments.** The reason why MOPITT was flown in TERRA is a coupling between the scheduling and the packaging problems. There was a **requirement by the Canadians to fly MOPITT on the first EOS satellite regardless of the rest of the payload, and that was the AM satellite**.

As a final note, the model also identified a class of alternative architectures that are potentially of interest. In this class of architectures a PM spacecraft is broken down into 2 or 3 smaller spacecraft, all flying in a PM train. The chemistry instruments appear to be better options for distribution over several satellites than the AQUA suite. It is important to note that **this class of architectures is only enabled if dedicated buses are used, or if a mid-size bus between the BCP-2000 and the T-330 is available**. If this is not the case, the sounders on one side, and the MLS instrument on the other side, already require two T-330 buses, which then can be filled out to capacity by adding the rest of the AQUA/AURA instruments.

5.3.4 Mission scheduling

5.3.4.1 Configuration management: summary of rules used

The missions considered for the EOS scheduling problem are the ones presented in Figure 43. The data continuity matrix is the one shown in Figure 42. As it was done in the instrument selection and packaging cases, before presenting and discussing the results of the EOS mission scheduling problem, we remind the different types of rules that were used to obtain these results. All the results presented in this section were obtained with the following set of rules:

Class of rule	Type of knowledge	Source of rule
Grammar and enumeration rules	Domain-independent, common to all set permuting problems	Library-PePs
Search heuristics	Domain-independent, common to all set permuting problems	Library-PePs
VASSAR-aggregation rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-requirement satisfaction rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-instrument capability rules	EOS-specific, common to selection, packaging, and scheduling	See Section 5.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Common to all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all set permuting problems	Library-PePs

Table 35: Summary of rules utilized in the EOS mission scheduling case study

In addition to these rules, a few enumeration constraints were added to the EOS instrument selection case study. These enumeration constraints are provided in Table 36. Note that most of these rules are not are strictly necessary, as the algorithm is intelligent enough to locate these conflicts. The goal of these rules is rather to focus on the interesting regions of the tradespace, and thus accelerate the convergence of the search process.

Missions concerned	Type of constraint	Dates concerned	Justification
JASON-1	between-dates	1997-2002	to close gap from TOPEX-POSEIDON
TERRA	Before-date	2002	To overlap with Landsat-7 ETM+
AQUA	after-date	2001	for overlap with A train
AURA	between-dates	2002-2008	for overlap with A train
METEOR	between-dates	1999-2003	constrained by international partner
ORBVIEW	before-date	2000	constrained by international partner
SORCE	after-date	2000	Because of TRL considerations
ACRIMSAT	before-date	2001	to close gap in radiation budget

Table 36: Additional enumeration constraints added to the EOS mission scheduling case study

While this may seem like a very restraining set of constraints, in reality there are thousands of architectures that satisfy them. Finally, in terms of budget, an annual budget of $500M/year was assumed for every year between 1997 and 2007.

5.3.4.2 Results from the architectural tradespace exploration

The generic search algorithm complemented with the rules enumerated in Table 46 was initialized with a random population containing the reference architecture, and the algorithm was run for 50 iterations. The last generation obtained before down-selection is plotted in Figure 56.

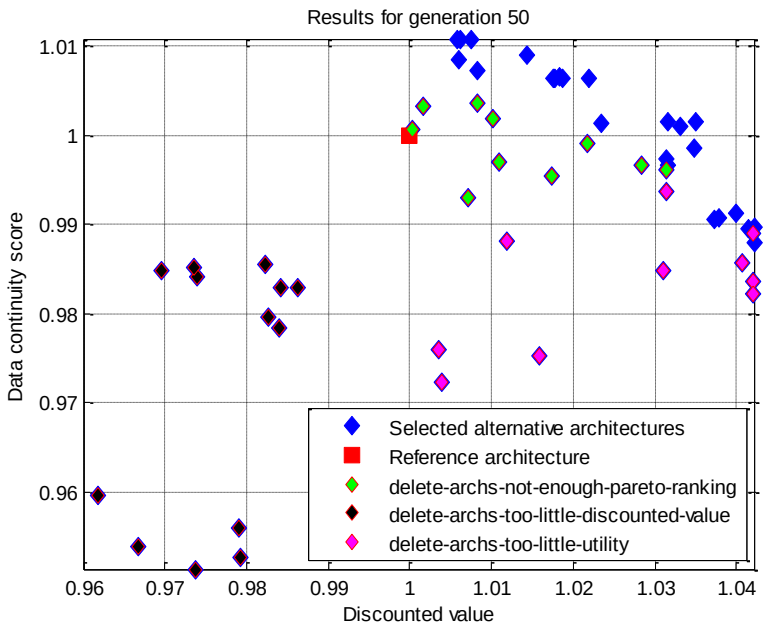


Figure 56: Population of EOS scheduling architectures after 100 generations in the discounted benefit-data continuity space.

In Figure 56, each diamond represents one EOS scheduling architecture (i.e., a permutation of missions) in the discounted benefit – data continuity space. Again, the color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process. The following down-selection rules were applied:

- Normalized data continuity > 0.95 (normalized w.r.t. the reference architecture)
- Normalized discounted benefit > 0.99 (normalized w.r.t. the reference architecture)
- Utility > 0.65
- Pareto ranking < 4

Figure 56 reveals the structure of the discounted value-data continuity tradespace. The last generation appears as a cloud of points that loosely ressembles an ellipse whose semimajor axis forms an angle of about 45degress with the horizontal axis. This suggests that the two metrics are positively correlated as opposed to being antagonist. Indeed, in the case of the NASA EOS, it appears that launching the most valuable missions first also tends to cover the most data gaps. This is not the case in general, as less valuable missions could cover very important data gaps.

The most important data gaps for the EOS case study were presented in Figure 42, and they are listed again here as a reminder: Earth radiation budget (incoming solar irradiance, outgoing SW and LW radiance), ocean color, sea level height, and sea surface winds. The ocean color gap can be covered by SEAWIFS or any of the two MODIS (AQUA, TERRA). The radiation budget gaps are covered by ACRIMSAT (solar irradiance) and the CERES instruments (AQUA, TERRA). The sea level height can only be covered by the altimetry mission, and the sea surface wind gap can only be covered by the scatterometer.

Mission scores and cost-effectiveness were shown in Figure 43 and Figure 44 respectively. TERRA, AQUA, and AURA were the top three missions in scientific score. The most cost-effective missions were AQUA, SAGE-III, and SEAWIFS, the latter providing relatively high value at very reduced cost through reflight of a very mature instrument in a partnership with external partners taking care of part of the costs.

The remaining architectures after down-selection are shown in Figure 57, together with the reference architecture and the utopia point.

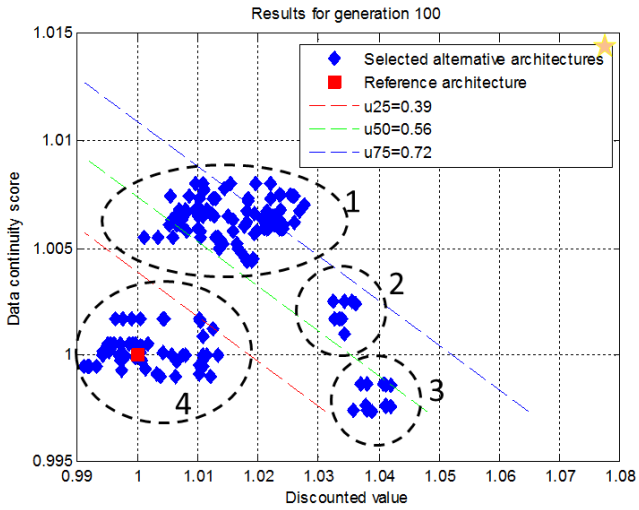


Figure 57: Fuzzy Pareto frontier of EOS scheduling architecture after 100 generations

Figure 57 shows the structure of the discounted value-data continuity tradespace. In this case, we observe four clouds of points that may potentially identify different classes of architectures. The launch dates of the architectures in each of these four clusters are shown in Figure 58 in order to analyze the differences between these four clusters in the architectural domain - as opposed to the metrics domain.

The boxplots Figure 58 show several differences between the four clusters. Launch dates in cluster 4 (bottom right), to which the reference architecture belongs, are naturally the closest to the reference architecture launch dates. AQUA is systematically launched earlier in cluster 2 (top right) and 3 (bottom left) than in clusters 1 (top left) and 4, which leads to a net gain in discounted value, since AQUA is the highest value mission. A similar argument applies to AURA: all clusters except cluster 4 launch AURA earlier, leading to benefits both in discounted value and data continuity. Launching AURA earlier leads to a delay in ICESAT. Note that no continuity requirement is identified for ice sheet topography measurements in the EOS case study. The requirement appeared after ICESAT, as a strong desire to continue the laser-based measurements started by ICESAT with ICESAT-II. Clusters 2 and 3 fly SAGE-III earlier, and QUIKSCAT later, which leads to higher discounted value at the price of some data continuity, as it takes longer to cover the scatterometry data gap. Finally, SORCE is launched much earlier in cluster 1 in order to cover the important data gap for Earth radiation budget.

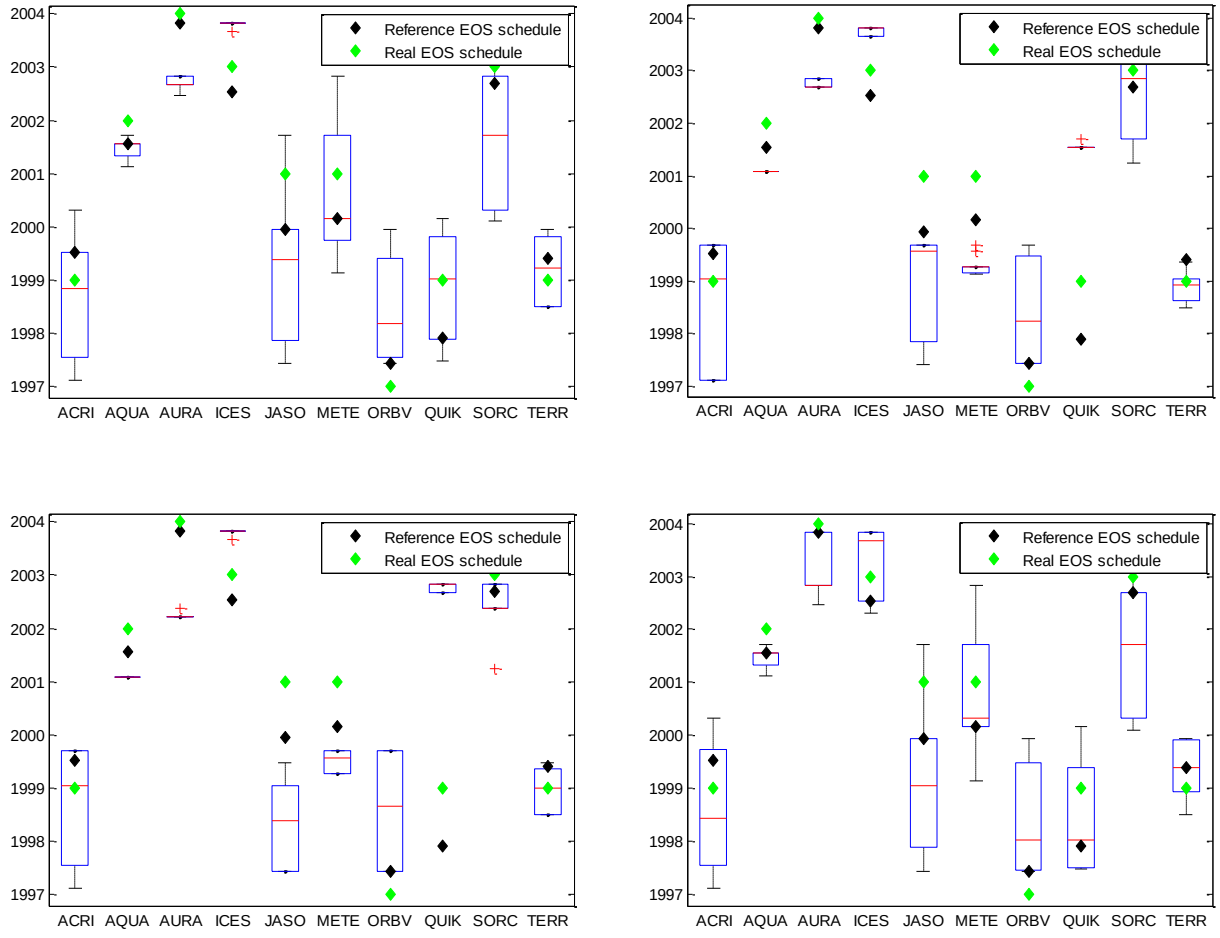


Figure 58: EOS scheduling tradespace exploration: launch dates statistics for four clusters of architectures shown in Figure 57

5.3.4.3 Discussion

The analysis of the EOS mission scheduling problem shows that the scheduling model at its current version is the one that is least capable to replicate results. The major differences are the following:

- AQUA and AURA are launched earlier for an increased discounted value
- ACRIMSAT and SORCE are launched earlier to cover the solar irradiance data gap
- In exchange, ICESAT, SEAWIFS, and QUIKSCAT are launched later

These differences are due to several simplifying assumptions in the model that affect the quality of the results:

- **Mission investment over time**: The model assumes that the budget for Earth science missions is entirely put on one mission at a time. This is particularly not true for the EOS case study. In fact, NASA budgets for Earth science between 1997 and 2004 are available on-line, and show that investments were systematically made on several missions at a time. A consequence of this is that the model does not allow for very close launches, since once a mission is launched, the development of the next mission has to start from the beginning. In reality though, up to four EOS missions were launched in 1999, not counting Landsat-7, which was technically also part of the EOS program, but was not included in the analysis because most of the money came from a different budget (USGS).
- **Choice of precursor missions for data continuity:** The list of precursor missions that included in the data continuity matrix plays a major role in the determination of the data continuity scores. For this EOS case study, all NASA, NOAA, ESA, CNES, and METEOSAT missions except EOS missions were included. Future analysis could include different sets of precursor missions.

5.4 Conclusion of the EOS case study

While internal verification of different components of the framework has been the object of earlier chapters, this chapter has dealt with overall model validation. The chapter has applied the framework presented in chapters 2 to 4 to a retrospective case study in order to show that it is capable of reproducing decisions made in the past, which is used as a proxy for modeling quality. While this is not a perfect measurement of performance, the ability of the model to replicate results is a necessary condition for it to be useful.

The framework has been applied to three SAPs: EOS instrument selection, EOS instrument packaging, and EOS mission scheduling. Results for the EOS instrument selection problem almost perfectly replicate the decisions made in the real program, with a few caveats that were identified and explained, mostly related to the EOSP polarimeter, needs of international partners, and couplings with the scheduling problem.

The EOS instrument packaging problem appeared as a harder problem to solve, but once the correct assumptions in terms of launch vehicles and standard buses were introduced into the model, the results were found to be in very good accordance with reality.

The results for the EOS mission scheduling problem were the least satisfactory. Model assumptions concerning the investment profile over time that have been used in previous work were deemed inappropriate for the EOS case study, which led to a few clear differences between the reference architecture and the best architectures identified by the model.

Overall, the results obtained are satisfactory enough to give us confidence to apply the framework to other case studies. In the next chapters, two case studies are studied in detail. After that, a cross-case analysis is conducted in Chapter 8.

6 Case Study 2: NASA Decadal Survey

6.1 Context and goals for the case study

In 2004, the NASA Office of Earth Science, the National Oceanic and Atmospheric Administration (NOAA) National Environmental Satellite Data and Information Service (NESDIS), and the U.S. Geological Survey (USGS) Geography Division asked the National Research Council (NRC) Space Studies Board (SSB) to "conduct a decadal survey to generate consensus recommendations from the Earth and environmental science and applications communities regarding a systems approach to space-based and ancillary observations that encompasses the research programs of NASA; the related operational programs of NOAA; and associated programs such as Landsat, a joint initiative of USGS and NASA." (NRC Committee on Earth Science and Applications from Space, 2007)

In response to this request, an ad-hoc NRC committee consisting of experts from different disciplines of the Earth sciences produced a report known as the "Decadal Survey". The DS lays out a reference architecture for an integrated EOSS for the next decade that will fulfill the needs of all the scientific communities in terms of space-based measurements, while also providing essential societal benefits (NRC Committee on Earth Science and Applications from Space, 2007).

This reference architecture consists of 15 missions for NASA and 2 missions for NOAA. A total of 39 instruments were assigned to these 17 missions on the basis of a variety of technical, scientific, and programmatic factors, including synergies between instruments, data continuity, orbit compatibility, different instrument maturity levels, and expected yearly budget. For each mission, the report provides a description of the scientific objectives fulfilled by the mission, the physical parameters measured, the instruments used, the orbit required, a rough estimation of the lifecycle mission cost, and the expected mission launch date.

This reference architecture was created based on a series of assumptions that were deemed reasonable at the time, such as mission cost estimates, yearly budget, and precursor missions outside the scope of the Decadal Survey study that were expected to be flying by this decade. However, some of these assumptions are no longer true.

First, mission cost estimates according to NASA have increased by 70% on average (Seher, 2009); note that this number accounts for the missions that have not yet started development and therefore have not yet had the opportunity to suffer any development-related cost overrun.

Second, yearly budget has decreased by about 50% with respect to the $750M/yr that can be inferred from the Decadal Survey[18]. And finally, some of the precursor missions have failed, or have been severely delayed (e.g. the Orbiting Carbon Observatory mission or OCO was lost at launch; the National Polar-orbiting Operational Environmental Satellite System, or NPOESS, was delayed, reorganized, and descoped; and the Glory mission was also lost at launch.) Therefore, the question arises whether this is still the best architecture possible given the current assumptions.

The purpose of this case study is thus to apply the methodology described in the thesis in order to provide recommendations for NASA, NOAA, and the USGS, in the form of alternative architectures that are potentially better than the reference architecture under the current set of assumptions.

6.2 Case study specific rules

6.2.1 Aggregation rules

6.2.1.1 1st level of decomposition of stakeholder needs: panels

Measurement requirements were categorized in six panels for the Decadal case study, according to the Decadal Survey report (NRC Committee on Earth Science and Applications from Space, 2007): climate (CLI), weather (WEA), water (WAT), land and ecosystems (ECO), human health (HEA), and solid Earth (SOL). These six panels were weighted in importance according to the findings of Tim Sutherland's thorough stakeholder analysis (Sutherland, 2009):

Table 37: 1st level of decomposition of stakeholder needs for the Decadal case study

Panel	Id	Description	Weight
Weather	WEA	Weather (including space weather and chemical weather)	21%
Climate	CLI	Climate variability and change	21%
Land	ECO	Land-use change, ecosystems dynamics, and biodiversity	21%
Water	WAT	Water resources and the global hydrological cycle	16%
Health	HEA	Human health and security	11%
Solid Earth	SOL	Solid Earth hazards, resources, and dynamics	11%
			100%

[18] This value of 750$M/year is not explicit in the Decadal Survey report, but it can be inferred from adding their own mission cost estimates and dividing the total cost by the 10 years they assumed it would take to launch them.

We note that in the actual Decadal Survey report there is a seventh panel named "Earth science applications and societal benefits". In reality, this panel acted as a cross-disciplinary panel and worked together with the six other panels in order to identify societal benefits that could stem from the missions proposed. **This seventh panel was left out of the analysis, as its objectives appear in the individual panel's reports.**

6.2.1.2 2nd level of decomposition of stakeholder needs: panel objectives

Scientific objectives were identified and ranked in importance for each of the six panels, based on the information available in the Decadal Survey report, and leveraging from previous work in the group by Theo Seher (Seher, 2009). This information was also contrasted with several experts that were involved in the Decadal survey committees. For example, seven objectives were identified for the weather panel:

Table 38: 1st level of decomposition of stakeholder needs for the Decadal case study: WEA panel objectives

Objective	Id	Description	Weight
1	WEA1	Atmospheric winds	19%
2	WEA2	High temporal resolution air pollution	15%
3	WEA3	All-weather temperature and humidity profiles	12%
4	WEA4	Comprehensive global tropospheric aerosol characterization	8%
5	WEA5	Radio Occultation	19%
6	WEA6	Comprehensive global tropospheric O3 measurements	15%
7	WEA7	Aerosol-cloud discovery	12%
			100%

These seven objectives were ranked in four groups of importance and assigned scores according to Seher's findings (Seher, 2009). Atmospheric winds and GPS radio occultation were top priorities of the panel, followed by air pollution and tropospheric ozone measurements. All-weather atmospheric sounding and aerosol-cloud properties are tied third in importance, leaving tropospheric aerosols as the least important objective for the panel. For the details of how these scores were calculated, the reader is referred to (Seher, 2009) or (Suarez, 2011).

Details about the 1st level of decomposition of stakeholder needs for the other six panels are provided in the Appendix 9.3.4. A total of 35 objectives were identified.

6.2.1.3 3rd level of decomposition of stakeholder needs: "subobjectives"

In the second and last level of decomposition of stakeholder needs, each of the 35 objectives was decomposed into as many subobjectives as required in order that a subobjective only makes reference to one measurement. 188 subobjectives were thus identified.

An example of this decomposition is shown in Table 18 for the radiation budget objective of the WEA panel. In this case, the capability to perform direct measurements of 3D atmospheric wind fields is given a higher priority than simpler indirect measurements of winds based for example on water vapor transport.

Details about the 2nd level decomposition of the rest of Decadal objectives are provided in the Appendix.

Table 39: 2nd level of decomposition of stakeholder needs for the Decadal case study: atmospheric winds subobjectives

Subobjective	Id	Description	Weight
1	WEA1-1	Atmospheric wind speed	20%
2	WEA1-2	Atmospheric wind direction	20%
3	WEA1-3	Ocean surface wind speed	30%
4	WEA1-4	Ocean surface wind direction	20%
5	WEA1-5	Water vapor transport winds	10%

6.2.2 Requirement satisfaction rules

For each of the 188 subobjectives, rules were created that express the measurement requirements for full satisfaction, and several cases of degraded satisfaction. An example of such rules is provided in Code 24 for the case of subobjective WAT4-1 concerning atmospheric humidity profiles.

```
(define-rule full-satisfaction-of-subobjective-WAT4-1
   "Conditions for full satisfaction of subobjective 1 of objective 4 of water panel"

IF there is a (Measurement  (of Parameter "1.3.1 Atmospheric humidity -indirect-")
(that is cloud-cleared yes) (with global coverage of a Region-of-interest ?x1) (and
Horizontal-Spatial-Resolution ?x2) (and sensitivity-in-low-troposphere-PBL High)
(that is taken-by ?who) (and has Temporal-resolution ?x3) (and Vertical-Spatial-
Resolution ?x4)
AND (?x1 ContainsRegion Global) ;; i.e. global coverage
AND (Horizontal-Spatial-Resolution ?x2 is at least Low-1km-10km)
AND (Temporal-resolution ?x3 is at least High-12h-24h)
AND (Vertical-Spatial-Resolution ?x4 is at least High-200m-orless)

   => (THEN)

ASSERT fact indicating (full-satisfaction (of subobjective WAT4-1) (from objective
"Water vapor transport") (related to parameter "1.3.1 Atmospheric humidity -indirect-
") (taken-by ?who))))
```

Code 24: Full satisfaction requirement rule for subobjective WAT4-1 from the Decadal case study

Additional information about the requirement satisfaction rules for the Decadal case study is provided in the Appendix 9.3.5. The complete set of rules is available on-line at: http://web.mit.edu/dselva/www/RBES/Decadal/.

6.2.3 Instrument capability rules

After describing stakeholder needs, the second important group of case study specific rules concerns instrument capabilities. 39 instruments were considered in this case study, and several characteristics necessary to assess requirement satisfaction and to perform lifecycle cost estimations were identified for each instrument (e.g., mass, power, data rate, dimensions).

The table containing instrument characteristics for the Decadal survey is provided in the Appendix 9.3.6. Furthermore, for each instrument, a rule was created that asserts the measurements that the instrument can take. An example of such rule for the ACE-ORCA instrument is provided in Code 25.

```
(define-rule CAPABILITIES::ACE-ORCA-measurements
   "Define measurement capabilities of instrument ORCA from mission ACE"

IF there is a (Manifested-instrument (Id ?id) (Name ACE-ORCA))

   => (THEN)

ASSERT Measurement (Parameter "3.1.1 Ocean color - 410-680nm -Chlorophyll absorption
and fluorescence, pigments, phytoplankton, CDOM-") (Id ACE-ORCA1)))
ASSERT Measurement (Parameter "3.1.2 Extended ocean color - UV -enhanced DOC, CDOM-")
(Id ACE-ORCA2)))
ASSERT Measurement (Parameter "3.1.3 Extended ocean color - NIR -atmospheric
correction-") (Id ACE-ORCA3)))
ASSERT Measurement (Parameter "1.1.1 aerosol height/optical depth") (Id ACE-ORCA4)))
ASSERT cross-registered (measurements ACE-ORCA1 ACE-ORCA2 ACE-ORCA3 ACE-ORCA4 )
            (degree-of-cross-registration instrument) (platform ?id)))
    )
```

Code 25: Instrument capability rule for the Decadal ACE-ORCA instrument

This rule asserts the four measurements ACE-ORCA can take when it detects that a copy of this instrument has been manifested. The measurements are assigned ID#s from ACE-ORCA1 to ACE-ORCA4. These ID#s play a role in the explanation facility. In addition to asserting these four measurements, this rule also asserts that these four measurements are cross-registered because they are taken by the same instrument. This is also important, as some synergy rules only apply to cross-registered measurements. Finally, note that this rule simply asserts the measurement facts without filling in the measurement attributes such as spatial resolution. Indeed, attribute inheritance from instruments and missions to measurements is carried out by the attribute inheritance rules, which are executed a step later in the VASSAR methodology.

Detailed information concerning the Decadal instrument capability rules is provided in the Appendix. This information comes from the Decadal Survey report and individual instrument or mission publications (e.g., (Cline, Davis, & Yueh, 2005; Freeman, Donnellan, & Rosen, 2008; Gaier et al., 2010; Jet Propulsion Laboratory, 2007; B. Lambrigtsen, Brown, Gaier, Kangaslahti, & Tanner, 2008; NASA Earth Science and Technology Office, 2009)), and was partially verified by experts.

6.3 Results

6.3.1 Preliminaries

Before presenting the results of the three sub-problems (i.e., instrument selection, instrument packaging, and mission scheduling), intermediate results that are used in the three sub-problems are presented and discussed. In particular, mission scores, single instrument scores (both marginal and in isolation), and bilateral interactions between instruments both at the science and engineering levels (i.e. S-DSM and E-DSM) are introduced.

6.3.1.1 Mission scores

Mission scores and a short discussion about each mission are provided in this subsection with the goal of introducing the missions architected in the Decadal survey report, and assessing their relative importance to the stakeholders of the project. Mission scores are shown in Figure 59.

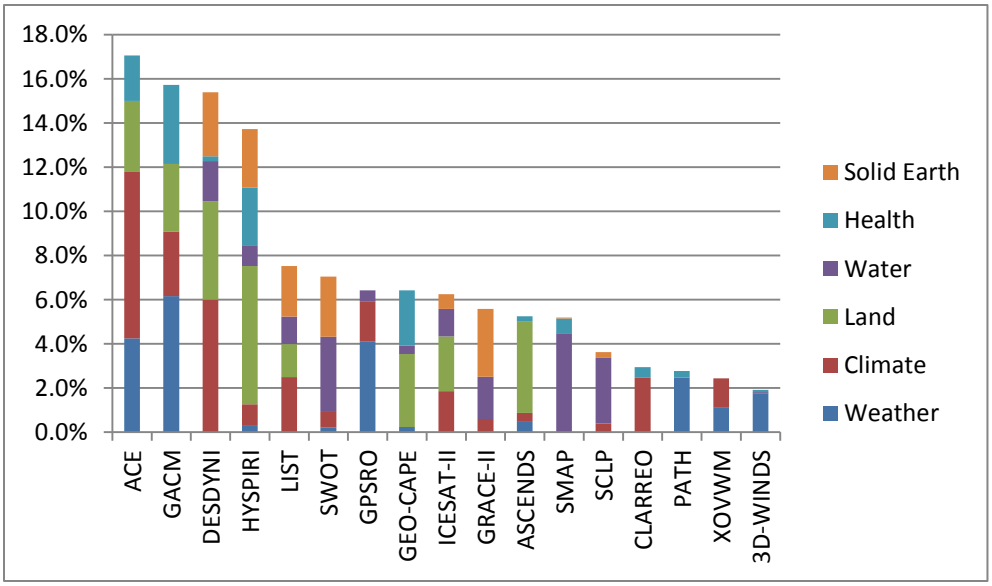


Figure 59: Decadal Survey reference mission scores by panel

These missions are commented one by one in the following paragraphs.

From the two instruments in the SMAP mission (a radar and a radiometer), we observe that the radar gets a score of 4.03% whereas the radiometer gets a score of 1.76%. These scores are explained as follows: a) the radar by itself cannot measure soil moisture with the accuracy of 4% required by the water panel; b) the radiometer by itself cannot measure freeze-thaw at all and therefore completely misses that objective; c) the radiometer by itself has the required accuracy to measure soil moisture, but it does not meet the requirements in terms of spatial resolution of 10km or better to satisfy the hydro-meteorology community.

The synergistic combination of the radar and the radiometer achieve the required accuracy and spatial resolution thanks to a disaggregation scheme, and thus get a score of 5.5%. **All this is captured in the explanation facility, including the data processing**.

The ICESAT-II lidar can potentially provide useful observations of sea ice, ice sheets, and vegetation structure. However, several concepts are currently being studied, and depending on which concept is finally selected, the objectives related to one or more of these disciplines can be affected. In particular, three concepts are currently being considered: a) a dual-wavelength full waveform lidar with multiple lasers similar to the ICESAT/GLAS instrument; b) a single-wavelength (1064-nm) quad-beam full waveform lidar with some cross-track capability and reduced FOV; c) a single-wavelength micro-pulse photon-counting lidar with multiple (16) beams.

As of March 2012, the single-wavelength photon counting lidar has been selected as primary concept for the ICESAT-II mission. This affects vegetation structure measurements, which are best done with full waveform lidars, because the returned waveform contains several features that correspond to different types of vegetation. In any case, the score of the ICESAT-II lidar, assuming the photon counting laser is finally chosen, is 6.2%. This score is obtained through full satisfaction of the ice sheet and sea ice objectives, plus partial satisfaction of the vegetation structure objectives (only measurements of vegetation height can be taken). Some benefit is also obtained from partially satisfied aerosol and cloud properties objectives, as well as land topography.

The CLARREO mission carries three instruments: a hyperspectral Fourier transform spectrometer (FTS) in the SWIR and TIR for characterization of Earth's emitted radiation, a suite of three hyperspectral spectrometers in the VNIR and SWIR for characterization of solar reflected radiation, and a GPS RO for atmospheric sounding. As of March 2012, the number of spacecraft of the mission is still unclear.

The CLARREO mission as envisioned today gets a score of 5.3%, through full satisfaction of the Earth radiation budget objectives from the climate panel, as well as the GPS RO reference objectives of the climate panel. It is therefore seen essentially as a climate mission. The infrared FTS by itself gets a score of 0.8%, whereas the suite of three VNIR+SWIR spectrometers gets a score of 0.6%. There are no pure synergies between these two instruments, since the model assumes that the VNIR and TIR capabilities are essentially additive, and that they can be cross-registered from different platforms.

Thus, an important fraction of the value of the CLARREO mission actually comes from the GPS part of the mission. The reason for this is that although the radiation budget objectives are clearly very important, they only provide value to the climate community. Other missions provide some value to several disciplines, which results in this model in higher scores. While the instruments on the CLARREO mission could potentially be capable of providing value to several other communities due to the potential of hyperspectral measurements in the SWIR and LWIR, this has not been taken into account in the model, as it does not appear in the mission proposal.

DESDYNI's instruments are a multi-beam infrared lidar and an L-band repeat pass interferometric SAR with multiple polarizations. The primary application of the SAR is to measure surface deformation in order to predict the likelihood of earthquakes, volcanic eruptions, and landslides (Dubayah et al., 2008; Freeman et al., 2008), (Johnson, Rosen, Hensley, & Freeman, 2009). It also has a polarimetric mode that enables measurements of 3D vegetation structure, biomass, and thus carbon. The primary application of the lidar is to measure vegetation structure, but it can also measure ice thickness. Together, the two instruments provide useful measurements for the solid Earth, cryosphere, and ecosystem communities. The DESDYNI mission as envisioned today gets a score of 10.6%. This value comes essentially from satisfaction of objectives related to surface deformation, ice sheet thickness and velocity, and vegetation height. The SAR by itself gets a score of 8.0%, missing the vegetation height objectives. The lidar by itself gets a score of 4.8%, missing several objectives in terms of surface deformation, vegetation structure, and other subobjectives that are partially satisfied by the DESDYNI SAR, such as soil moisture, or snow cover. The synergistic behavior between the SAR and the lidar stems from the fact that the SAR provides the global coverage and the 3D structure, while the lidar provides less frequent, high accuracy measurements of forest height and size distribution of the vegetation. However, it the Decadal survey report explicitly mentions that the time scale of cross-registration required to capture these synergies is on the order of one or more weeks. As a consequence, the synergy score between these two instruments is zero, because the synergy is also captured when the instruments are flown on different spacecraft.

The SWOT mission carries a wide-swath Ku-band interferometer radar altimeter as main payload. This instrument is expected to increase the value of satellite altimetry measurements by providing better coverage, spatial, and temporal resolution through the use of a wide-swath instead of the classical narrow-swath nadir-looking altimeter.

However, the inclusion of this swath capability introduces a variable error that varies across cross-track distance. In order to keep the error budget below 4-5 cm, it is necessary to fly several other instruments, including a dual-frequency nadir-looking altimetry for precise ionospheric correction, a GPS receiver for precise orbitography and additional measurements of total electron content, and a three-channel microwave radiometer for wet tropospheric correction.

This complete payload achieves a score of 7.8% through satisfaction of objectives from the climate, water, and solid Earth panels related to ocean circulation, river and lake elevation, hydrocarbon reservoir monitoring, and ocean bathymetry. Furthermore, some objectives in the weather panel are satisfied by the supporting instruments (in particular the MWR, and the GPS receiver if it is used for atmospheric sounding). The value of the mission is affected if any of the supporting instruments is missing: if the GPS is not selected, the POD contribution to error budget increases; if the MWR is not selected, the wet and dry tropospheric contributions to the error budget increase; if the dual-frequency nadir altimeter is not selected, the error budget is also slightly perturbed through the ionospheric contribution.

The GEO-CAPE mission as proposed in the Decadal Survey consists of: a) a UV and VNIR wide area spectrometer capable of mapping North America every hour with a spatial resolution of 7km at nadir; b) a steerable 250-m spatial resolution hyperspectral spectrometer with a 300km swath; c) a gas filter correlation radiometer to continue the CO measurements of MOPITT. These instruments together can measure ocean color with emphasis on coastal zones, aerosol optical depth, and vertical profiles of NOx, formaldehyde, and ozone precursors, with limited sensitivity in the troposphere. Under these assumptions, the GEO-CAPE mission gets a score of 7.9% from satisfaction of coastal ecosystems objectives. The UV/VNIR spectrometer by itself gets a score of 4.3%, whereas the steerable imager gets a score of 3.8%. The combination of these two instruments can provide cloud-cleared high spatial resolution ocean color and atmospheric chemistry products. The gas correlation radiometer by itself provides negligible value because it lacks a cloud mask, sensitivity in the troposphere, and vertical spatial resolution (its value is about 0.2% if we assume that a cloud mask ancillary product is available). Note that the GEO-CAPE mission by itself does not have enough sensitivity in the lower troposphere and vertical spatial resolution to satisfy many of the atmospheric chemistry objectives. These objectives are fully satisfied by the synergistic combination of GACM and GEO-CAPE.

The HYSPIRI mission carries two high spatial resolution hyperspectral sensors: one in the 8-12um region, similar to TERRA/ASTER, and one in the VNIR/SWIR region, similar to EO/HYPERION. The combination of high spatial resolution, hyperspectral capability, and steering capability, makes this mission attractive to virtually all disciplines of the Earth sciences.

This is apparent in the model, as HYSPIRI provides some value to all panels. The main applications of HYSPIRI are in solid Earth (volcanoes, surface composition), and ecosystems (vegetation state, land use and landcover status, coral reef health and extent, fire monitoring), and human health (land and ocean surface temperature, land use, land cover, vegetation), but some value is also provided to the weather, climate, and water panels through surface temperature measurements, cloud information, land use, vegetation, and land cover status. Moreover, many of these applications also have very high societal value (e.g., disaster monitoring). As a consequence of all this, HYSPIRI gets a score of 13.1%. The TIR part of HYSPIRI scores 10.0% and the VIS part gets a score of 10.7%.

The ASCENDS mission features a dual wavelength CO2-O2 laser sounder with an integrated laser altimeter to measure total column length, together with an infrared radiometer for atmospheric correction and cloud mask, and a gas filter correlation radiometer for CO measurements. The mission as proposed today gets a score of 5.3%, which is on the average of most Decadal missions. It is interesting to note that an important fraction of this score comes from measurements that are meant to support the main CO2 measurement, namely the total column length and vertical profiles of thin clouds and aerosols taken by the altimeter. The lidar by itself achieves a score of 1.6% (most of this loss is due to a missed synergy with the IR sounder), and the gas filter correlation radiometer achieves a score of 0.8% by itself. If the O2 channel is removed from the sounder, the resulting single-wavelength lidar has a score of 1.6%.

The ACE mission carries a payload consisting of four instruments: a) a multi-beam dual-wavelength lidar for precise cloud and aerosol height measurements; b) a dual-frequency (94GHz and 34GHz) cloud radar for cloud properties (mostly droplet size, liquid water, and ice/liquid phase transition); c) a multi-angle, multi-wavelength polarimeter for cloud and aerosol properties (mostly particle micro-physical and radiative properties); d) a UV/VIS imaging spectrometer that would provide continuation of all MODIS, VIIRS, and OMI aerosol and ocean color bands.

Most of the aerosol products are generated from a combination of the lidar and the polarimeter, the lidar providing the vertical spatial resolution, and the night capability, while the polarimeter provides improved information of particle shape and scattering properties, in particular in the presence of optically thick clouds.

The ACE program is currently considering several different packaging architectures including one monolithic spacecraft, and 4 two-spacecraft architectures. They are also considering the inclusion of several other instruments in the suite. The mission as planned today (one spacecraft, no other instruments) gets a score of 15.3%.

This high score is explained because the mission satisfies many high priority objectives of the climate, weather, and ecosystems panels (clouds, aerosols, and ocean color). Some value is provided as well to the health panel mainly through the aerosol measurements. The value of the ocean color instrument by itself is 3.7%. The value of the cloud radar by itself is of 5.1%, satisfying objectives from the weather, climate, and to a lesser extent of the health panel through indirect precipitation measurements. The value of the lidar and the polarimeter are harder to separate because they are highly synergistic in the production of the aerosol data products. As a group, their value is 8.1%.

GPSRO is a constellation of satellites in LEO carrying GPS receivers in radio occultation mode. Assuming that the constellation is populated enough to guarantee the desired number of soundings per day (namely 2500 or more), the score of the GPSRO constellation is 5.9%. As we decrease the number of satellites in the constellation, the value decreases (5.0% for 5 satellites or 2250 soundings per day, 4.0% for 4 satellites of 2250 soundings per day, and so forth).

XOVWM is a mission for measuring ocean surface wind speed and directions in all-weather and all-wind conditions, as well as near the coasts, which calls for a higher spatial resolution. The concept proposed in the Decadal Survey for this mission features a dual-frequency Ku and C-band scatterometers with SAR capability in C-band, and an X-band radiometer for atmospheric correction. This payload on a single spacecraft, in isolation of the rest of the global EOSS, gets a score of 2.5%, due to the lack of temporal resolution. Losing cross-registration with the X-band radiometer results in a loss of 0.6% due to atmospheric correction. Losing cross-registration with the C-band scatterometers results in a loss of 0.6% of the score, due to the complementary nature of the two measurements (sensitivity in rain and sensitivity in high winds versus horizontal spatial resolution). In reality, the possibility of adding more spacecraft into a constellation is being studied, and data is also available from European scatterometers (e.g., ASCAT). If this data is available and the temporal resolution requirement is achieved, then the score becomes 3.5%.

The GACM mission consists of a UV/VIS spectrometer for ozone measurements and precursors, a SWIR spectrometer for CO, CH4, and other GHGs, a microwave limb sounder, and a differential absorption lidar to achieve the high vertical spatial resolution. Together, these four instruments achieve a score of 12.1%, satisfying objectives from all panels except the water and solid Earth panels.

GACM is a highly synergistic set of instruments, as they are sensitive in different layers of the atmosphere. For this reason, some individual instrument scores in isolation are relatively low, such as the 1.4% for the MW limb sounder, which has sensitivity in the upper troposphere and stratosphere, and the 0.5% for the SWIR instrument, which only provides total column measurements.

The UV/VNIR instrument gets a score of 8.8% mainly due to its vertical resolution and sensitivity in the lower troposphere and planetary boundary layer, as these are two of the priorities of the Decadal survey committee.

The PATH mission intends to put a microwave sounder such as the AMSU on the EOS spacecraft on a GEO orbit. The goal is to provide high temporal resolution, all-weather temperature and humidity. In order to achieve the required spatial resolution from a GEO orbit, the instrument uses 2-D interferometry, similarly to what the European SMOS mission does. The current baseline for the PATH mission is to fly the GEOSTAR sensor as an opportunity payload on one of the GOES spacecraft (B. Lambrigtsen et al., 2008). The science score of the GEOSTAR instrument is 2.8%.

The LIST mission intends to put into orbit the first imaging lidar for high resolution land topography. The concept for the instrument is still uncertain. Three major concepts are competing: a single-beam laser with a scanning mechanism (LVIS), a single-beam laser with an optical device that splits the beam into several beams (LRO), and a push-broom lidar (LL). For any of the three instrument concepts, we assume a horizontal spatial resolution of 5m with 10-cm accuracy in the altimetry measurement. Such instrument achieves a score of 4.7%, mainly through satisfaction of the high resolution surface topography objective, as well as partial satisfaction of vegetation height and ice sheet topography.

The SCLP mission is the first satellite mission capable of measuring fresh water in snow on land and ice sheets. It features a dual-frequency X-/Ku band SAR for high spatial resolution measurement, and a passive radiometer for cross-calibration with heritage measurements. The two instruments together achieve a score of 3.4% that come essentially from snow objectives.

The 3D-WINDS mission is the first mission to measure tropospheric winds directly from space using Doppler lidar. This is a measurement of the outmost importance for numerical weather prediction. However, since this mission is very focused on the satisfaction of just one objective, it gets a relatively low score (1.9%). The mission consists of two Doppler wind lidars, a non-coherent UV lidar, and a coherent $2.06\mu m$ lidar. They have complementary sensitivities: while the UV lidar has good sensitivity in the presence of high concentration of aerosols, the CO2 lidar has better sensitivity in clean air. Together, they provide sensitivity in the lower and upper troposphere.

This subsection discussed mission scores, and presented the reader with the main aspects and trade-offs of each mission. In the following section, individual instrument scores (marginal, and in isolation) are discussed. These scores are important for the instrument selection and packaging problems.

Before ending this section, mission cost estimates as provided by the model are given in Figure 60. These cost estimates are used in the scheduling problem, and they will also be used in the GEOscan case study in order to compare their relative cost-effectiveness.

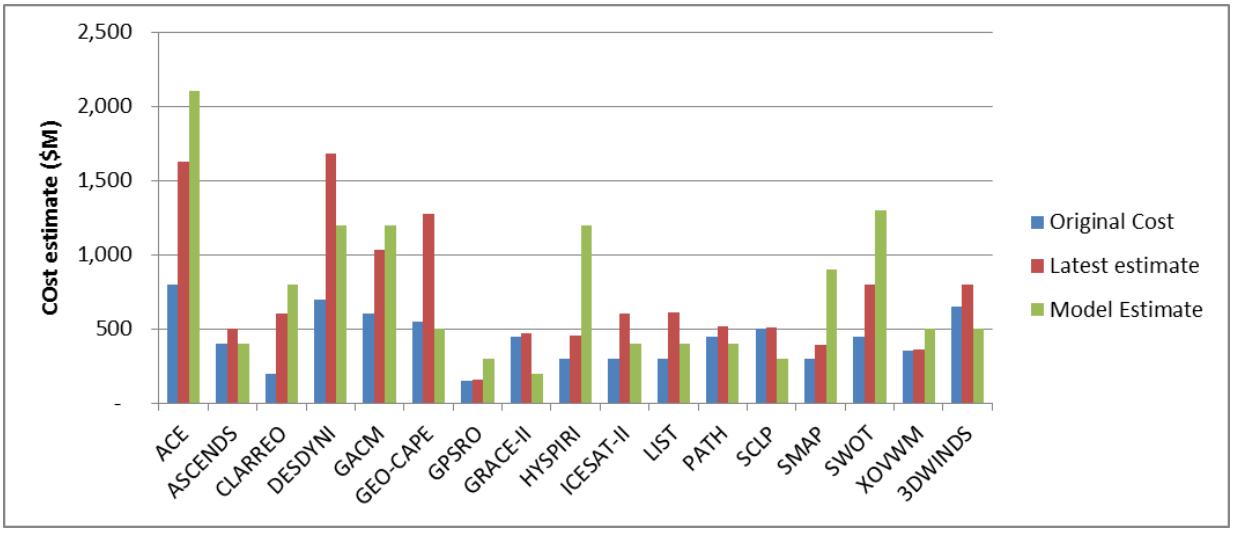


Figure 60: Decadal mission cost estimates (original estimates by NRC, latest estimates by NASA, model estimates)

6.3.1.2 Instrument scores

Instrument scientific scores in isolation for the Decadal survey case study are presented in Figure 61. The top four instruments in Figure 61 are well correlated with the top missions in Figure 59, except for the ACE instruments, which get lower individual scores. Two explanations are possible: a) instruments of the ACE mission have low redundancies between them; b) instruments of the ACE mission have high pure synergies between them. Reality is a combination of these two reasons. The ACE CPR and ocean color instruments have zero redundancy between them, so their capabilities are essentially superimposed.

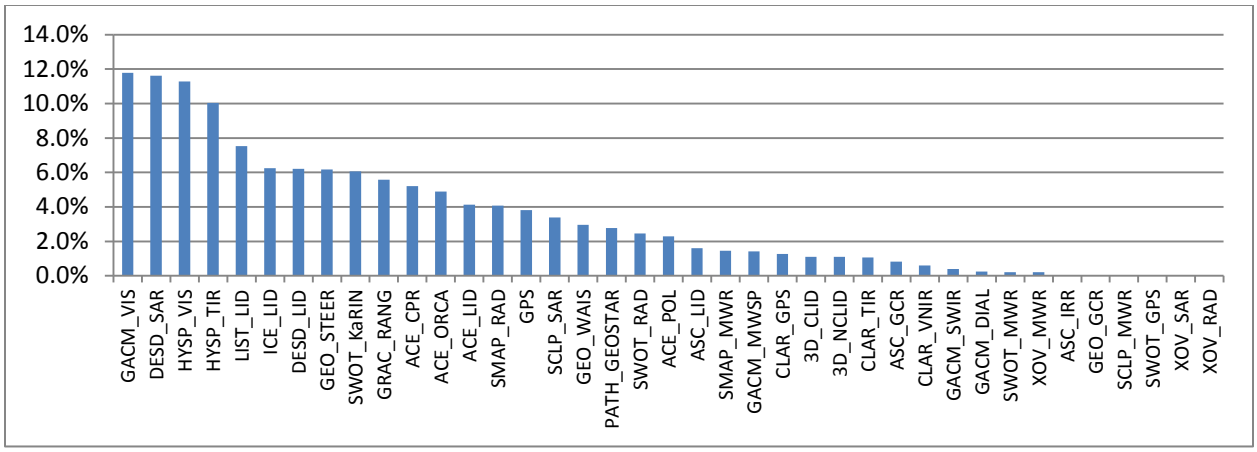


Figure 61: Decadal instruments benefit scores in isolation

On the other hand, the ACE lidar and polarimeter have very high synergies in the generation of aerosol data products. As a consequence, the synergistic combination of these two instruments gets a higher score than their plain superposition. Furthermore, there are several instruments with very low or null scores. Most of these instruments are supporting primary instruments, such as the infrared radiometer for ASCENDS, the microwave radiometer for SWOT, XOVWM and SCLP, and the GPS receiver for SWOT. The real value of these instruments is thus the synergy with their corresponding primary instruments. Furthermore, in the case of the scatterometry mission, the SAR instrument without the rest of the payload (C-band real aperture scatterometer, microwave radiometer, GPS) is unable to meet the mission requirements and results in a null score.

Figure 62 shows the cost-effectiveness of these instruments, computed by dividing the individual instruments scores in isolation by the individual costs. These costs represent an estimation of the lifecycle cost of a single-instrument mission carrying that instrument.

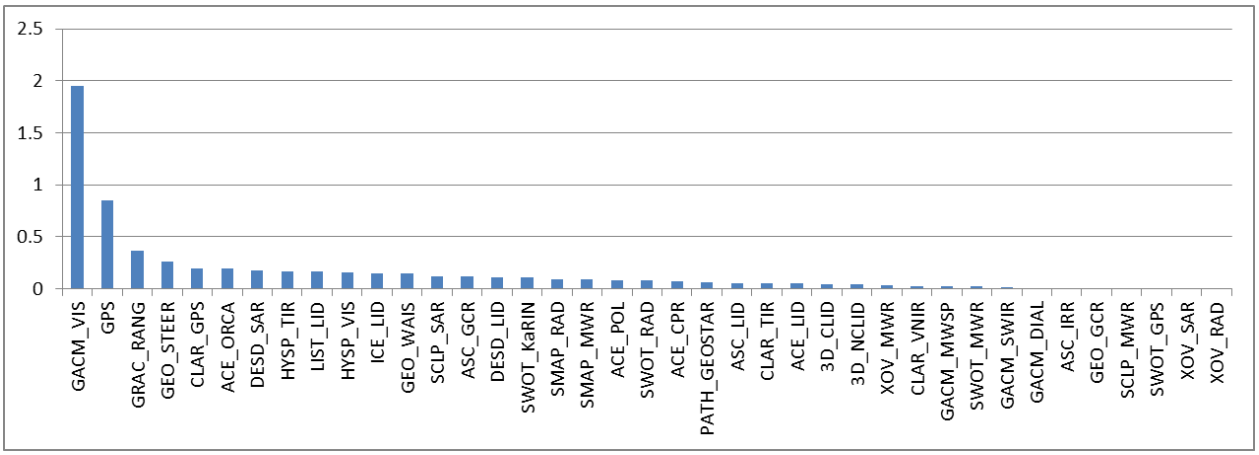


Figure 62: Decadal instruments cost-effectiveness when considered in isolation

It is interesting to see that the most cost-effective instrument by almost a factor of two is the UV/VNIR spectrometer in GACM. This is so because this instrument satisfies many important objectives of the weather, climate, and health panels with a relatively mature technology. The GPS receivers in radio occultation mode also appear in the front of the list, due to their extremely low cost and relatively high return for the weather and climate panels.

Marginal descoping scores $V_i^{'} = V_{\{ref\}} - V_{\{ref \setminus i\}}$ are now considered, as these capture synergies between instruments. The reference selection architecture for the Decadal case study is as described in the previous subsection, and therefore it consists of 39 instruments. This architecture is consistent with the report as it was released in 2007, and it does not reflect some of the latest changes.

For example, in the ACE mission, other instruments in addition to the polarimeter, the lidar, the cloud radar, and the spectrometer are currently being considered[19], but these are left out of this preliminary analysis, as not enough information is available at the moment. The marginal scores of these instruments with respect to the Decadal reference architecture are provided in Figure 63.

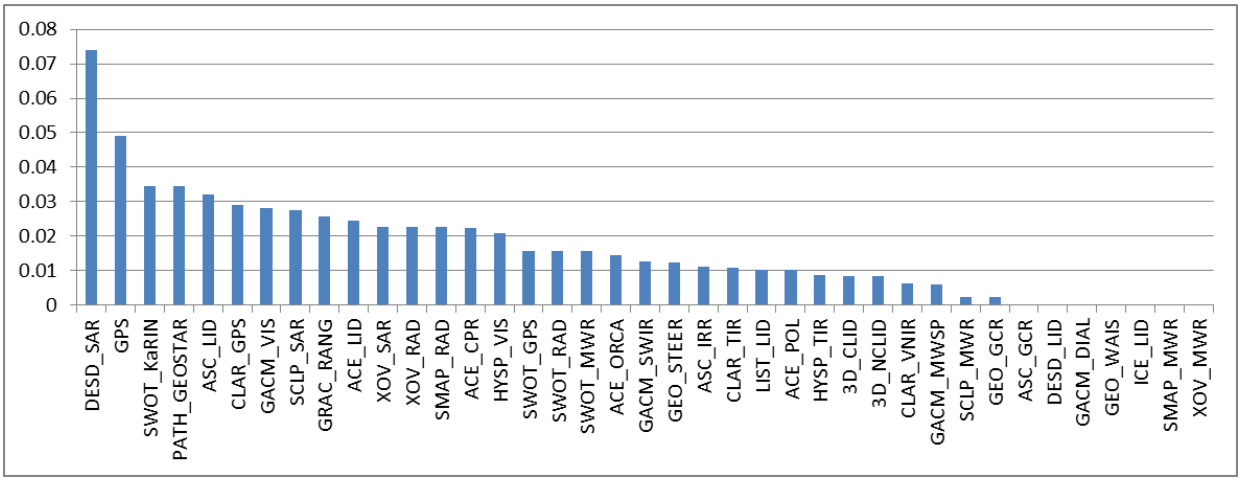


Figure 63: Marginal scores for the Decadal instruments w.r.t. the reference Decadal architecture

It is interesting to note that the DESDYNI SAR, which was virtually cancelled (i.e., out of the NASA budget) during the last fiscal year, is actually the instrument with the highest marginal score[20]. The **SARs generally achieve high scores (both marginal and in isolation)**, as they have unique imaging all-weather high resolution imaging capabilities.

[19] See http://dsm.gsfc.nasa.gov/ace/documents/ACE_Report13_Mission_v9.pdf

[20] DESDYNI appeared again in the NASA budget in the presidential request for FY12

Another interesting point is the fact that most **lidars generally obtain high scores in isolation but low marginal scores**. In this case, the reason is that the model detects several lidars as mostly redundant in their capabilities. Part of this is due to insufficient modeling fidelity of the current tools at the instrument level. An analysis of the differences between the 8 lidars in the Decadal survey shows that that there are four main attributes that set the concept of a lidar: a) the spectral content, which can include one or more of the following bands: UV (O3) band, 532nm band (vegetation), 760nm band (O2), 1064nm band (clouds and aerosols), 1270m, (O2) 1570nm (CO2), 2060 nm (CO2); b) the number of beams (single beam for sounding versus multi-beam for imaging); c) the property being measured, which typically is backscatter intensity for aerosol and cloud properties, difference in intensity between two bands for laser sounding, time delay in altimetry, and Doppler shift in wind applications; d) technology of the receiver: full waveform versus photon-counting.

The eight lidars in the Decadal survey differ on one or more of these characteristics. However, there have been proposals to join several lidar missions into a single instrument (e.g. the Decadal Survey report mentions that the DESDYNI lidar could be replaced by an ICESAT-II lidar augmented with multi-beam capability).

There are several other particular examples of instruments that obtain low marginal scores. For example, the X-band passive microwave radiometer for XOVWM gets a very low marginal score because it is essentially a cross-calibration instrument.

6.3.1.3 S-DSM and E-DSM

Bilateral interactions between the instruments are captured by the B-DSM and the C-DSM defined in Section 9.1.2, which were relabeled as S-DSM and E-DSM for the case of EOSS.

The S-DSM for the Decadal instruments considered in the packaging problem is shown in Table 40. In Table 40, each cell was computed as the difference of the score of a two-instrument mission minus the score obtained when the two single-instrument subobjectives are superimposed, i.e.: $\text{S-DSM}_{\{i,j\}} = V_{\{i,j\}} - V_i \cup V_j$ where the operator $\cup$ computes a score by aggregating the maxima of the individual instrument subobjective satisfaction scores, and thus takes into account redundancies between instruments. Therefore, positive values of S-DSM (green cells in Table 40) indicate positive synergies between instruments, and negative values (red/orange cells in Table 40) indicate negative scientific synergies. Negative scientific synergies are possible due to design compromises in the presence of multiple instruments that lead to suboptimal science performance by one of the instruments. For example, instruments taking measurements in which diurnal sampling plays an important role in the error budget (e.g., altimeters) may be affected by being put together with other instruments because that a compromise in design may lead to selecting a suboptimal orbit (namely, an SSO).

Table 40: S-DSM for the Decadal case study

	ASC_LID	ASC_GCR	ASC_IRR	CLAR_TIR	CLAR_VNI	CLAR_GPS	DESD_SAR	DESD_LID	HYSP_TIR	HYSP_VIS	ICE_LID	SMAP_RA	SMAP_MV
ASC_LID	0.000	0.000	0.026	-0.002	-0.018	0.000	-0.076	0.000	-0.044	-0.051	-0.014	0.000	-0.004
ASC_GCR	0.000	0.000	0.003	-0.002	-0.012	0.000	-0.068	0.000	-0.042	-0.044	-0.008	-0.008	-0.002
ASC_IRR	0.000	0.000	0.000	0.000	-0.005	0.000	-0.061	0.000	-0.038	-0.037	0.000	0.000	0.000
CLAR_TIR	0.000	0.000	0.000	0.000	-0.014	0.000	-0.071	-0.002	-0.044	-0.047	0.000	-0.011	-0.003
CLAR_VNI	0.000	0.000	0.000	0.000	0.000	-0.015	-0.107	-0.057	-0.065	-0.044	-0.046	-0.025	-0.005
CLAR_GPS	0.000	0.000	0.000	0.000	0.000	0.000	-0.067	0.000	-0.044	-0.048	0.000	0.000	-0.003
DESD_SAR	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.085	-0.142	-0.131	-0.090	-0.078	-0.082
DESD_LID	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.068	-0.092	-0.096	0.000	-0.015
HYSP_TIR	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.074	-0.081	-0.054	-0.049
HYSP_VIS	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.081	-0.058	-0.038
ICE_LID	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	-0.023	-0.011
SMAP_RA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.011
SMAP_MV	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

The pictorial representation of this adjacency matrix is shown in Figure 64. We note that there are very few truly synergistic pairs of instruments, due to competing orbit requirements between spacecraft. The only positive interactions on Figure 64 concern the ASCENDS instruments between them, and the SMAP instruments between them. Since there are many other synergies between instruments, this suggests that the loss due to competing orbit requirements and limited resources on the spacecraft is greater than the gain from pure synergies. However, note that this S-DSM only captures bilateral interactions, and therefore it misses synergies that require more than two instruments.

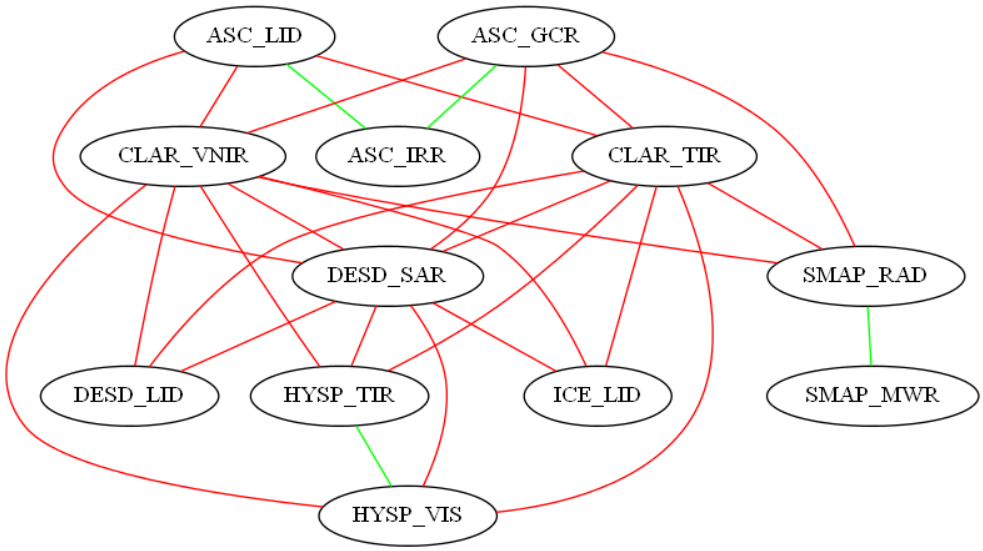


Figure 64: Pictorial representation of the S-DSM for the Decadal case study

On the engineering side, the E-DSM is shown in Figure 65.

	ASC_LID	ASC_GCR	ASC_IRR	CLAR_TIR	CLAR_VNI	CLAR_GPS	DESD_SAR	DESD_LID	HYSP_TIR	HYSP_VIS	ICE_LID	SMAP_RA	SMAP_MV
ASC_LID	0	120	108	146	781	-18	2568	-17	750	3040	385	561	406
ASC_GCR	0	0	-25	-17	83	-13	55	93	74	77	-2	-23	-24
ASC_IRR	0	0	0	-36	69	-33	56	-26	76	80	-14	-25	95
CLAR_TIR	0	0	0	0	144	-22	135	129	156	163	18	-9	60
CLAR_VNI	0	0	0	0	0	18	-76	830	-54	-61	606	299	-118
CLAR_GPS	0	0	0	0	0	0	13	-19	20	21	-15	-14	-71
DESD_SAR	0	0	0	0	0	0	0	2445	-87	-64	2313	732	-145
DESD_LID	0	0	0	0	0	0	0	0	795	802	387	482	468
HYSP_TIR	0	0	0	0	0	0	0	0	0	-57	644	455	-124
HYSP_VIS	0	0	0	0	0	0	0	0	0	0	581	464	-112
ICE_LID	0	0	0	0	0	0	0	0	0	0	0	404	467
SMAP_RA	0	0	0	0	0	0	0	0	0	0	0	0	25
SMAP_MV	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 65: E-DSM for the Decadal case study

In this case, each element of this matrix is computed as the difference between the costs of the two-instrument missions minus the sum of the costs of the two single-instrument missions.

Hence, positive entries of this matrix represent negative interactions between instruments, while positive entries of this matrix represent potential savings by putting the instruments on a shared platform.

The graphical representation of this adjacency matrix is provided in Figure 66. It is interesting to see for example that some instruments such as the GPS have green connections with most other instruments, which means that they are good candidates for opportunity payloads, while others have red connections with most instruments, which suggests that they are good candidates for flying on dedicated spacecraft. Again, however, this matrix only captures bilateral interactions.

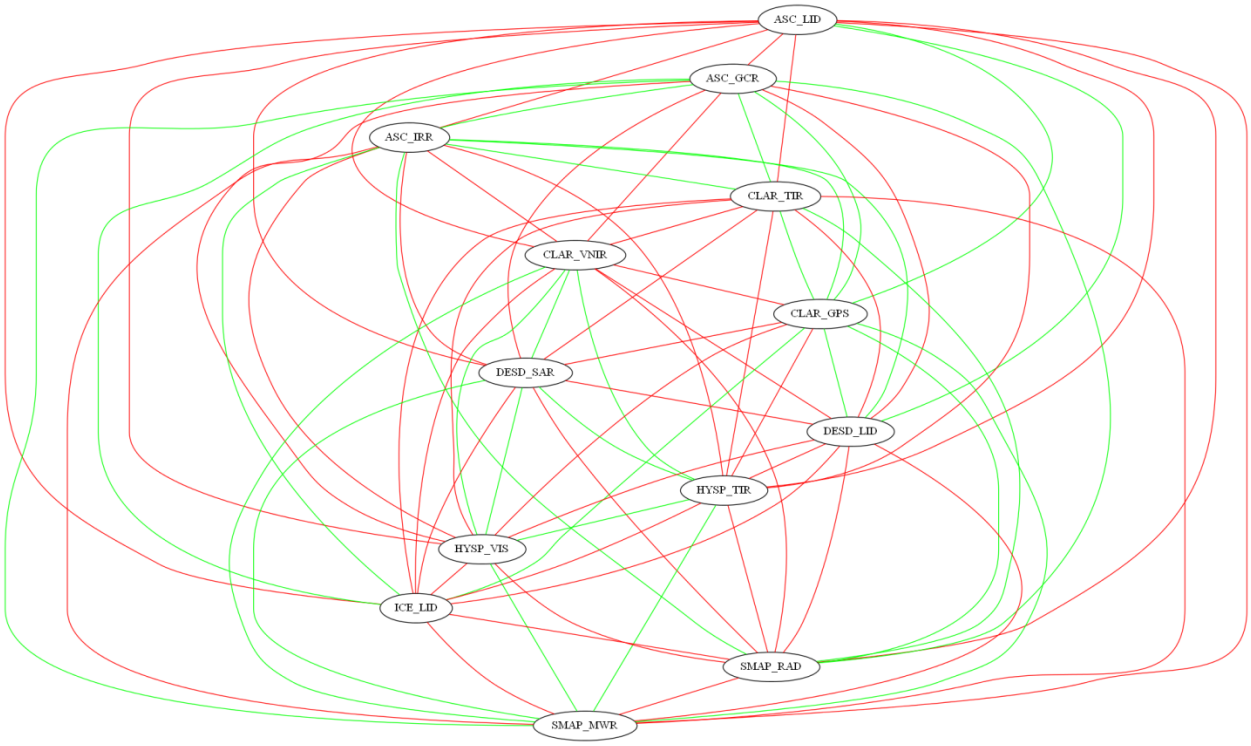


Figure 66: Pictorial representation of the E-DSM in the Decadal case study. Only negative interactions are shown.

The complete S-DSM and E-DSM containing all instruments, as opposed to only the instruments considered in the packaging problem, are available on http://web.mit.edu/~dselva/www/RBES/Decadal/.

6.3.1.4 Data continuity matrix

Introduced in Section 4.4.7, the data continuity matrix DCM is a multi-domain matrix where $DCM(i,j)$ represents the number of instruments that take measurement i during time interval j.

The data continuity matrix for the Decadal case study was computed by considering only domestic missions (i.e., no international missions), obviously except for the planned Decadal missions, between the years 2010 and 2025. Only a reduced subset of measurements deemed critical in terms of data continuity were considered. The matrix is shown in Figure 67.

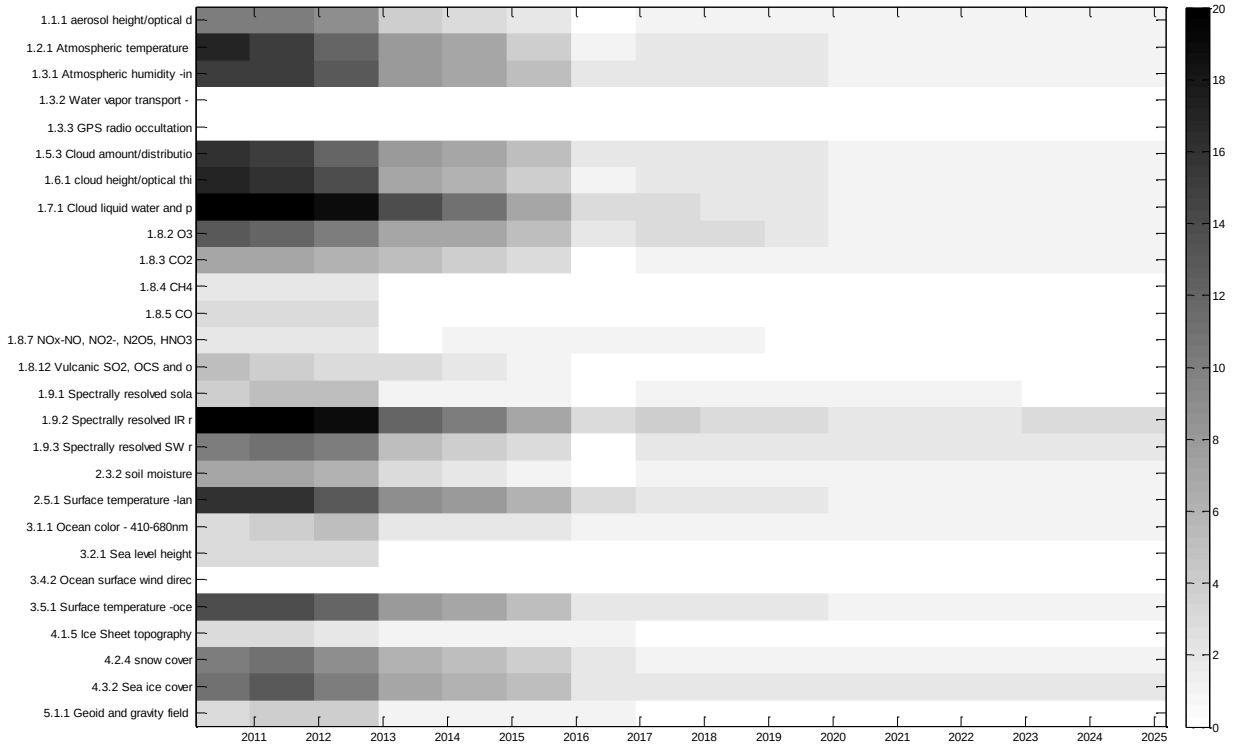


Figure 67: Data continuity matrix for the Decadal Survey case study

As shown in Figure 67, the major potential data gaps that are likely to appear in this decade if only NASA missions are considered are the following:

- Water vapor transport
- GPS RO
- Scatterometry

And to a lesser extent:

- Atmospheric chemistry
- Altimetry
- Ice sheet topography
- Ocean color
- Gravity field

Hence, the data continuity metric from the mission scheduling problem will favor architectures covering at least partially these data gaps. It may also be considered to add hard constraints concerning the launch dates of certain missions in order to close some of these data gaps. This will be discussed in more detail in section 5.3.4.

6.3.2 Instrument selection

6.3.2.1 Configuration management: summary of rules used

The Decadal instrument selection problem considers the 39 instruments listed on Figure 61. Before presenting and discussing the results of the Decadal instrument selection case study, we remind the different types of rules that were used to obtain these results. According to the thesis objectives, some of these results are taken from the library of classes of SAPs, some are domain-specific but common to all case studies, some are domain-independent and SAP class-independent, and some are specific to the Decadal instrument selection problem.

The results presented in this section were obtained with the following set of rules:

Class of rule	Type of knowledge	Source of rule
Grammar and enumeration rules	Domain-independent, common to all down-selecting problems	Library-DsPs
Search heuristics	Domain-independent, common to all down-selecting problems	Library-DsPs
VASSAR-aggregation rules	Decadal-specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-requirement satisfaction rules	Decadal -specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-instrument capability rules	Decadal -specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Common to all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all down-selecting problems	See section 2.3.6

Table 41: Summary of rules utilized in the Decadal instrument selection case study

In addition to these rules, the following enumeration constraints were added to the Decadal instrument selection case study. Note that none of these rules are strictly necessary, as the algorithm is intelligent enough to locate these conflicts. The goal of these rules is rather to focus on the interesting regions of the tradespace, and thus accelerate the convergence of the search process.

Type of constraint	Instruments concerned	Justification
GROUP	SWOT_SAR SWOT_MWR SWOT_GPS	It doesn't make sense to select the MWR or the GPS, without the SAR

Table 42: Additional enumeration constraints added to the Decadal instrument selection case study

6.3.2.2 Results from the architectural tradespace exploration

The generic search algorithm complemented with the rules enumerated in Section 6.3.2.1 was initialized with a random population containing the reference architecture, and the algorithm was run for 10 iterations. The sixth generation obtained before down-selection is plotted in Figure 68 (not much variation was obtained after this iteration). In Figure 68, each diamond represents one Decadal selection architecture (i.e., a subset of instruments) in the cost-science space. The color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process. The following down-selection rules were applied:

- Normalized Science (computed as described in Section 4.4.1) > 0.67
- Lifecycle cost (computed as described in Section 4.4.2) < 4 FY00$B
- Normalized programmatic risk (computed as described in Section 4.4.3) < 0.1
- Utility (computed as described in Section 2.3.6.2 with weights 40% science, 40% cost, 20% risk, 0% fairness) > 0.5
- Pareto ranking (computed as described in Section 2.3.6.1) < 4

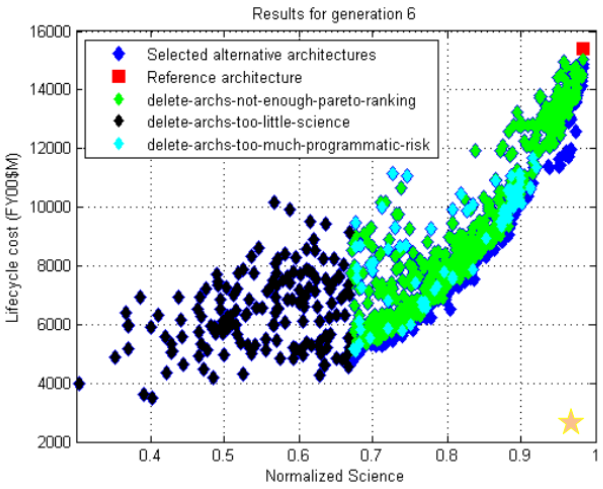


Figure 68: Population of Decadal selection architectures after 6 generations in the science-cost space

Most architectures that are not eliminated by the science cap are eliminated because of their poor Pareto rankings. Only a few architectures are eliminated due to programmatic risk considerations. The remaining architectures after down-selection are shown in Figure 69, together with the reference architecture and the utopia point.

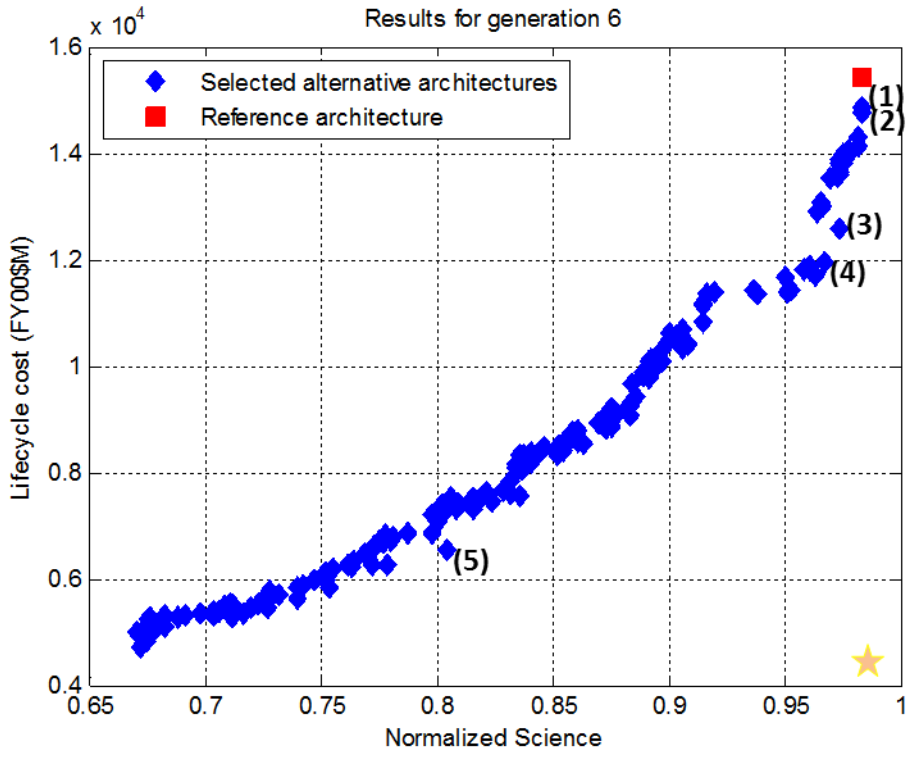


Figure 69: Fuzzy Pareto frontier of Decadal selection architecture after 6 generations

It is interesting to see that the upper part of this fuzzy Pareto frontier disappears as we introduce a more stringent down-selection rule in terms of the minimum utility compared to what is shown on Figure 69. This is so because for a utility that is computed as 40% cost, 40% science, and 2% risk, optimal utility is achieved in the region of the science-cost space where $science \approx 0.7$. As the ratio between the weights for science and cost is increased, the position of this optimal point naturally approaches the region of $science \approx 1.0$. This can be seen on Figure 70. On Figure 70, the figures are obtained with different weights for the utility functions, as follows: upper left: science = cost = 2*risk; upper right: science = 2*cost=2*risk; lower left: science = 3*cost = 2*risk; lower right: science = 4*cost = 2*risk. Note how the position of the optimal point in the science-utility space moves from 0.7 to almost 1.0.

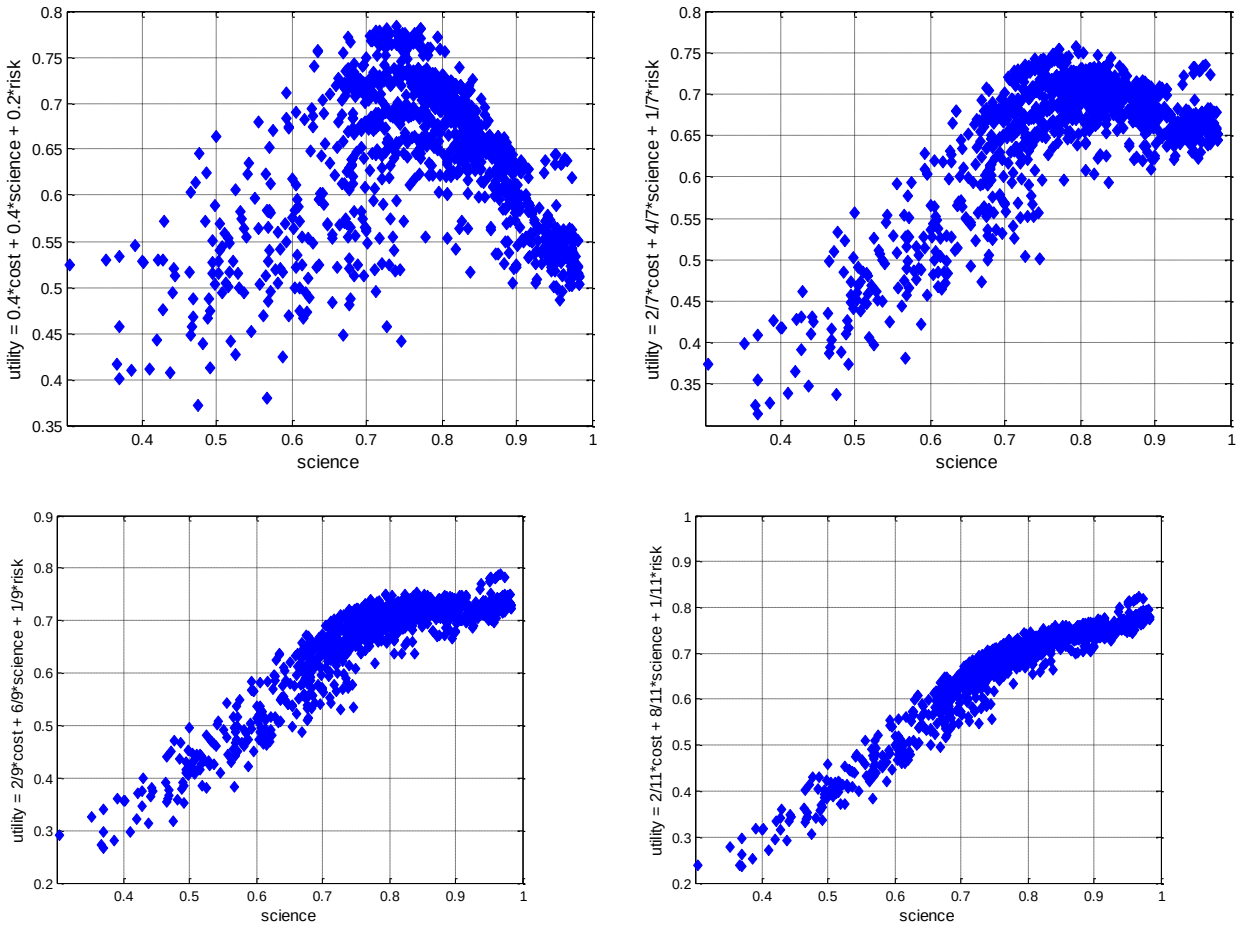


Figure 70: Science versus overall utility for Decadal selection architectures for various relative weights

Figure 69 shows that the reference architecture is dominated by two architectures that also achieve the highest possible science score at a slightly lower cost. These two architectures together with a few other ones are shown in Table 43.

Table 43 raises an issue that was discussed previously: that there might be some redundancy in the eight lidars proposed in the Decadal survey. In order to delve a little deeper in this piece of analysis, a full factorial enumeration of the architectures composed of all the possible subsets of these eight lidars was performed. The science and cost scores of these 255 architectures are shown in Figure 71.

Arch. id#	% science loss w.r.t. ref (ref science = 0.984)	% cost gain w.r.t. ref (ref cost = $15.4B)	Instruments deleted w.r.t. ref
1	0.0%	3.6%	ICE_LID
2	0.0%	4.3%	ICE_LID + ASC_GCR
3	1.0%	18.4%	ASC_GCR+ GACM_DIAL+ LIST_LID
4	1.6%	22.4%	GACM_MWSP + GACM_DIAL + LIST_LID
5	18.2%	57.6%	ACE_ORCA ACE_LID + CLARREO + DESD_LID + GACM (exc. VNIR) + GEO_WAIS + HYSP_TIR (not VNIR) + LIST + SMAP_MWR + SWOT + XOV_SAR (not real aperture) + 3D_NCLID (not coherent)

Table 43: Difference between the Decadal reference selection architecture and the top alternative architectures highlighted in Figure 69

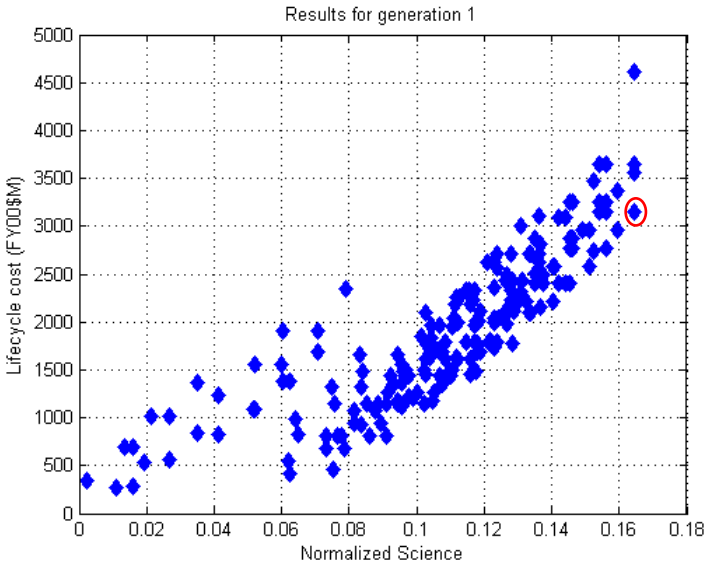


Figure 71: Science vs cost for all 256 combinations of lidars in the Decadal Survey (from 1 lidars to all 8 of them)

The least costly combination of lidars achieving the maximum science score is highlighted in Figure 71. This combination uses only six out of the eight initial lidars: the lidars on ICESAT-II and GACM are canceled. The maximum science score is still achieved through DESDYNI (for ice) and ACE (for ozone).

Another interesting piece of analysis is to look at the percentage of the instruments that are most frequently selected in the top architectures in Figure 69. The results of this piece of analysis are shown in Figure 72.

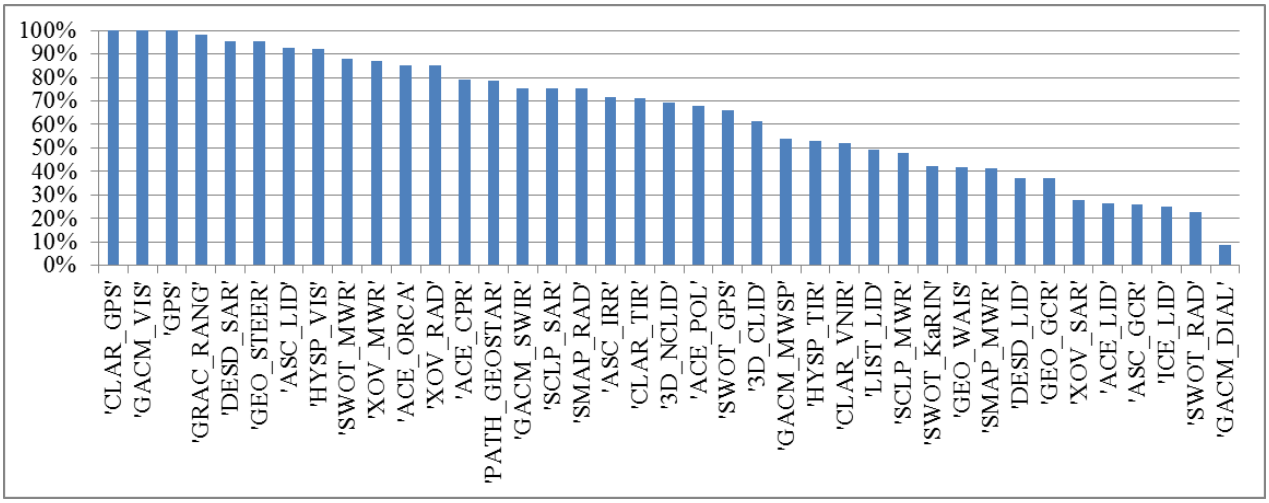


Figure 72: Percentage of top Decadal selection architectures carrying each instrument

Top instruments according to Figure 72 are the GPS radio occultation instruments, the VNIR spectrometer on GACM, the GRACE gravity instrument, the DESDYNI L-band SAR, the steerable imaging spectrometer on GEO-CAPE, the CO_2/O_2 DIAL on ASCENDS, and the VNIR part of the HYSPIRI mission. These instruments are present on 90% or more of the top architectures. Conversely, the lidar on GACM, the real aperture radiometer on SWOT, the lidar on ICESAT, and the gas correlation radiometer on ASCENDS are present on less than 30% of the top architectures. This would suggest that the former are the core of the Decadal survey instrument set, while the latter are candidates to constitute optimal descoping paths.

These results are discussed in more depth in the next section.

6.3.2.3 Discussion

Overall, the results for the Decadal instrument selection case study are consistent with the trends identified in the EOS case study: **instruments that satisfy many objectives from different stakeholders** (e.g., multi-purpose high resolution VNIR imagers, or weather instruments) **achieve higher scores in the model than more sophisticated instruments that focus on a few specific objectives**.

The model identified **some potential redundancies in the eight lidars** proposed by the NRC committee. Out of these eight lidars, two could potentially be deleted without noticeably affecting the science case, namely ICESAT-II and GACM. However, this conclusion is prematurely drawn by looking only at the instrument selection portion of the SAP. While the instrument packaging problem is unlikely to favor the inclusion of these two extra lidars, **the scheduling problem might justify this potential redundancy** in order to have a continuous lidar capability over a longer period of time.

In the case of ICESAT-II, it is particularly true that there is a desire to launch as soon as possible in order to close a potential gap left by the precursor GLAS instrument. One may think that this might in turn reverse the situation in make the lidar on DESDYNI appear somewhat redundant. However, the model identifies that this is not so, as the capabilities of the multi-beam DESDYNI lidar exceed the capabilities of ICESAT-II in particular for vegetation studies. Hence, adding the constraint of including the ICESAT-II lidar as a gap filler and cross-calibration instrument between ICESAT and DESDYNI results in an architecture that contains one more lidar than the optimal unconstrained one. A similar argument can be made with GACM and ACE. While the DIAL portions of the two lidars for ozone sounding may be redundant, spacing the two lidars in time may provide a longer data series.

Another potential redundancy identified by the model is in the manifest of two gas filter correlation radiometers for CO and CH4 monitoring. While one is in GEO and the other one in LEO and therefore somewhat complementary in terms of coverage and temporal resolution, there are other instruments in the suite that are capable of measuring these molecules with similar performance (e.g., the SWIR portion of GACM).

Similarly, the model identifies the partial redundancy between the SAR and the real aperture portions of SWOT. In this case, the main motivation to fly the real aperture altimeter is to provide a cross-calibration with the new wide swath altimeter technology.

Final recommendations about selection architecture for the Decadal case study are made later in this chapter, once the couplings with packaging and scheduling have been identified and understood.

6.3.3 Instrument packaging

The instrument packaging problem for the Decadal Survey looks at the Tier I missions and Tier II missions for which more information is available about the instruments. More precisely, all instruments of the following missions are considered: SMAP, ICESAT-II, CLARREO, and DESDYNI (all Tier I missions), and ASCENDS, HYSPIRI (Tier-II missions). This represents a total of 13 instruments, or 27 million architectures. Other Tier-II missions were left out of the analysis for several reasons:

- GEO-CAPE is a GEO mission and therefore not susceptible to merging with any LEO mission
- SWOT and XOVWM are altimetry/scatterometry missions with potentially very particular orbital requirements. While it would be interesting to look at packaging architectures within these two missions, it is far less interesting to look at combinations of these two missions with any of the other missions

- ACE is a late Tier-II mission. Packaging trade-offs between ACE instruments and the other Tier-I and Tier-II missions are very relevant. The ACE instruments were left out of the analysis in order to limit the size of the tradespace, because it was the Tier-II mission with the latest expected launch date. However, some preliminary results of including the ACE mission in the set are available and will be discussed.

6.3.3.1 Configuration management: summary of rules used

Before presenting and discussing the results of the Decadal instrument packaging problem, we remind the different types of rules that were used to obtain these results. All the results presented in this section were obtained with the set of rules on Table 44.

Class of rule	Type of knowledge	Source of rule
Grammar and enumeration rules	Domain-independent, common to all set partitioning problems	Library-PaPs
Search heuristics	Domain-independent, common to all set partitioning problems	Library-PaPs
VASSAR-aggregation rules	Decadal-specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-requirement satisfaction rules	Decadal -specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-instrument capability rules	Decadal -specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Common to all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all partitioning problems	Library-PaPs

Table 44: Summary of rules utilized in the Decadal instrument packaging case study

After a few generations, it also became clear that architectures with more than ten satellites, or more than eight instruments per satellite, are too costly and therefore do not need to be considered. Hence, rules were added to limit the maximum number of subsets in the partitions, and maximum number of elements in the subsets, as described when the class of partitioning problems was introduced in Section 9.1.2.1. No additional hard constraints were added for the Decadal packaging problem.

6.3.3.2 Results from the architectural tradespace exploration

The generic search algorithm complemented with the rules enumerated in Section 6.3.3.1 was initialized with a random population containing the reference architecture, and the algorithm was run for 30 iterations. The last generation obtained before down-selection is plotted in Figure 73.

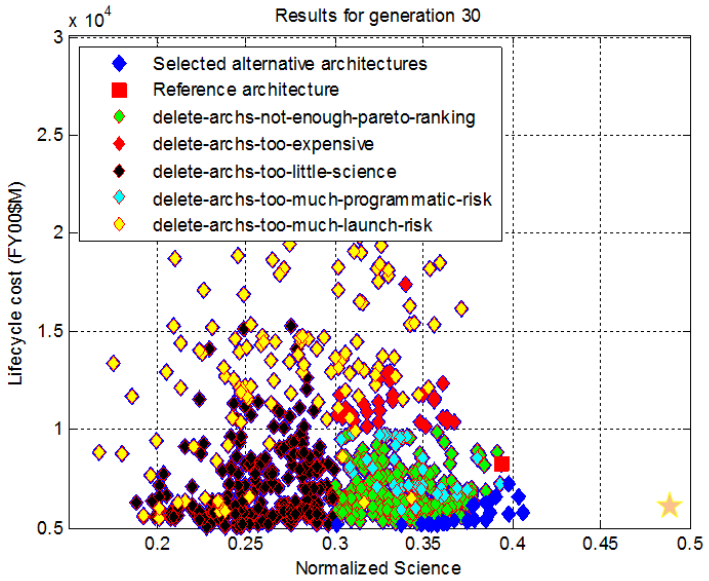


Figure 73: Population of Decadal packaging architectures after 30 generations in the science-cost space

In Figure 73, each diamond represents one Decadal packaging architecture (i.e., a partition of instruments) in the cost-science space. Again, the color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process. The following down-selection rules were applied:

- Normalized Science (computed as described in Section 4.4.1) > 0.30
- Lifecycle cost (computed as described in Section 4.4.2) < 10 FY00$B
- Normalized programmatic risk (computed as described in Section 4.4.3) < 0.5
- Normalized launch risk (computed as described in Section 4.4.4) < 0.5
- Utility (50% science, 35%cost, 7.5% programmatic risk, 7.5% launch risk) > 0.6
- Pareto ranking (computed as described in Section 2.3.6.1) < 4

The remaining architectures after down-selection are shown in Figure 74, together with the reference architecture and the utopia point.

Three alternative architectures were identified on Figure 74 that also achieve the highest scientific score, and have costs that are indistinguishable from the reference cost once the uncertainty in modeling parameters is taken into account. These architectures are highlighted on the fuzzy Pareto frontier, and described in more detail in Table 45.

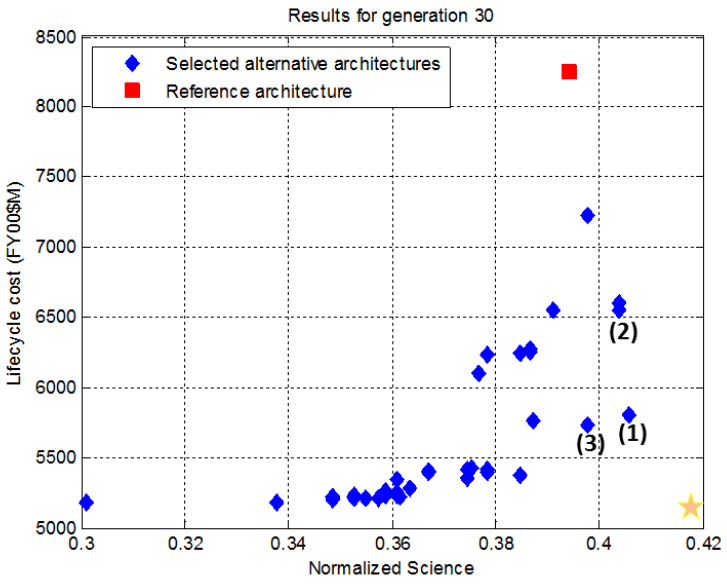


Figure 74: Fuzzy Pareto frontier of Decadal packaging architecture after 30 generations

Arch	#sats	#instruments per satellite	Allocation
1	7	3 3 1 1 2 1 2	ASC_LID ASC_GCR ASC_IRR CLAR_TIR CLAR_VNIR CLAR_GPS **DESD_SAR** **DESD_LID** HYSP_TIR HYSP_VIS ICE_LID SMAP_RAD SMAP_MWR
2	6	4 2 2 1 2 2	ASC_LID ASC_GCR ASC_IRR **CLAR_TIR** **CLAR_VNIR** ICE_LID **CLAR_GPS** DESD_SAR **DESD_LID** HYSP_TIR HYSP_VIS SMAP_RAD SMAP_MWR
3	6	4 3 2 1 1 2	ASC_LID ASC_GCR ASC_IRR **CLAR_GPS** **CLAR_TIR** HYSP_TIR HYSP_VIS **CLAR_VNIR** DESD_SAR **DESD_LID** ICE_LID SMAP_RAD SMAP_MWR
ref	6	3 3 2 2 1 2	ASC_LID ASC_GCR ASC_IRR CLAR_TIR CLAR_VNIR CLAR_GPS DESD_SAR DESD_LID HYSP_TIR HYSP_VIS ICE_LID SMAP_RAD SMAP_MWR

Table 45: Difference between the Decadal reference packaging architecture and the top alternative architectures

Several interesting comments can be made by analyzing Table 45. First, there are architectures that achieve both lower cost and higher science than the reference architecture. We now try to use the explanation facility to understand why the reference architecture does not achieve the best scores. Figure 75 provides a pictorial representation of the reference architecture and the three alternative architectures. For each satellite, mass, orbit, lifecycle cost, and launch vehicle class are shown, in addition to the complexity penalties defined in 0. The legend for the complexity penalties is as follows: Green means inactive penalty, red means active penalty; M stands for mechanisms penalty, T stands for Thermal penalty, D stands for data rate penalty, A stands for ADCS penalty, S stands for scanning penalty, and E stands for EMC penalty.

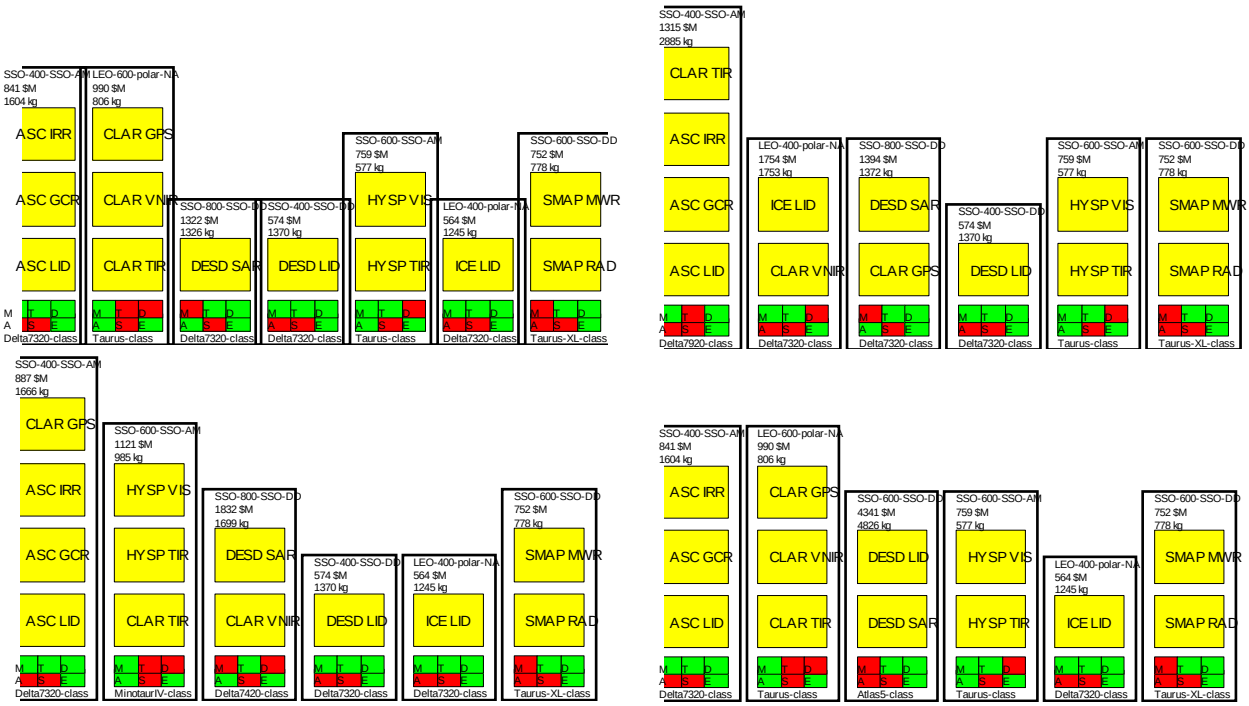


Figure 75: Detailed comparison between reference architecture (bottom right) and three alternative packaging architectures for the Decadal case study

In terms of science, Alternative architecture 1 (top left on Figure 75) achieves a higher science score than the reference architecture because by separating the SAR and lidar portions of DESDYNI, the science output of both instruments increases. In the reference architecture, the SAR and the lidar fly together in a compromise dawn-dusk SSO at 600km. In Alternative Architecture 1, the SAR improves coverage by flying at a higher altitude (800km), and the lidar improves spatial sampling by flying at a lower orbit. The explanation facility shows that the improvement in coverage of the SAR provides benefits in terms of snow cover, hydrocarbon monitoring, and surface deformation data products. Flying the lidar lower has benefits in terms of vegetation data products.

The other alternative architectures achieve lower science than the reference because the CLARREO instruments fly as opportunity payloads on other spacecraft that do not meet their orbital requirements. In particular, requirement rules for Earth radiation budget require a true polar, non SSO orbit for the CLARREO instruments in order to have true global coverage. When this requirement is not met, the science output of the CLARREO instruments decreases, which may or may not be compensated by the cost savings corresponding to flying the CLARREO instruments as opportunity payloads.

In terms of cost, differences between the architectures on Figure 75 are driven by:

- **Whether the DESDYNI lidar is flown at 600km or 400km**. Flying the lidar at 600km requires is much more costly due to the increase in power requirements to maintain SNR.
- **Whether an Atlas-5 launch is required**. The reference architecture is the only one using the Atlas-5 launch vehicle, which results in higher launch costs than the alternative architectures by 7.5% ($20-$25M).
- **Whether large satellites are used at low orbits.** Satellites flying at 400km suffer from high atmospheric drag that may drive the design of the ADCS. The larger and more massive the satellite is, the more stringent the requirements on the ADCS. The model identifies that this situation can be avoided by using smaller satellites at 400km.
- **Whether the current NASA ground stations are capable of dealing with the required spacecraft data rate.** The model penalizes spacecraft that require downlink speeds higher than 500Mbps because that is beyond the capabilities of the current NASA ground station.

It is interesting to note that alternative architectures on Figure 75 have either 6 or 7 satellites. The question arises whether this is a generalizable statement. In order to answer this question, Figure 73 was redrawn with the colors of the diamonds now representing the number of satellites in the architecture. The resulting chart is provided in Figure 76.

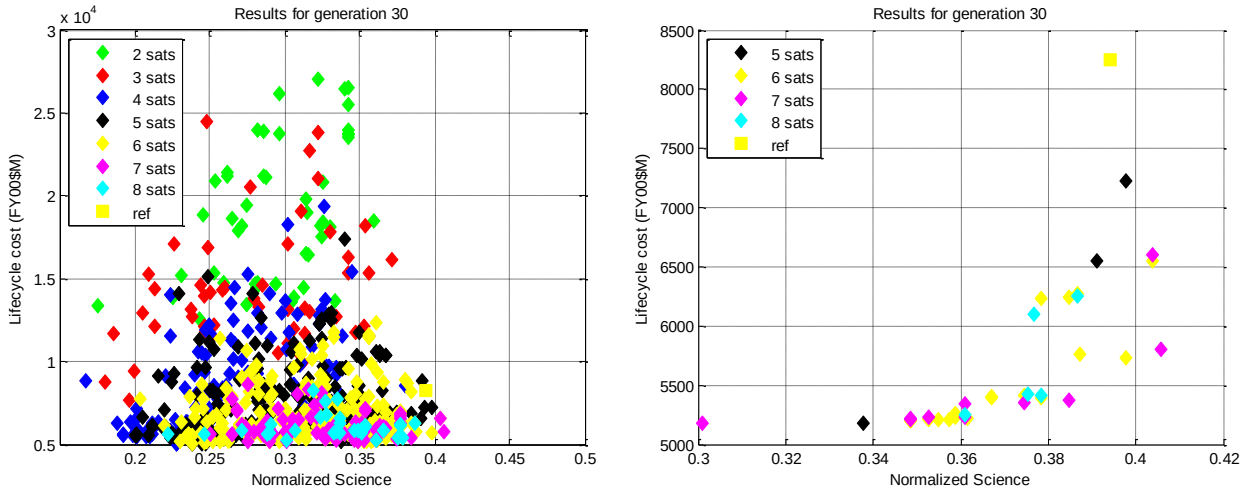


Figure 76: Packaging architectures in the cost-science space by number of satellites

Several consequences can be drawn from Figure 76:

- All **architectures with 2, 3, and 4 satellites were eliminated** by the tool during the process of searching through the tradespace because they were heavily dominated
- **Architectures with 5, 6, and 7 satellites are capable of achieving the highest science scores**. No architectures with less than 5 satellites are capable of achieving the highest science scores.
- **The less costly architectures are 5-8 satellite architectures**.

Another interesting insight of Figure 74 is the vertical stratification of architectures that was already seen on the EOS case study. This stratification appears because there are many different ways of achieving a given level of synergy. The stratification is more noticeable when cross-registration of instruments on different spacecraft on the same orbit is allowed, simulating a train configuration. This stratification suggests that an isoperformance (iso-science) approach that shows the architectures achieving the maximum science score in the cost-risk space may bring additional insight. Such figure is shown on Figure 77. The positions of the three architectures from Figure 75 on the first cost-risk iso-science Pareto frontier are shown.

One can conclude that at this stage, alternative architecture #1 seems to be the most promising one, as it achieves the highest science score in the most efficient way, and it is also non-dominated in the cost-risk space.

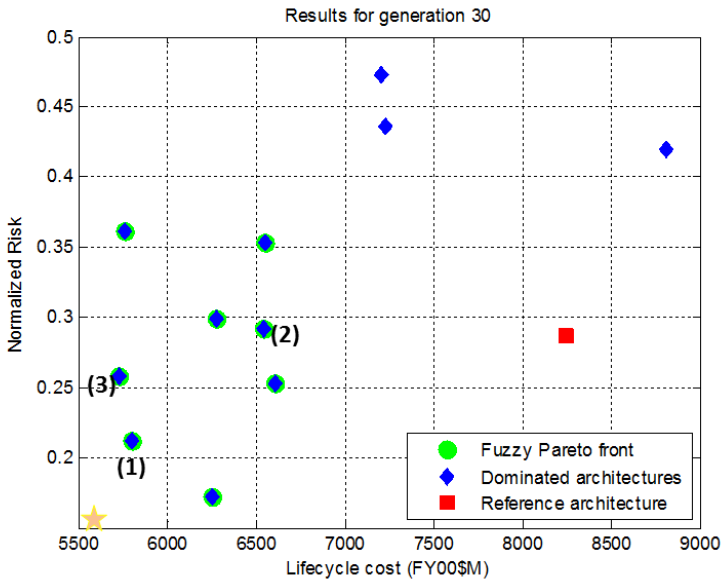


Figure 77: Iso-science, cost-risk space in the Decadal packaging problem

6.3.3.3 Discussion

The main consistent findings of the Decadal packaging case study concern the DESDYNI mission. Results show that **it is neither necessary nor desirable that the radar and lidar portions of DESDYNI share a common platform**. From the science perspective, while the cross-registration of lidar and radar data is potentially harder if the two instruments do not share a platform, the loss in performance due to very different orbital requirements more than compensates for these facts. From the engineering perspective, the two instruments are large and very resource consuming, and therefore putting them on the same spacecraft results in large, expensive satellites.

Second, the option to fly the instruments in the CLARREO mission as opportunity payloads instead of as a stand-alone mission should be considered. These instruments don't have strong orbital requirements, and are good "companions" for other payloads, as they are unlikely to induce any problems on the bus, or drive the design of the bus (relatively high TRL, low mass and low power instruments, albeit high data rate for the VNIR portion). In particular, the ICESAT-II mission – or any other cryospheric mission with a true polar orbit – could be a host spacecraft for one of the instruments. While this option is desirable from the cost perspective, it potentially has negative consequences on the science output of the CLARREO instruments. It also has implications in terms of launch risk, as more assets are put at risk per launch.

The best alternative architecture identified at this stage appears to be very similar to the reference architecture, with the DESDYNI mission split in its SAR and lidar portions.

6.3.4 Mission scheduling

The mission scheduling problem for the Decadal Survey considers all 17 missions proposed in the report. The yearly budget is assumed to be $500M from 2010 up to 2050 if necessary. Mission cost estimates are the ones shown on Figure 60.

6.3.4.1 Configuration management: summary of rules used

As it was done in the instrument selection and packaging cases, before presenting and discussing the results of the Decadal mission scheduling problem, the different types of rules that were used to obtain these results are reminded.

All the results presented in this section were obtained with the set of rules provided in Table 46.

Class of rule	Type of knowledge	Source of rule
Grammar and enumeration rules	Domain-independent, common to all set permuting problems	Library-PePs
Search heuristics	Domain-independent, common to all set permuting problems	Library-PePs
VASSAR-aggregation rules	Decadal-specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-requirement satisfaction rules	Decadal-specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-instrument capability rules	Decadal-specific, common to selection, packaging, and scheduling	See Section 6.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Common to all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all permuting problems	Library-PePs

Table 46: Summary of rules utilized in the Decadal mission scheduling case study

Since, unlike the EOS case study, the purpose of the Decadal survey case study is not to validate but to provide recommendations, no additional rules were added. This means that the Decadal mission scheduling case study was run with the configuration by default for any mission scheduling problem (except for the down-selection rules which were naturally tailored as the search through the tradespace advanced).

6.3.4.2 Results from the architectural tradespace exploration

The generic search algorithm complemented with the rules enumerated in Section 6.3.4.1 was initialized with a random population containing the reference architecture, and the algorithm was run for ten iterations.

The last generation obtained before down-selection is plotted in Figure 78. In Figure 78, each diamond represents one Decadal scheduling architecture (i.e., a sequence of missions) in the discounted value-data continuity space.

Again, the color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process. The following down-selection rules were applied:

- Normalized Data continuity score (computed as described in Section 4.4.7) > 0.97
- Normalized discounted value score (computed as described in Section 4.4.5) > 0.97
- No down-selection constraint was added using fairness, as it is considered a non-critical metric
- Utility (70% data continuity + 15% discounted value + 15% fairness) > 0.6
- Pareto ranking (computed as described in Section 2.3.6.1) < 4

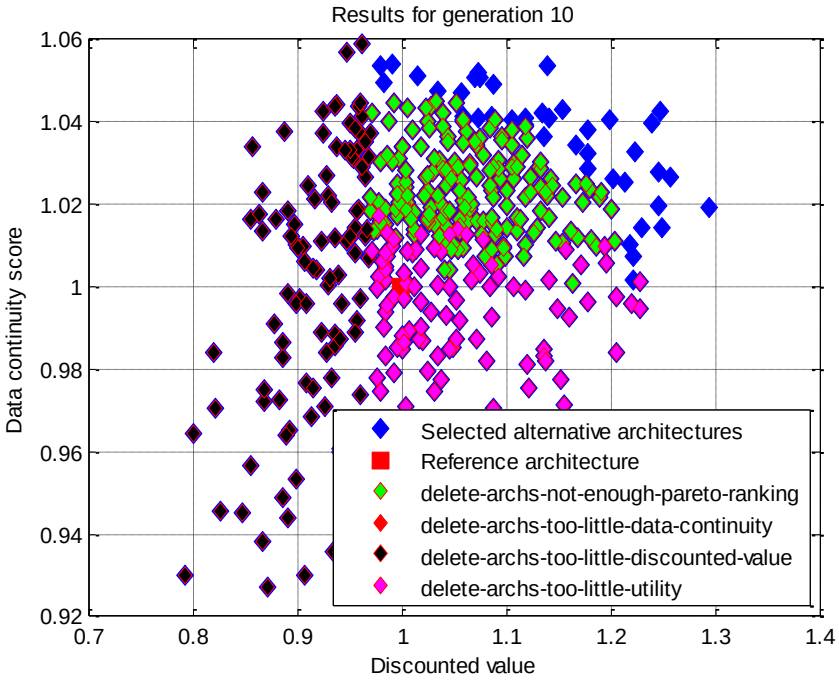


Figure 78: Population of Decadal mission scheduling architectures after 10 generations in the science-cost space

The remaining architectures after down-selection are shown in Figure 79, together with the reference architecture and the utopia point.

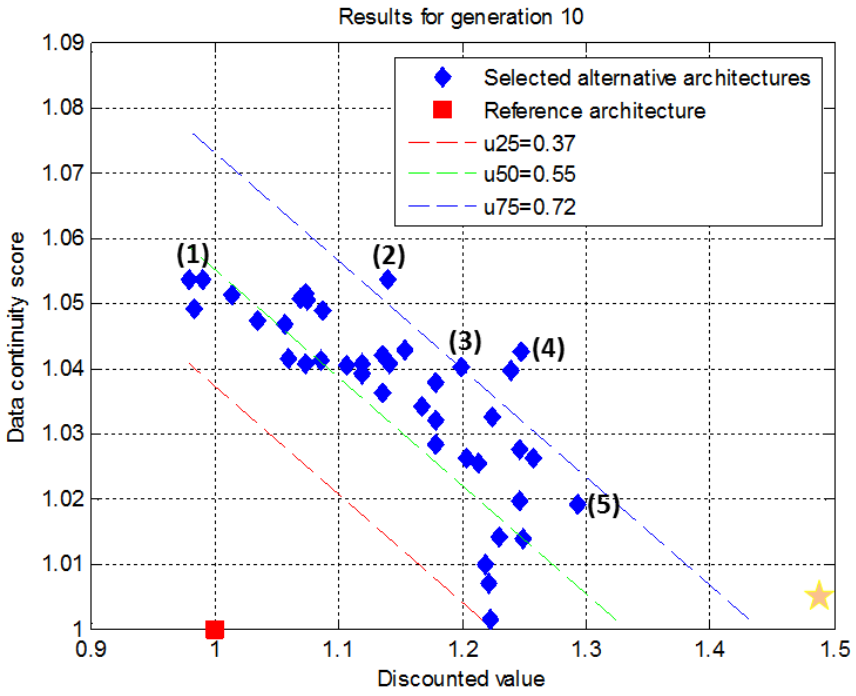


Figure 79: Fuzzy Pareto frontier of Decadal scheduling architecture tradespace after 10 generations

The first thing that we notice from Figure 79 is that there a many architectures that dominate the reference architecture, i.e., they achieve a higher discounted benefit while covering more and more important data gaps. In particular, five alternative architectures were identified on Figure 79. These architectures are highlighted on the fuzzy Pareto frontier, and described in more detail in Table 47.

Arch	Sequence of missions
1	3DWINDS SMAP ASCENDS GPSRO CLARREO SWOT XOVWM GEO-CAPE PATH HYSPIRI GACM DESDYNI SCLP LIST ACE ICESAT-II GRACE-II
2	GPSRO SCLP HYSPIRI CLARREO GRACE-II GEO-CAPE SWOT PATH XOVWM GACM ACE ICESAT-II ASCENDS SMAP LIST DESDYNI 3DWINDS
3	LIST CLARREO HYSPIRI XOVWM ASCENDS GACM GRACE-II SWOT ICESAT-II ACE GPSRO GEO-CAPE SCLP SMAP PATH DESDYNI 3DWINDS
4	LIST SMAP HYSPIRI CLARREO GPSRO XOVWM GACM PATH SWOT GRACE-II SCLP GEO-CAPE DESDYNI ASCENDS ACE 3DWINDS ICESAT-II
5	HYSPIRI LIST GRACE-II SMAP SWOT SCLP ICESAT-II GPSRO GACM DESDYNI ASCENDS XOVWM 3DWINDS CLARREO PATH GEO-CAPE ACE
ref	SMAP ICESAT-II DESDYNI CLARREO GPSRO ASCENDS ACE GEO-CAPE SWOT HYSPIRI XOVWM GACM GRACE-II LIST PATH SCLP 3DWINDS

Table 47: Difference between the Decadal reference scheduling architecture and the top alternative architectures

The statistics of the launch dates of the top Decadal scheduling architectures are shown in Figure 80, in order to analyze the differences between alternative architectures and the reference architecture in the architectural domain - as opposed to the metrics domain.

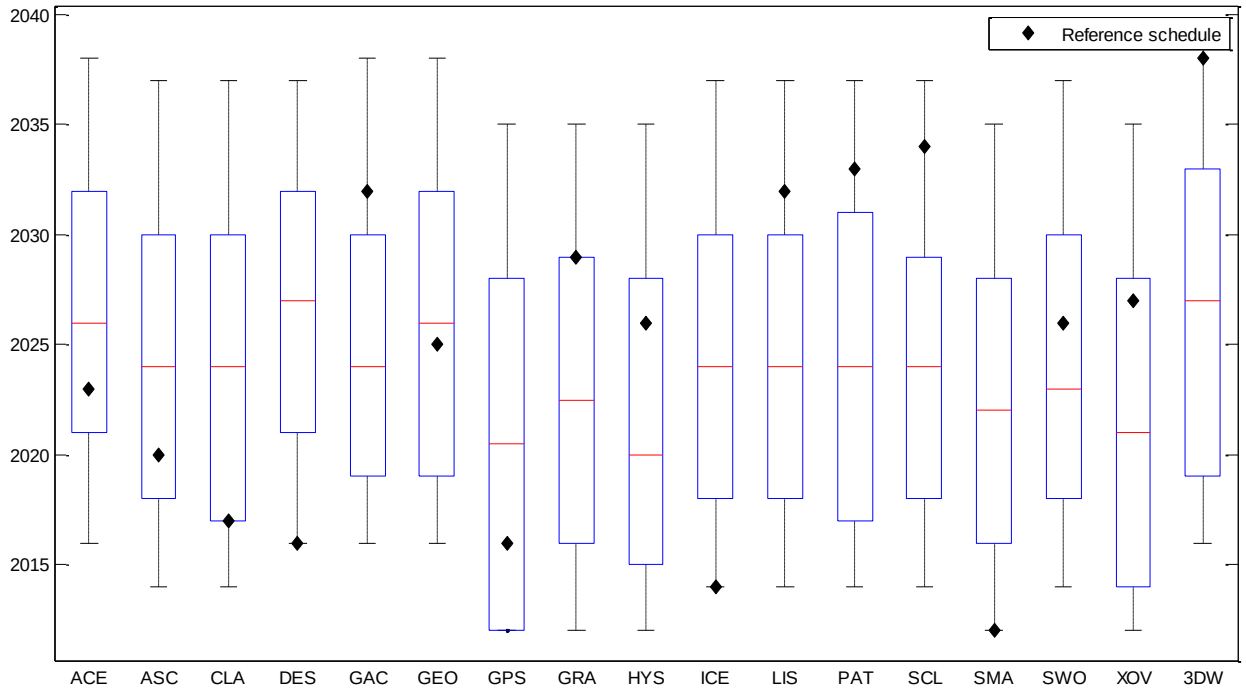


Figure 80: Statistics of launch dates for top Decadal scheduling architectures

The reference architecture is shown with black diamonds. Figure 80 reveals a "lack of signal" in this particular view. All missions have a median launch date between 2020 and 2027 and a 25th – 75th percentile range on the order of a decade. Since these are the statistics over the entire fuzzy Pareto frontier, this might indicate a bi-modal tradespace where architectures that are good in data continuity are different from architectures that are good in terms of discounted value, but the differences average out when one considers the entire tradespace.

In order to check whether this is the case, the statistics are plot separately for top architectures in data continuity and top architectures in discounted value. The top 10% architectures in data continuity, top 10% architectures on discounted value, and the statistics of launch dates for each of these two groups are shown on Figure 81. Figure 81 provides more insight than the previous figure because it is possible to compare the statistics of the launch dates across the two groups.

For example, the DESDYNI mission is launched later in the alternative architectures than in the reference architecture, and more so when data continuity is considered. GACM and HYSPIRI tend to be launched in the first tier rather than in tier II or II as it is in the current plan. The signal is stronger when looking at discounted value. **Virtually all the architectures in the top 10% architectures in discounted value – except for outliers - have HYSPIRI launched before 2015**.

GRACE-II and SMAP are launched noticeably earlier is emphasis is put on discounted value rather than data continuity. On the other hand, if data continuity is favored, XOVWM, GEO-CAPE, ASCENDS, and to a lesser extent SCLP are launched earlier to close potential data gaps in scatterometry, atmospheric chemistry (especially CO2), and snow cover.

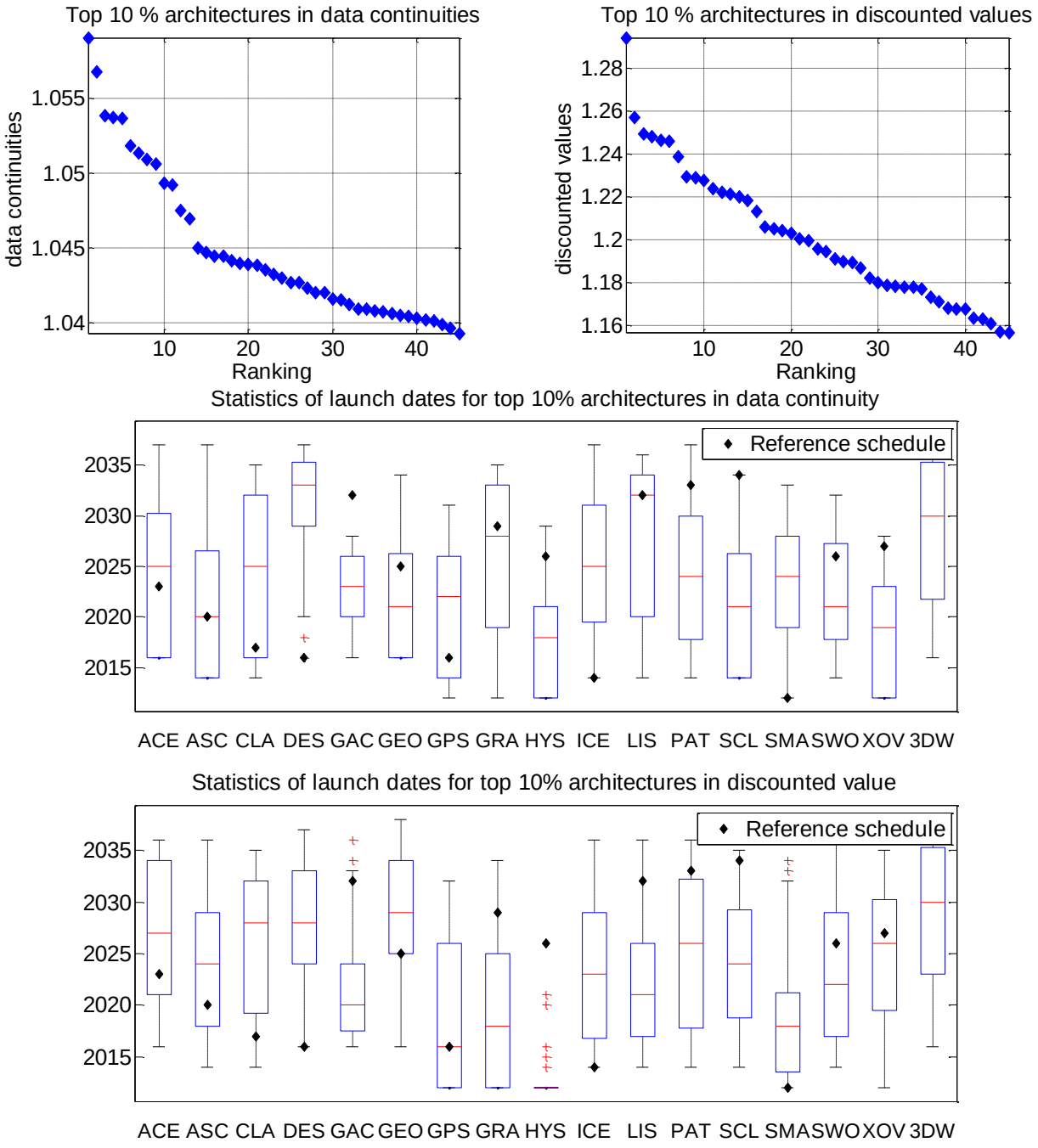


Figure 81: Launch dates for top 10% architectures in data continuity and discounted value

6.3.4.3 Discussion

As seen in the EOS case study, the Decadal mission scheduling problem has proved to be harder to solve than the instrument selection problem but simpler than the instrument packaging problem.

This scheduling problem has shown that GPSRO, GACM and HYSPIRI should be top priorities to be launched in the next years, regardless of the relative preferences of decision makers between data continuity and discounted value.

Decision makers who give a higher weight to data continuity should then prioritize the scatterometry and atmospheric chemistry missions (namely XOVWM, ASCENDS, and GEO-CAPE) as well as the snow and cold regions SAR (SCLP).

On the other hand, decision makers who give a higher weight to discounted value (i.e., launch cost-effective missions first), should prioritize GRACE-II and SMAP.

This version of the mission scheduling problem is a completely unconstrained one. This means that **technology readiness was not taken into consideration**. The lines of funding for missions and basic technology are currently separate in the NASA Earth science budget, and the model currently lacks the capability of simulating these two processes and their interactions. This is one of the top priorities for future work.

6.4 Conclusion

This case study has provided insight into the architecture of the Decadal Survey, which is summarized below in the form of three key recommendations:

- The NRC committee recommended the development of 8 lidars for the next decade. While most of them are very different in technology and application, some **potential redundancies have been identified by the model in two areas: laser altimeters for clouds and aerosol properties, and differential absorption lidars for ozone sounding**. These redundancies are identified with the static instrument selection model. Some of these redundancies partially disappear when time is taken into account in the scheduling problem.
- The DESDYNI reference mission flies a large SAR and a lidar on a common platform at a compromise altitude of 600km, as per the reference architecture. The model identifies that **it is not desirable to fly the radar and the lidar on the same platform, not only because of cost considerations, but also for scientific issues.** The science of the SAR improves at 800km because of improved coverage, and the cost and science and of the lidar both improve at 400km because of reduced power requirements and better spatial sampling at this lower altitude.

- The **launch of GPSRO, HYSPIRI, and GACM should be prioritized** because these three missions are all highly cost-effective and have high potential to cover data gaps.

7 Case Study 3: Iridium Next

7.1 Context and goals for the case study

On June 2 2010, Iridium LLC announced a comprehensive plan for their next generation constellation of communication satellites in LEO called Iridium NEXT (Castro, Knowles, & Goodman, 2010). This plan included the development, fabrication, and deployment of 66 new satellites plus spares to replace the current aging constellation in their 800km near-polar orbits. As part of the agreement signed with the prime contractor Thales, Iridium announced the largest hosted payload opportunity ever proposed by a commercial operator. Iridium and Thales offered a nanosatellite class slot (50kg, 50W, 50 x 50 x 70 cm, 100kbps) on each of the 66+ satellites. The ICD for the potential hosted payloads is available on-line[21]. Payloads that are capable of meeting these interface requirements by the time of the CDR of Iridium NEXT (i.e., 1Q 2013) will be eligible to be launched in the first launch in 2015.

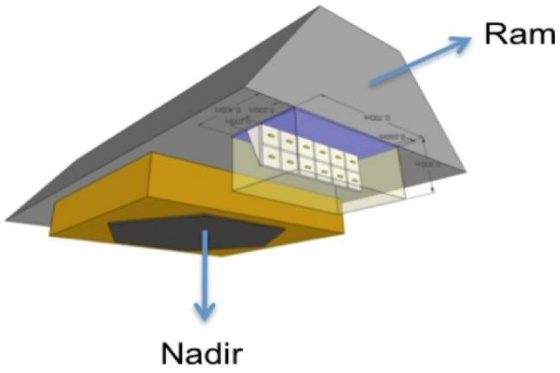


Figure 82: An Iridium NEXT satellite showing the full hosted payload (image credit: http://geoscan.jhuapl.edu/)

Immediately after this press release, several companies and research laboratories started studying the potential of this unprecedented opportunity for Earth observation. Indeed, a massive constellation of satellites in LEO promised to break one of the traditional trade-offs in satellite-based Earth observation: that of coverage vs temporal resolution vs spatial resolution.

Traditional Earth observing systems are single-satellite missions in either GEO or LEO orbits. GEO satellites guarantee very good temporal resolutions on the order of 15min or less, but they cannot guarantee global coverage, and they suffer from poor spatial resolution due to the high altitude. On the other hand, LEO satellites in polar or SSO have global coverage and good spatial resolution, at the price of a very much reduced temporal resolution ranging from 12h to several days or even weeks.

[21] www.iridium.com/DownloadAttachment.aspx?attachmentID=1224

The Iridium NEXT hosted payload program can overcome this trade-off with a constellation of satellites in LEO, thus offering global coverage, GEO-class temporal resolution, and LEO-class spatial resolution, at a fraction of the price of traditional architectures. This represents a once-in-a-lifetime opportunity that can lead to new scientific discoveries. It is for many the "new and better way" of doing satellite-based Earth observation.

The first agreement between Iridium and a third party concerning the hosted payload program was announced by both Iridium LLC and Orbital sciences on February 3, 2011 (Castro & Rhodes, 2011). By this agreement, Orbital sciences committed to acquire 20% of the hosted payload capacity for up to $100M, although it was unclear at that time whether that implied 20% each slot on the 66+ satellites, or the complete slot on 20% of the satellites. Orbital Sciences was also selected by the prime contractor Thales to be the satellite and hosted payload integrator. SpaceX was selected as primary launch provider.

In parallel, the Applied Physics Lab at the John Hopkins University started conversations with Iridium and the NSF in order to study a concept for an optimal allocation of these slots. Their concept is based on the fractionation of each individual satellite slot into smaller units that can then be assigned to two different types of sensors: system sensors, and hosted sensors.

System sensors are the same across all 66+ satellites and are meant to be mature instruments (TRL>6) that respond to key scientific and societal needs that require global coverage. On the other hand, hosted sensors are PI-led, high-risk, high-return instruments that can be different across satellites and that are meant to activate a broader variety of stakeholders including universities. The goal of GEOscan is to achieve a risk balanced program with this mix of system and hosted sensors.

A workshop was organized by APL on March 2011 under NSF sponsorship to gather instrument recommendations from universities and organizations for GeoSCAN. Several dozens of instruments were proposed during the workshop. After months of analysis by APL and their collaborators, including Draper Laboratory for the systems engineering effort, the preliminary instrument selection was revealed. Six instruments were preselected to be system sensors:

- A **GNSS receiver in occultation mode**, for atmospheric temperature and humidity sounding, as well as ionospheric measurements of total electron content.
- A **microbolometer** to measure outgoing SW and LW radiation, for radiation budget studies.
- A set of **accelerometers**, for precise mapping of the Earth's gravity field.
- A **UV, VNIR and SWIR spectrometer**, potentially for atmospheric chemistry, biology, and vegetation measurements.

- A **VNIR imager** for multi-purpose land and ocean imagery, as well as cloud mask input to other instruments.
- A **dosimeter**, for space weather measurements of different types of radiation.

This synergistic set of instruments will provide data sets that will satisfy several needs of many disciplines of the Earth sciences, at a fraction of the price of a traditional Earth observing mission. Hence the inevitable question of the comparison in terms of cost-effectiveness of GeoSCAN with the traditional architecture, namely the missions proposed in the Decadal Survey.

More precisely, the goals of this case study are listed below:

- To analyze the performance of the GeoSCAN instrument suite, both as individual instruments and as a synergistic payload.
- To compare its cost-effectiveness with that of the Decadal Survey
- To study the system sensor selection problem and either confirm or disagree with the set of pre-selected instruments.
- To generate recommendations for hosted sensors based on instrument selection, packaging and scheduling considerations.

7.2 Case study specific rules

7.2.1 Aggregation and requirement satisfaction rules

The GeoSCAN program is an opportunity payload program and therefore it does not have a predefined set of precise scientific objectives. Instead, the GeoSCAN program has broad focus areas of interest such as climate change, disaster monitoring, or weather prediction. In order to conduct a quantitative case study, precise objectives and measurement requirements are required. Thus, **after conversations with the main protagonists of the GEOscan instrument selection, it was decided to take the set of objectives and measurement requirements from the Decadal Survey**.

This is indeed the most recent exhaustive set of measurement requirements available in the literature that covers the whole spectrum of disciplines of the Earth sciences, with the exception of space weather. Moreover, taking the Decadal set of objectives provides a common frame of reference to compare the Decadal architecture with GeoSCAN in terms of cost-effectiveness. It would be unfair to compare them using different sets of requirements.

Therefore, value aggregation rules and requirement satisfaction rules for the GeoSCAN case study are the ones described in Sections 6.2.1 and 6.2.2.

Note that this set of requirements may not be the best to distinguish between different GeoSCAN architectures, because the differences across architectures are likely to be in terms of coverage, temporal resolution, and variable measurement error, three variables that were almost constant across Decadal architectures - compared to the GEOscan case study-, and thus for which there were few scientific requirements. This suggests that this set of scientific requirements would not be useful to select how many instruments of each kind are necessary to achieve a certain level of science. That said, this is not one of the goals of the case study, as it has already been set that the 66 satellites will carry all the system sensors.

7.2.2 Instrument Capabilities

Since there are only a handful of instruments, instrument capability rules can be seen in detail. The summary of the instrument characteristics (i.e., mass, power, data rate, dimensions) is provided in Table 48.

Table 48: Summary of instrument characteristics for the GeoSCAN case study

Instrument	Mass (kg)	Power (avg/peak, W)	Data rate (kbps)	Dimensions (cm x cm x cm)	Cost per s/c ($k)	Dev cost ($k)
Microbolometer	0.6	0.3/5.0	1.0[22]	10 x 9.0 x 10	110	1000
GPS RO	0.2	1.5/1.7	2.0	8.0 x 13 x 1.3	85	0
Accelerometer	0.2	1.3/1.3	0.7	3.8 x 3.8 x 3.8	100	8000
Imager	0.3	0.6/0.6	4.2	4.0 x 4.0 x 4.0[23]	100	1000
Spectrometer	0.3	0.9/1.3	3.3	3.9 x 9.8 x 10.6	60	0
Dosimeter	0.02	0.3/0.4	0.1	3.6 x 3.6 x 0.1	6	0

The measurement capabilities of these instruments are reviewed one by one in the next paragraphs. The first system sensor is the microbolometer. It is primarily a radiation budget instrument. The microbolometer can measure both total radiation (SW+LW, $0.2 - 200\mu m$) and SW radiation ($0.2\ to\ 5\mu m$) with an accuracy better than 0.5W/m^2. It has a wide field-of-view of 124deg, which leads to a revisit time < 1.5min for all latitudes except the poles, and it has an on-orbit calibration capability. This very good temporal resolution provides a large population of samples that can be averaged out in order to reduce measurement variability due to noise.

[22] The data rate for the microbolometer was unknown and a value of 1.0kbps was assumed based on the other instruments in the set

[23] The dimensions of the imager were unknown and it was assumed that it measures 4cm x 4cm x 4cm based on the dimensions of the other instruments and on Figure 92.

The following rule asserts the corresponding measurements when a microbolometer is manifested in the GeoSCAN case study:

```
(define-rule CAPABILITIES::BOLOMETER-measurements
   "Define measurement capabilities of the GeoSCAN microbolometer"
IF there is a Manifested-instrument (with Id ?id) and (Name LORENTZ_ERB)
   => (THEN)

ASSERT Measurement (Parameter "1.9.3 Spectrally resolved SW radiance -0.3-2um-") (Id
LORENTZ_ERB1)))
ASSERT Measurement (Parameter "1.9.2 Spectrally resolved IR radiance -200-2000cm-1-")
(Id LORENTZ_ERB2)))
ASSERT cross-registered (measurements LORENTZ_ERB1 LORENTZ_ERB2) (with degree-of-
cross-registration instrument) (on platform ?id)))
```

Code 26: Capability rule for the GeoSCAN microbolometer (a.k.a. LORENTZ ERB)

The second system sensor is the GNSS radio occultation receiver, or GPSRO. The GPSRO measures the difference in Doppler shift between two GPS signals: one received from a reference, non-occulting GPS satellite, and one received from an occulting GPS satellite. This difference in Doppler shift comes from the difference in path length of the atmosphere that the two waves traverse. Knowing the real positions of the two GPS satellites and the GPS receiver, it is possible to infer a bending angle of the signal coming from the occulting GPS, and therefore the refractivity of the atmosphere. From refractivity, one can infer several atmospheric parameters, namely total electron content in the ionosphere, temperature or pressure, and humidity. It is expected that a GPSRO on an Iridium NEXT satellite provides around 300 occultations per day, which for a five year lifetime mission, results in a total number of occultations that exceeds the number provided by current missions such as COSMIC, GRACE, CHAMP, or TerraSAR-X. The following rule asserts the corresponding measurements when a GPSRO is manifested in the GeoSCAN case study:

```
(define-rule CAPABILITIES::GPSRO-measurements
   "Define measurement capabilities of the GeoSCAN microbolometer"
IF there is a Manifested-instrument (with Id ?id) and (Name CTECS)
   => (THEN)

ASSERT Measurement (Parameter "1.3.1 Atmospheric humidity -indirect-") (Id CTECS1)))
ASSERT Measurement (Parameter "1.2.1 Atmospheric temperature fields") (Id CTECS2)))
ASSERT Measurement (Parameter "1.3.3 GPS radio occultation") (Id CTECS3)))
ASSERT Measurement (Parameter "A8.Total electron content in ionosphere") (Id
CTECS4)))
ASSERT Measurement (Parameter "1.8.1 H2O") (Id CTECS5)))
ASSERT Measurement (Parameter "1.3.4 Atmospheric pressure") (Id CTECS6)))
ASSERT cross-registered (measurements CTECS1 CTECS2 CTECS3 CTECS4 CTECS5 CTECS6)
(with degree-of-cross-registration instrument) (on platform ?id)))
```

Code 27: Capability rule for the GeoSCAN GPSRO (a.k.a. CTECS)

The third system sensor is the gravity sensor. The gravity sensor is a 3-axis accelerometer that will provide non-gravitational accelerations (i.e., the result of forces such as drag) with a sensitivity of $100\, ng/\sqrt{(Hz)}$ or better in the 0.0001-0.001Hz band. In combination with GPS information providing precise relative position and velocity of the spacecraft, one can infer the accelerations purely due to gravity and thus the Earth's gravity field. The synergistic combination of the GPS receiver and the accelerometers is capable of providing information about the Earth's gravity field, from which information concerning glacier mass balance and ocean mass distributions can be obtained. These facts are captured in the following rule:

```
(define-rule CAPABILITIES::GRAVITY-measurements
   "Define measurement capabilities of the GeoSCAN accelerometer"
IF there is a Manifested-instrument (with Id ?id) and (Name GRAVITY)
   => (THEN)

ASSERT Measurement (Parameter "5.1.1 Geoid and gravity field variations") (Id
GRAVITY1)))
ASSERT Measurement (Parameter "3.2.6 Ocean mass distribution") (Id GRAVITY2)))
ASSERT Measurement (Parameter "4.1.3 glacier mass balance") (Id GRAVITY3)))
ASSERT Measurement (Parameter "2.7.3 groundwater storage") (Id GRAVITY4)))
ASSERT Measurement (Parameter "3.2.2 seafloor topography") (Id GRAVITY5)))
ASSERT cross-registered (measurements GRAVITY1 GRAVITY2 GRAVITY3 GRAVITY4 GRAVITY5)
(with degree-of-cross-registration instrument) (on platform ?id)))
```

Code 28: Capability rule for the GeoSCAN accelerometer

The fourth system sensor is the visible imager. The visible imager is a 1024 x 1024 pixel CMOS camera with configurable with different filters in the 400nm-1000nm spectral range. The main goal of the visible imager is to provide multi-purpose 4km-resolution land and ocean imagery for real-time applications such as disaster management. A 100deg FOV provides an 8-minute revisit time. The measurement capabilities of the visible imager are summarized in Code 29.

```
(define-rule CAPABILITIES::IMAGER-measurements
   "Define measurement capabilities of the GeoSCAN visible imager"
   IF there is a Manifested-instrument (with Id ?id) and (Name IMAGER)
   => (THEN)
ASSERT Measurement (Parameter "2.6.3 disaster monitoring") (Id IMAGER1)))
ASSERT Measurement (Parameter "2.7. Fire detection and monitoring") (Id IMAGER2)))
ASSERT Measurement (Parameter "2.6.1 land use") (Id IMAGER3)))
ASSERT Measurement (Parameter "2.6.2 landcover status") (Id IMAGER4)))
ASSERT Measurement (Parameter "1.5.4 cloud mask") (Id IMAGER5)))
ASSERT Measurement (Parameter "1.5.2 Cloud type") (Id IMAGER6)))
ASSERT Measurement (Parameter "1.5.3 Cloud amount/distribution -horizontal and
vertical-") (Id IMAGER7)))
ASSERT Measurement (Parameter "2.4.2 vegetation state") (Id IMAGER8)))
ASSERT cross-registered (measurements IMAGER1 IMAGER2 IMAGER3 IMAGER4 IMAGER5 IMAGER6
IMAGER7 IMAGER8) (degree-of-cross-registration instrument) (platform ?id)))
```

Code 29: Capability rule for the GeoSCAN visible imager

This imager is synergistic with other sensors, providing for example a cloud mask product. Furthermore, disaggregation schemes could potentially be used to combine the coarse spatial resolution but frequent measurements of the imager with finer resolution, less frequent measurements of other instruments.

The fifth system sensor is the spectrometer. The spectrometer covers the spectral range between 200nm-2000nm with a spectral resolution of 0.5nm (although this does not necessarily mean full sampling at this resolution). Its 50deg FOV results in a 37min revisit time. Two main potential limitations of the spectrometer are its signal-to-noise ratio and the trade-off between imaging capability and spectral sampling. Should the signal-to-noise ratio be sufficient, such an instrument would be able to make extremely important measurements in the UV (e.g., O_3, NO_x, SO_2) and NIR (e.g., CO, CO_2, CH_4, aerosols). Furthermore, if the instrument had enough imaging capability, the $0.7\mu m$ water vapor spectral feature could be used to produce water vapor and water vapor transport images. These optimistic capabilities are summarized in Code 30. The spectrometer is one of the most promising instruments in the set, but its real capabilities of the spectrometer are still uncertain at this stage of development. Note that, while the capabilities outlined in Code 30 assume an instrument optimized for atmospheric chemistry, its spectral range is also compatible of other applications, such as ocean color, and vegetation measurements.

```
(define-rule CAPABILITIES::SPECTROMETER-measurements
   "Define measurement capabilities of the GeoSCAN spectrometer configured for
atmospheric chemistry"
 IF there is a Manifested-instrument (with Id ?id) and (Name SPECTROMETER)

   => (THEN)

ASSERT Measurement (Parameter "1.8.1 H2O") (Id SPECTROMETER1)))
ASSERT Measurement (Parameter "1.8.2 O3") (Id SPECTROMETER2)))
ASSERT Measurement (Parameter "1.8.3 CO2") (Id SPECTROMETER3)))
ASSERT Measurement (Parameter "1.8.4 CH4") (Id SPECTROMETER4)))
ASSERT Measurement (Parameter "1.8.5 CO") (Id SPECTROMETER5)))
ASSERT Measurement (Parameter "1.8.11 SO2") (Id SPECTROMETER6)))
ASSERT Measurement (Parameter "1.8.12 Vulcanic SO2, OCS and other vulcanic aerosols")
(Id SPECTROMETER7)))
ASSERT Measurement (Parameter "1.8.13 Black carbon and other polluting aerosols") (Id
SPECTROMETER8)))
ASSERT Measurement (Parameter "1.1.1 aerosol height/optical depth") (Id
SPECTROMETER9)))
ASSERT Measurement (Parameter "1.3.2 Water vapor transport - Winds") (Id
SPECTROMETER10)))
ASSERT cross-registered (measurements SPECTROMETER1 SPECTROMETER2 SPECTROMETER3
SPECTROMETER4 SPECTROMETER5 SPECTROMETER6 SPECTROMETER7 SPECTROMETER8 SPECTROMETER9
SPECTROMETER10) (degree-of-cross-registration instrument) (platform ?id)))
```

Code 30: Capability rule for the GeoSCAN spectrometer

The last system sensor is the dosimeter. The dosimeter is capable of measuring total ionizing dose with a sensitivity of 14 μRad. While the dosimeter provides valuable measurements to the space weather community as expressed in the 2003 space weather decadal survey, it does not satisfy any of the objectives expressed in the Earth science decadal survey, which is used as a reference in this case study as mentioned earlier. As a consequence, the dosimeter was left out of the analysis.

In addition to the instruments preselected to be the GeoSCAN system sensors, which were just presented, three more instruments are described in the following paragraphs: a millimeter-wave atmospheric sounder based on (Blackwell et al., 2011), a GNSS receiver configured in reflectometry mode, as per PAU's (Camps et al., 2008) and PARIS' designs (Acevo et al., 2010), and a common DORIS receiver (Dorrer, Laborde, & Deschamps, 1991). These or similar instruments were candidates in the selection process but they were finally not selected.

The mm-wave sounder can measure profiles of atmospheric temperature and pressure through hyperspectral sampling of O_2's rotational line at 118GHz, and it can measure profiles of atmospheric water vapor, humidity, cloud liquid water, and precipitation, through hyperspectral sampling of H_2O's rotational line at 183GHz. Similar technology using larger apertures has been successful in measuring ozone, ClO, and other trace gases using near-by rotational lines, such as the 184GHz O_3 line. However it is unclear that a radiometer of this size would be capable of achieving enough signal-to-noise ratio to perform these atmospheric chemistry measurements. Thus, we will consider two versions of this instrument: a) a basic version (MWAS-BAS) measuring only temperature, humidity, liquid water and precipitation through the 118GHz and 183GHz channels; b) an advanced version (MWAS-ADV) that will have a few atmospheric chemistry channels instead of the oxygen (temperature) channels. The characteristics of the advanced version of the instrument are shown in Code 31. The capabilities of the basic version are restrained to the atmospheric temperature, humidity, water vapor and disaster monitoring measurements.

There is evidence that a GPS receiver in reflectometry mode could measure at least sea surface height and sea surface wind speed (Martin-Neira, Caparrini, Font-Rossello, Lannelongue, & Vallmitjana, 2001), with sufficient accuracy to provide useful data products. Other uses have also been studied both theoretically and experimentally, such as soil moisture (Rodriguez-Alvarez et al., 2008), land topography and vegetation height (Rodriguez-Alvarez et al., 2011), sea ice, land snow/ice thickness (Gleason, Adjrad, & Unwin, 2005). A recent (2010) survey of the capabilities of GNSS reflectometry confirmed these measurements (Jin & Komjathy, 2010). The instrument capabilities considered in this thesis are consistent with these findings and are shown in Code 32.

```
(define-rule CAPABILITIES::MWAS-ADV-measurements
   "Define measurement capabilities of the advanced version of the MWAS instrument"
IF there is a Manifested-instrument (with Id ?id) and (Name MWAS-ADV)

   => (THEN)

ASSERT Measurement (Parameter "1.3.1 Atmospheric humidity -indirect-") (Id MWAS-
BAS1)))
ASSERT Measurement (Parameter "1.7.1 Cloud liquid water and precipitation rate") (Id
MWAS-BAS2)))
ASSERT Measurement (Parameter "1.8.1 H2O") (Id MWAS-BAS3)))
ASSERT Measurement (Parameter "1.8.2 O3") (Id MWAS-BAS4)))
ASSERT Measurement (Parameter "1.8.5 CO") (Id MWAS-BAS5)))
ASSERT Measurement (Parameter "1.8.7 NOx-NO, NO2-, N2O5, HNO3") (Id MWAS-BAS6)))
ASSERT Measurement (Parameter "1.8.9 CFCs/HFCs") (Id MWAS-BAS7)))
ASSERT Measurement (Parameter "1.8.10 H2O2, OH, HO2 and isotopes -HDO, H218O-") (Id
MWAS-BAS8)))
ASSERT Measurement (Parameter "1.8.12 Vulcanic SO2, OCS and other vulcanic aerosols")
(Id MWAS-BAS9)))
ASSERT Measurement (Parameter "1.8.14 ClO, BrO, halogen compounds") (Id MWAS-BAS10)))
ASSERT Measurement (Parameter "2.6.3 disaster monitoring") (Id MWAS-BAS11)))
ASSERT cross-registered (measurements MWAS-BAS1 MWAS-BAS2 MWAS-BAS3 MWAS-BAS4 MWAS-
BAS5 MWAS-BAS6 MWAS-BAS7 MWAS-BAS8 MWAS-BAS9 MWAS-BAS10 MWAS-BAS11) (degree-of-cross-
registration instrument) (platform ?id)))
```

Code 31: Instrument capabilities of the advanced version of the MWAS instrument

```
(define-rule CAPABILITIES::REFLECTOM-measurements
   "Define measurement capabilities of the advanced version of the MWAS instrument"
IF there is a Manifested-instrument (with Id ?id) and (Name REFLECTOM)

   => (THEN)

ASSERT Measurement (Parameter "3.2.1 Sea level height") (Id REFLECTOM1)))
ASSERT Measurement (Parameter "3.4.1 Ocean surface wind speed") (Id REFLECTOM2)))
ASSERT Measurement (Parameter "3.6.1 Ocean wave height and spectrum") (Id
REFLECTOM3)))
ASSERT Measurement (Parameter "2.2.2 Hi-res topography") (Id REFLECTOM4)))
ASSERT Measurement (Parameter "2.4.3 vegetation height") (Id REFLECTOM5)))
ASSERT Measurement (Parameter "2.3.2 soil moisture") (Id REFLECTOM6)))
ASSERT Measurement (Parameter "4.2.2 snow depth") (Id REFLECTOM7)))
ASSERT Measurement (Parameter "4.3.2 Sea ice cover") (Id REFLECTOM8)))
ASSERT Measurement (Parameter "4.3.1 Sea ice thickness") (Id REFLECTOM9)))

ASSERT cross-registered (measurements REFLECTOM1 REFLECTOM2 REFLECTOM3 REFLECTOM4
REFLECTOM5 REFLECTOM6 REFLECTOM7 REFLECTOM8 REFLECTOM9) (degree-of-cross-registration
instrument) (platform ?id)))
```

Code 32: Capabilities of the GNSS reflectometer

Finally, the DORIS receiver is capable of providing total electron content measurements and precise orbitography, as shown in Code 33.

```
(defrule CAPABILITIES::DORIS-measurements
   "Define measurement capabilities of the DORIS receiver"
   ?this <- (Manifested-instrument (Id ?id) (Name DORIS) (flies-in ?miss))

   =>

ASSERT Measurement (Parameter "A8.Total electron content in ionosphere") (Id
DORIS1)))
ASSERT Measurement (Parameter "A9.Precise Orbit Determination") (Id DORIS2)))
ASSERT cross-registered (measurements DORIS1 DORIS2) (degree-of-cross-registration
instrument) (platform ?id)))
```

Code 33: Capabilities of the DORIS receiver

7.3 Results

7.3.1 Preliminaries

Before presenting the results of the main questions at hand (i.e., instrument selection, comparison with Decadal architecture), intermediate results are presented and discussed. In particular, single instrument scientific scores, and bilateral interactions between instruments both at the science and engineering levels (i.e. S-DSM and E-DSM) are introduced, as well as the data continuity matrix that applies for the Iridium case study.

7.3.1.1 Instrument scores

Instrument scores in isolation computed using the Decadal objectives are shown in Figure 83, and explained below in more detail instrument by instrument.

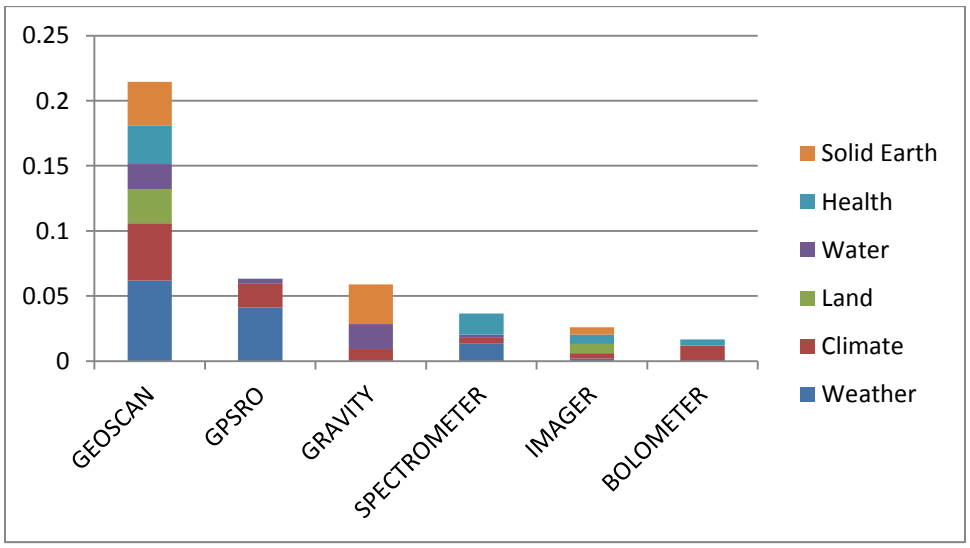


Figure 83: GeoSCAN instrument scores in isolation, including total score of the whole GEOSCAN mission

Imager: The imager gets a score of 2.6% distributed between the weather, climate, ecosystems, solid Earth, and human health panels. This value comes entirely from disaster management, cloud type, and vegetation state measurements. The human health measurement requirements in terms of cloud type and vegetation state measurements are fully satisfied, but the climate and weather measurement requirements are only partially satisfied due to the 4-km horizontal spatial resolution of the instrument, which is not sufficient according to Decadal survey requirements. Cloud amount and distribution measurement requirements are missed due to the lack of vertical spatial resolution of the imager. Concerning the disaster management objectives, they are all partially satisfied due to the combination of low horizontal spatial resolution and poor spectral sampling. The landcover status measurements are not judged to provide measurable value based on the same combination of horizontal spatial resolution and spectral sampling.

Microbolometer: The microbolometer gets a score of 1.7% that comes from satisfaction of radiation budget measurement requirements by the climate and health panels. All these objectives are fully satisfied.

GPSRO: The GPSRO instrument gets a score of 6.3% that comes from satisfaction of weather, climate, water, and health objectives. The Decadal weather and climate panels made an explicit requirement to have a constellation of GPSRO receivers in LEO, which is fully satisfied. The measurement requirements for atmospheric temperature and humidity measurements are fully satisfied for the climate panel, but only partially satisfied for the weather panel, which desires a higher vertical spatial resolution with sensitivity in the lower troposphere to input into the numerical weather prediction assimilation algorithms. The same explanation applies to the water panel.

Accelerometers: The gravity payload (i.e. the combination of the accelerometers and the GPS receiver) gets a score of 5.9% that comes from satisfaction of objectives of the solid Earth, water, and climate panels. All the solid earth and water objectives for gravity fields, glacier mass balance, ocean mass distribution, and groundwater storage are fully satisfied by the constellation. The glacier mass balance requirements by the climate panel are not fully satisfied because they require a higher spatial resolution that can only be achieved through a laser measurement (e.g. Icesat-II).

Spectrometer: The spectrometer in its “optimistic version” (i.e., where it has enough signal-to-noise ratio to measure all greenhouse gases of interest) gets a score of 3.7% through satisfaction of climate, health, weather, and water objectives related to greenhouse gases, aerosols, and water vapor. All these measurement requirements are only partially satisfied due to a combination of poor imaging capability and lower accuracy with respect to larger instruments.

Millimeter-wave sounder: As explained in the section describing instrument capabilities, two versions of the MWAS instrument are considered: MWAS-BAS which has the 118GHz O2 channels for temperature plus the 183GHz water vapor channels for humidity and liquid water, and MWAS-ADV, which instead of the temperature channels, features chemistry channels, namely the ozone at 184Ghz and ClO at 204GHz. MWAS-BAS can measure atmospheric humidity, atmospheric temperature, cloud liquid water and precipitation with good accuracy and vertical resolution. All these measurements are extremely important to the weather and climate panel, and humidity and precipitation measurements are also relevant to the water and human health panels, which results in an overall instrument score of 5.0%. This score assumes that a cloud mask product is available, for example from the imager. Moreover, this score does not take into account that the temporal resolution and diurnal sampling of this architecture exceeds the capabilities required by the Decadal objectives. MWAS-ADV achieves a higher score (9.1%) thanks to the additional chemistry instruments that satisfy several measurement requirements from the health and weather panels.

Reflectometer: The GPS receiver in reflectometry mode gets an overall score of 8.6% through partial satisfaction of several objectives of the climate, health, and solid earth panels that are otherwise not achievable by any other GeoSCAN sensor. Scientific objectives related to sea level height, sea surface wind, sea ice cover and thickness, vegetation height, and land topography are partially satisfied. We note that these measurements are still to be fully demonstrated. Even if they were demonstrated, the accuracy and horizontal spatial resolution of such measurements would likely be insufficient to be comparable to larger instruments and in particular, KaRIN (the Decadal survey wide-swath altimeter), and LIST (the imaging laser altimeter). We note that about 3 percentage points from the original 8.6% value (i.e. 35%) depends on the performance of the reflectometer concerning soil moisture, which is deemed a critical measurement for the water panel.

7.3.1.2 Bilateral instrument synergies

The model identifies two synergies between the GeoSCAN system sensors, as illustrated in Figure 84.

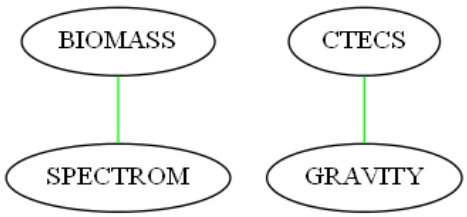


Figure 84: Bilateral synergies between GeoSCAN system sensors

The imager and the spectrometer have synergies because the cloud mask product of the imager is used to improve the quality of the atmospheric chemistry products of the spectrometer. This synergy has a strength of +2.2%. The GPSRO and the accelerometers are also synergistic (+2.3%), because all useful gravity products are obtained through the combination of non-gravitational accelerations from the accelerometer and precise orbit determination from the GPS receiver.

If we introduce the reflectometer and the hyperspectral millimeter wave sounder, the new diagram is shown in Figure 85.

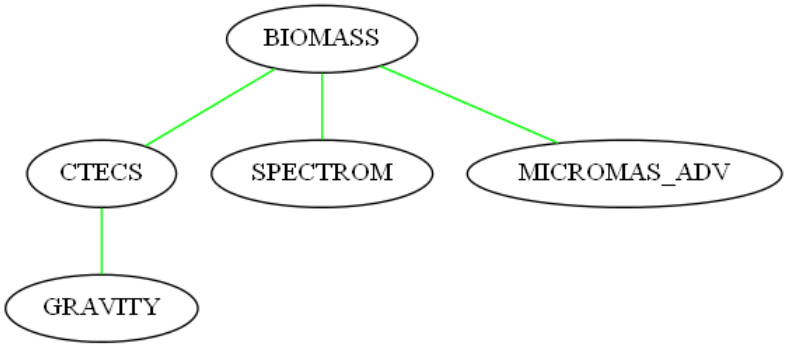


Figure 85: Bilateral synergies between GeoSCAN candidate sensors

Again, this only reflects bilateral synergies that have an impact in score according to Decadal objectives. In reality, there are synergies that are not captured by this graph because either they are of higher order, or they do not have an impact on the Decadal objectives. For instance, the reflectometer is synergistic with:

- both the MMW sounder and the GPS RO because the atmospheric humidity measurement reduces the wet tropospheric component of the altimetry error budget
- both the DORIS receiver and the GPS RO because the total electron content measurement reduces the ionospheric component of the altimetry error budget.

All these synergies are captured by the model, .i.e., the new and modified data products are created by the rules engine, but if they do not satisfy any additional objective, this “real synergy” does not translate into a positive score in the corresponding entry of the S-DSM.

7.3.1.3 Marginal instrument scores

As explained in Section 5.3.1.1, scores in isolation are not an ideal way of representing instrument value because they neglect synergies between instruments.

A better way of representing instrument value with respect to a certain subset of instruments considered as the "reference", is to compute the marginal scores, as the difference between the score of the reference and the score of the reference without the instrument at hand: $V_i' = V_{\{ref\}} - V_{\{ref \setminus i\}}$. The marginal instrument scores for the GeoSCAN selected system sensors are shown below.

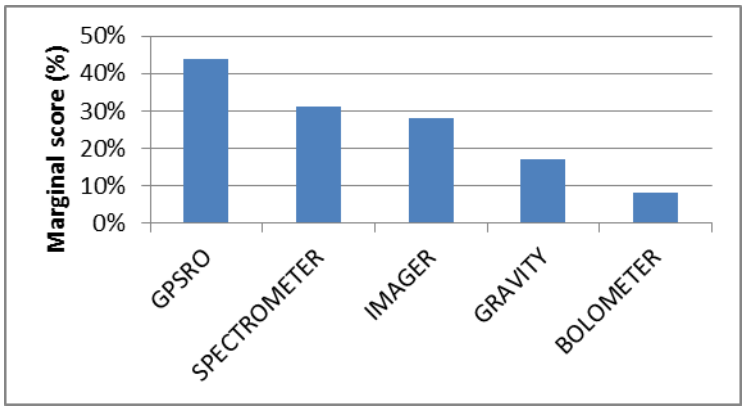


Figure 86: Marginal scores for the GeoSCAN selected system sensors

It is apparent from Figure 86 that the value of the imager is much greater when the synergies with the other instruments are considered. The bolometer on the other hand is the lowest marginal score, as it has little synergies with the other instruments, even though its measurements have high relative value.

Another interesting analysis is to compute the upward marginal scores for the three non-selected instruments, namely the reflectometer, the DORIS, and the two versions of the mmw sounder (MWAS). These computations are shown in Figure 87.

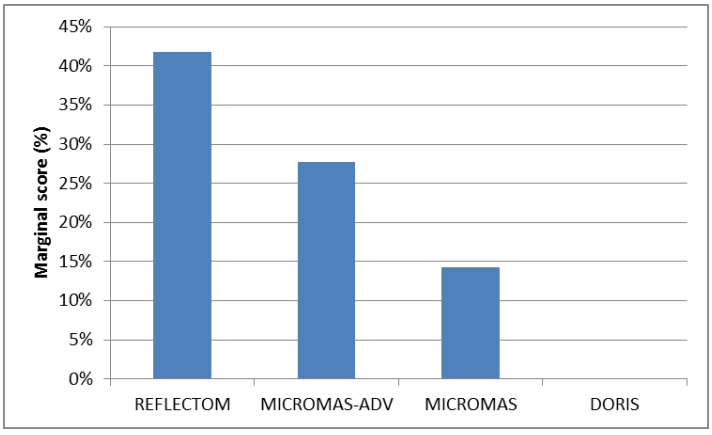


Figure 87: Marginal scores of potential GeoSCAN hosted payloads w.r.t. system sensors

It is clear from Figure 87 that it is not interesting to choose DORIS given the current set of system sensors because it is essentially redundant with the GPS receivers in both their measurements of total electron content and precise orbit determination. From the three others, the reflectometer gets the higher marginal score, followed by the two versions of the MMW sounder. This ranking coincides roughly with the ranking of uncertainty in the capabilities of these instruments and therefore should not be used as the sole basis for instrument selection.

7.3.1.4 Data Continuity Matrix

The data continuity matrix for the GeoSCAN case study was computed by considering all domestic and ESA missions (obviously except for GeoSCAN) between the years 2010 and 2025. Only a reduced subset of measurements considered critical in terms of data continuity were considered. The matrix is shown in Figure 88.

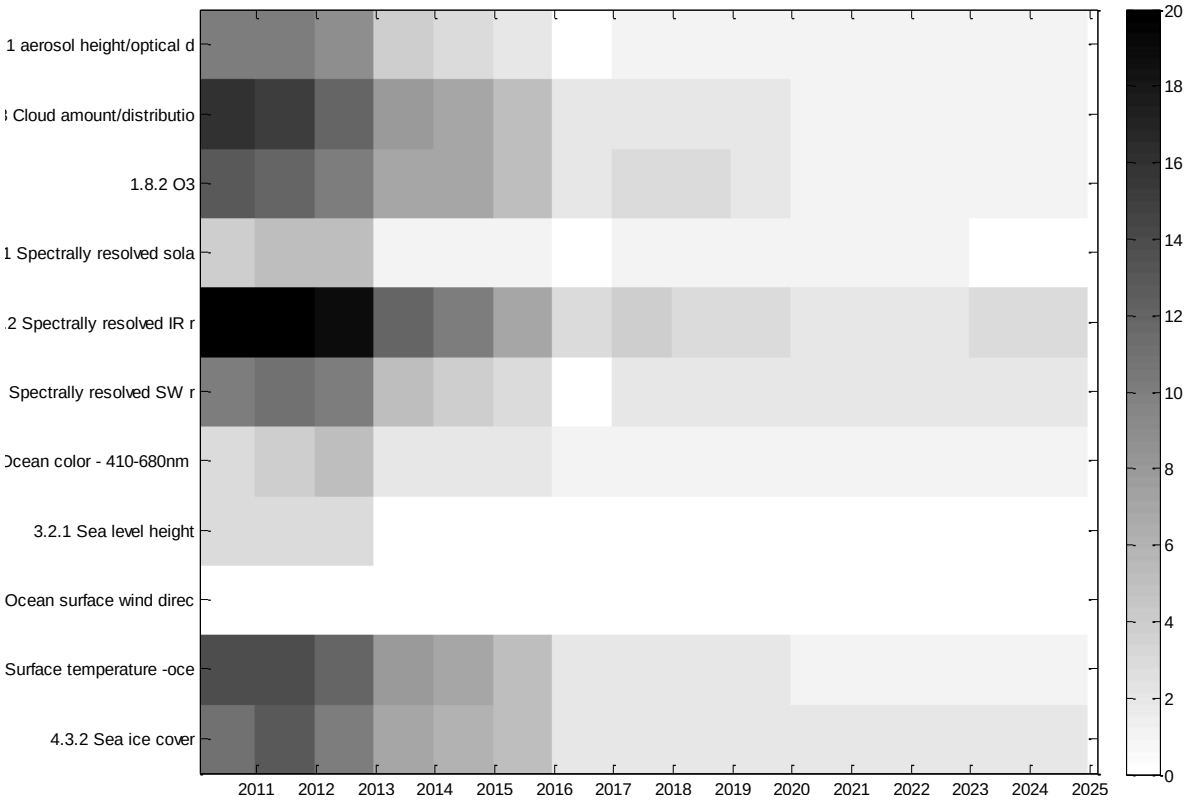


Figure 88: Data continuity matrix for the GeoSCAN case study

As shown in Figure 88, the major potential data gaps for the next decade, when the Decadal missions are taken into account, are the following:

- Earth radiation budget

- Altimetry
- Scatterometry
- Aerosols

Hence, the instrument that is most likely to cover an important data gaps is the microbolometer. The spectrometer can also play a key role if its capabilities to provide useful aerosol data products are confirmed.

7.3.2 Cost-effectiveness comparison

One of the goals of this case study is to compare the performance and more importantly the cost-effectiveness of the GEOscan program with those of the Decadal Survey set of missions. The comparison is done using the tier I and tier II reference Decadal missions, on the basis of Decadal objectives. The result of such comparison is shown on Figure 89 and Figure 90. Weighted scores use panel weights from (Sutherland, 2009); unweighted scores use uniform panel weights.

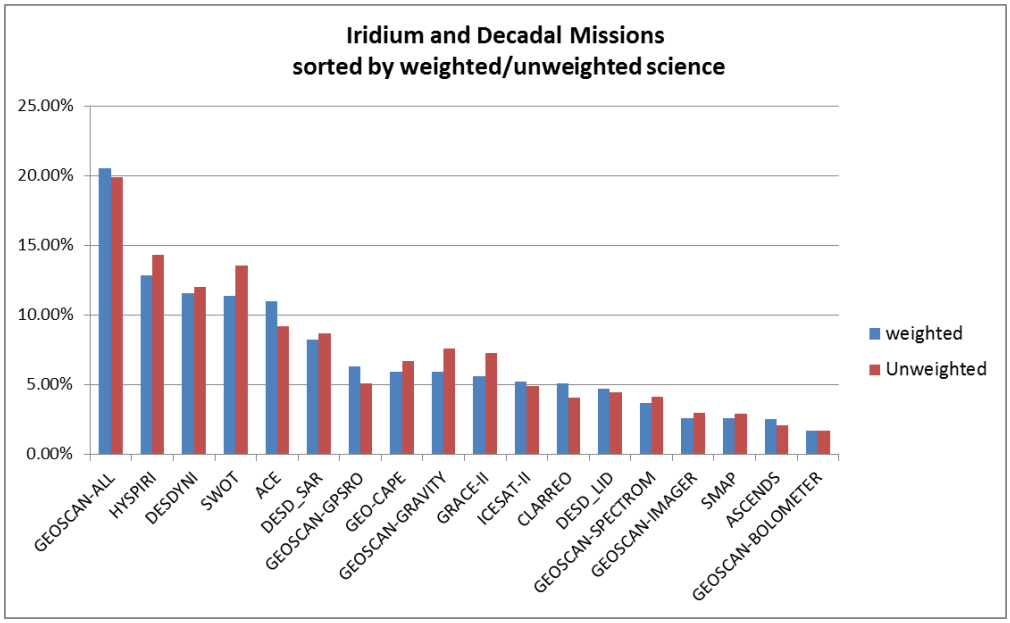


Figure 89: Comparison of GEOScan vs Decadal Survey missions in terms of scientific/societal benefit.

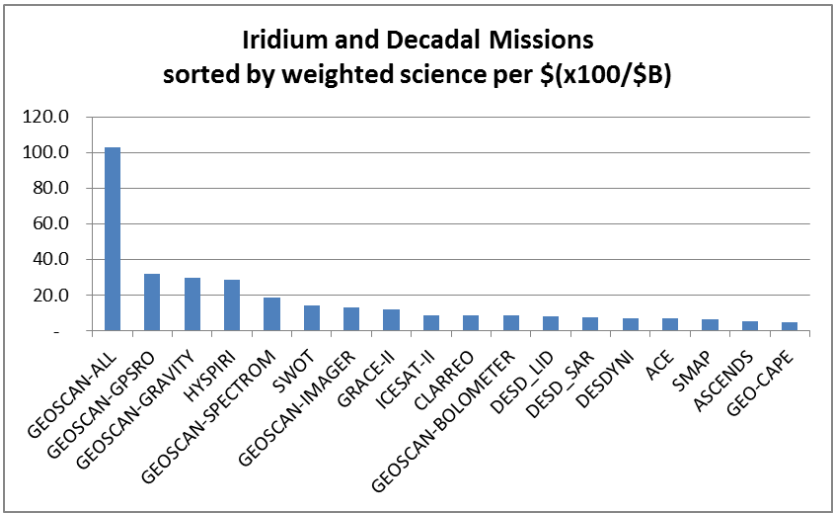


Figure 90: Comparison of GEOScan vs Decadal Survey missions in terms of scientific/societal benefit

Figure 89 shows the ranking of GEOscan and the Decadal missions in terms of science (both unweighted and weighted using the panel weights defined in Section 6.2.1.1). The main result is obvious from the chart: **GEOscan as a whole** (i.e., the 66 satellites carrying the suite of system sensors) **has a higher scientific and societal value than any one Decadal mission,** regardless of whether the panel weights are or are not taken into account. Furthermore, the ranking of individual GEOscan missions (i.e., the 66 satellites carrying for example the GPSRO) is comparable to that of Decadal missions.

These conclusions are accentuated when cost-effectiveness is taken into account. Four out of the top five missions in cost effectiveness are GEOscan missions. Note that the cost effectiveness calculations assume that the cost of both individual and the complete GEOscan mission is $200M, and the costs of the Decadal missions are the ones provided in Figure 60. Figure 90 shows that GEOscan as a whole is several times more cost-effective than any Decadal mission. Moreover, the cost-effectiveness of any one Decadal mission is comparable to a GEOscan individual mission assuming a cost of $200M for the latter.

7.3.3 System sensor selection problem

The GeoSCAN system sensor selection problem takes as an input a set of candidate instruments and outputs several subsets of instruments that are optimal. The term “optimal” here refers to cost-effectiveness, i.e., maximization of science for a certain cost, or equivalently, minimization of cost for a certain science level.

The candidate instruments considered for GeoSCAN system sensors are all the instruments described in Section 7.2.2, including the instruments discarded by GeoSCAN, namely MWAS, the GPS receiver in reflectometry mode, and the DORIS receiver. Space weather instruments are not considered here, as no scientific objectives concerning space weather were laid out in the Earth science decadal survey.

Before presenting and discussing the results of the EOS instrument selection case study, we remind the different types of rules that were used to obtain these results. According to the thesis objectives, some of these results are taken from the library of classes of SAPs, some are domain-specific but common to all case studies, some are domain-independent and SAP class-independent, and some are specific to the EOS instrument selection problem.

The results presented in this section were obtained with the following set of rules:

Class of rule	**Type of knowledge**	**Source of rule**
Grammar and enumeration rules	Domain-independent, common to all down-selecting problems	Library-DsPs
Search heuristics	Domain-independent, common to all down-selecting problems	Library-DsPs
VASSAR-aggregation rules	Decadal/GeoSCAN-specific	See Section 7.2
VASSAR-requirement satisfaction rules	Decadal/GeoSCAN-specific	See Section 7.2
VASSAR-instrument capability rules	GeoSCAN-specific	See Section 7.2
VASSAR-attribute inheritance rules	Domain-specific, common to all EOSS, all SAPs	VASSAR
VASSAR-synergy rules	Common to all SAPs	VASSAR
VASSAR-explanation rules	Multiple types	VASSAR
Down-selecting rules	Domain-independent, common to all down-selecting problems	See section 2.3.6

Table 49: Summary of rules utilized in the GeoSCAN instrument selection case study

No additional enumeration constraints were taken into account as given the small size of the tradespace, full factorial enumeration of all instrument sets is possible. The results of this full factorial enumeration of 255 architectures in the cost-science space are shown in Figure 91. In Figure 91, each diamond represents one GeoSCAN selection architecture (i.e., a subset of instruments) in the cost-science space. The color of this diamond indicates the reason –if any- why this architecture was eliminated in the down-selection process. The following down-selection rules were applied:

- Lifecycle cost < $50M (**cost of instruments only**)
- Normalized Science > 0.15
- Utility > 0.55
- Pareto ranking < 3
- Normalized programmatic risk < 0.15

- Fairness > 0.01

In addition to these rules, an ad-hoc down-selection rule was incorporated in this Iridium case study to take into account geometrical constraints. This heuristic rule deletes architectures that are unlikely to fit in the GeoSCAN allotted space, shown in Figure 91. More precisely, a this rule enforces that a maximum of two instruments are chosen out of a set of "large instruments", which contains the two versions of the MMW sounder, the imager, and the accelerometers. These instruments are much bigger than the rest and thus are assumed to drive the geometrical constraints of the architecture (see Figure 92).

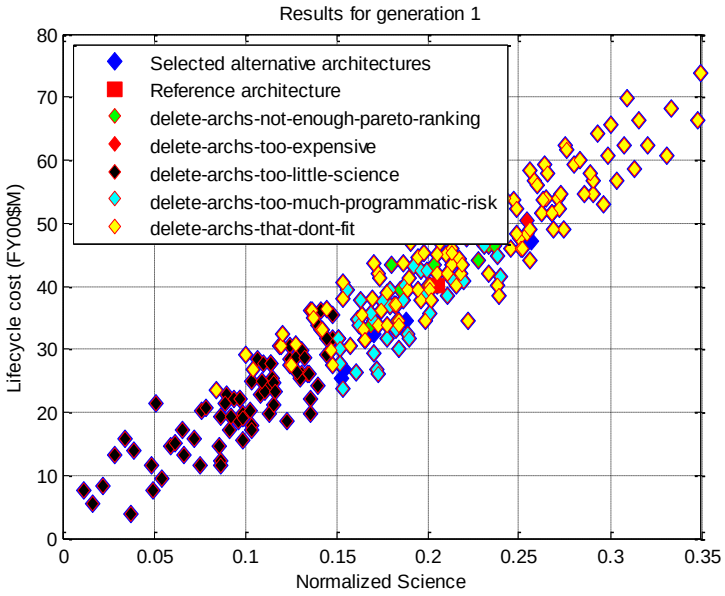


Figure 91: Science cost tradespace for all GeoSCAN selection architectures

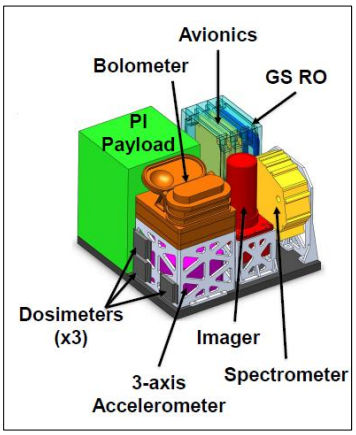


Figure 92: Preliminary allocation of system sensors and hosted payload on GeoSCAN

The remaining architectures after down-selection are shown in Figure 93, together with the reference architecture and the utopia point. Figure 93 shows two different fuzzy Pareto fronts that are obtained when the down-selection rules previously described are applied, with the only difference of the maximum value of programmatic risk allowed, which is 15% in one case, and 30% in the other. As a reminder, programmatic risk in the instrument selection problem is estimated by counting the fraction of the instruments selected that have an initial TRL < 5.

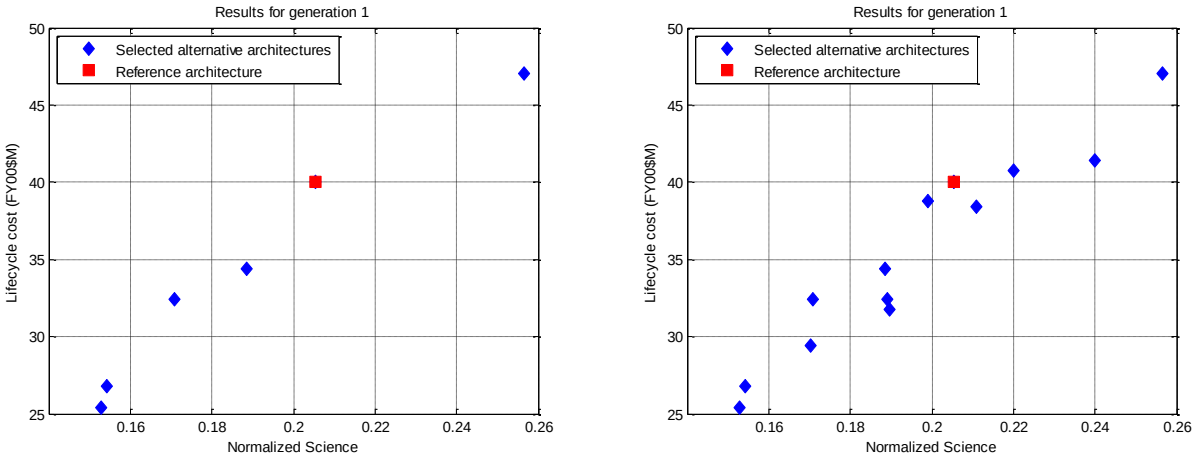


Figure 93: Fuzzy Pareto frontier of GeoSCAN selection architectures for two max levels of programmatic risk: 15% (left) and 30% (right)

In the case of GeoSCAN, it was considered that only the advanced version of the MMW sounder and the reflectometer had an initial TRL less than five. Although functional versions of these instruments have been developed and in at least partially tested, they would have to be slightly modified (number of channels, dimensions) for GeoSCAN, which explains the decision to assign them a relatively lower TRL than the other instruments.

A few architectures were highlighted on the Fuzzy Pareto front of Figure 93. These architectures are described in Table 50.

Table 50: Detail of GEOSCAN selection architectures on fuzzy Pareto front for max programmatic risk = 30%

Arch. id#	Instruments added w.r.t. ref	Instruments deleted w.r.t. ref
1	REFLECTOM	GRAVITY
2	REFLECTOM	LORENTZ_ERB GRAVITY
3	REFLECTOM	LORENTZ_ERB BIOMASS
4	MWAS-ADVANCED	GRAVITY BIOMASS

Figure 93 shows that **for low maximum allowed levels of programmatic risk (15%), the set of instruments selected as system sensors lies on the Pareto front of the science-cost trade space**. However, in a scenario where a higher programmatic risk is tolerated (30% of instruments with TRL<5), non-dominated architectures systematically include one of the high risk instruments (i.e., the reflectometer or the advanced version of the mmw sounder) instead of one or more of the low risk instruments (e.g., accelerometers, microbolometer), as can be seen on Table 50. Furthermore, the choice to delete these instruments and not others is consistent with the marginal scores discussed in 7.3.1.3.

7.4 Conclusion

In the third and last case study, the framework was applied to the Iridium GEOscan program. The three main goals of the case study were: a) to study the system sensor selection problem; b) to compare the cost-effectiveness of GEOscan with that of the Decadal Survey; c) to suggest candidate instruments for hosted payloads.

The system sensors selection problem was studied in detail. The study largely agrees with the system sensors selected by GEOscan, with the caveat of the uncertainty in the capabilities of the spectrometer.

A comparison of the science value and cost-effectiveness of the GEOscan program with the Decadal Tier I and Tier II missions has shown that GEOscan as a whole is several times more cost effective than any Decadal mission if the cost of $200M for the GEOscan program is confirmed. GEOscan also provides more scientific and societal value than any single Decadal mission. Individual GEOscan missions are usually not comparable to Decadal missions in terms of absolute science value (with the exception of GPSRO and the gravity mission), but they are definitely comparable in terms of cost-effectiveness.

Finally, the millimeter-wave atmospheric sounder and the GPS in reflectometry mode were identified as top priorities for the hosted payloads.

8 Conclusions

8.1 Thesis Summary

This thesis has introduced a new class of decision support tools for system architecting called rule-based system architecting that incorporates a rule-based engine for improved scalability and traceability when dealing with knowledge-intensive system architecting problems.

The thesis started by recognizing that there are three different bodies of knowledge required to solve an SAP: knowledge that is independent of both the domain and the class of SAP, knowledge that is domain-independent, but SAP-class specific; and knowledge that is domain-specific and SAP-class independent. One of the major points of the thesis is that the utilization of a rule-based system enables the physical separation of the three bodies of knowledge, which brings about the desired scalability.

Most of the theory of rule-based system architecting was developed in Chapter 2. First, the idea of the existence of "classes" of SAP was introduced. Five classes of SAP were identified and formally defined: assigning problems, which are the closest to Simmon's ADG formulation of SAPs; set partitioning and covering problems, where the optimization is done over the set of all possible partitions of a set; down-selecting problems, in which the optimization occurs over the set of all the possible subsets of a set; permuting problems, where the optimization occurs over the set of all possible permutations of a set of elements; and connecting problems where the optimization occurs over the set of all possible graphs based on a given set of nodes.

For each class of SAPs, SAP class-specific knowledge was given in the form of three types of rules that can be reused from problem to problem: grammars for enumerating feasible architectures, search heuristic rules for searching through the architectural tradespace effectively, and selection rules for down-selecting a subset of preferred architectures.

Furthermore, a generic tradespace search algorithm was presented that can be used to solve any SAP. More precisely, it is a population-based algorithm designed to be augmented with heuristics that contain both domain-specific and SAP-class specific knowledge in the form of rules.

Chapter 3 focused on approximate evaluation rules, and introduced a generic methodology that can be used to assess the value of an architecture using rules. This methodology (VASSAR) consists of the following types of rules: attribute inheritance rules, capability rules, emergence rules, requirement satisfaction rules, value aggregation rules, and explanation rules. The methodology has 11 steps: 1) assert subsystem facts according to architecture decomposition; 2) inherit subsystem properties from system properties using attribute inheritance rules; 3) assert component facts according to subsystem decomposition; 4) inherit component properties from subsystem and system properties, or from a component database, using attribute inheritance rules; 5) assert capability facts through matching of subsystem facts and capability rules; 6) inherit capability properties from component, subsystem, and system properties; 7) assert new and modify existing capabilities by running the emergence rules module; 8) transform back and forth from the numerical world to the fuzzy world using fuzzy attribute rules; 9) assert full and partial subobjective satisfaction facts from matching capability facts to requirement satisfaction rules; 10) obtain the value of the architecture from applying the value aggregation rules to the subobjective satisfaction facts; 11) obtain the explanations behind the value of the architecture by using explanation rules to show the value delivery loop.

Chapter 4 focused on domain-specific knowledge, and introduced the domain of application of this thesis, namely Earth Observing Satellite systems (EOSS). A precise definition of what it is understood by architecture of an EOSS was provided, that identified three main categories of architectural decisions: instrument selection, instrument packaging, and mission scheduling. Next, the domain specific knowledge was provided in the form of sets of rules that can be used in the evaluation process: orbit selection rules, power budget rules, complexity-corrected mass budget rules, launch vehicle selection rules, and standard bus selection rules. It was shown how the generic VASSAR methodology introduced in Chapter 3 was tailored for EOSS. Finally, several metrics appropriate for architecting EOSS were introduced and described, namely scientific and societal benefit, lifecycle cost, programmatic risk, launch risk, fairness, and data continuity.

The methodology was then applied to three case studies, namely the NASA Earth Observing System, the NASA Decadal Survey, and the GEOscan Iridium case study. The NASA EOS case study was used as a benchmark for the tool. The goal of this case study was to show that the tool can provide useful results by attempting to replicate the decisions that were made in real life for EOS. Instrument selection decisions were successfully replicated overall, except for externalities such as needs from international partners. Instrument packaging decisions were also replicated once the assumptions in terms available buses and launch vehicles were taken into account. Replicating mission scheduling decisions was more challenging and required detailed yearly budget and mission cost data.

The NASA Decadal Survey case study provided several insights into the instrument selection, instrument packaging and mission scheduling problem. For example, a potential redundancy amongst the 8 lidars proposed by the committee was identified: there are opportunities for fusion of the ICESAT-II lidar and the DESDYNI lidar, and the GACM and ACE lidars. Furthermore, the model proposes an alternative class of architectures based on intensive use of trains at key orbits (400 km dawn-dusk SSO for lidars and high energy instruments, 600 km dawn-dusk SSO for imaging SARs, 700km PM orbit atmospheric chemistry studies and continuity with EOS and the A-train, 800km AM SSO for radiation, passive optical imaging, and continuity with weather satellites.

Finally, the GEOscan Iridium case study generally confirmed the optimal selection of the system sensors for the GEOScan program under a tight schedule and assuming a certain degree of risk-aversion. It was also shown that under a more aggressive risk-aversion profile the optimal solution would include higher risk-high reward technologies such as the GPS in reflectometry mode and the advanced millimeter-wave sounder. These two instruments in particular, together with other imagers with some spectral sampling in certain spectral regions of interest (e.g. ocean color, vegetation), were proposed as hosted payloads for the constellation.

8.2 Main Contributions

8.2.1 Methodological contributions

This thesis sets the theoretical foundations for a holistic architecting framework based on a rule-based system. More precisely, the methodological contributions of the thesis are the following:

- It **introduces the notion of classes of system architecting problems**. More precisely, it defines five classes of SAP: assigning problems, set partitioning and covering problems, set permuting problems, down-selecting problems, and connecting problems. This classification is useful because it helps the system architect formulate a system architecting problem as a combinatorial optimization problem.
- It **demonstrates how to separate the three bodies of knowledge needed to solve an SAP** (domain- and SAP class-independent, domain-specific, SAP class-independent, and SAP-class-specific) using a rule-based expert system, in such a way that the scalability of the tool is improved. In other words, it becomes easier to modify existing knowledge and to add new knowledge.

- It **provides a library of classes of SAP**, in which SAP class-specific knowledge for each of the five classes is given in the form of rules that can be directly reused in another application. In particular, the following knowledge is made available for each class: rules and encoding schemes for efficient enumeration of architectures; rules for efficient search through the architectural tradespace; rules for down-selection of architectures. This library is useful because it helps the system architect explore the architectural tradespace efficiently.
- It **proposes a methodology** (VASSAR) **to** a**ssess the value of an architecture** using rules. This methodology enables modeling of emergent behavior through emergence rules and ensures traceability of the value delivery loop through explanation rules. This methodology is useful because it helps system architects make the most of the rule-based system architecting framework by implementing their architecture evaluation functions using rules.
- It applies the whole framework to the domain of Earth Observation Satellite Systems architecting, and in doing so, **the first expert system entirely devoted to architecting Earth observing missions and programs is created**. The value of this resides mostly in the fact that thousands of rules containing expert knowledge about how to architect an EOSS is made publicly available in a format that is compatible with leading rule-based engines. Therefore, this large body of knowledge can be reused.

Rule-based system architecting is also inherently more transparent than other tools, since the trace of rules executed provides a framework around which an explanation facility can be built. Moreover, this explanation facility can also be physically separated from the rest of the code provided that the right data structures are created.

8.2.2 Main findings from the case studies

The major findings from the three case studies are summarized below:

1. **NASA EOS case study**
    - The EOSS RBES was generally capable of replicating the decisions made by EOS management, with a few exceptions that were identified and understood. The most flagrant ones were explained once needs from international partners were taken into account (e.g., selection of MOPITT from Canada, and selection of Japanese AMSR-E instead of European MIMR)
    - Most of the preferred architectures for NASA EOS included the EOSP polarimeter, which was deleted. EOSP was a relatively small and low cost instrument that had really unique capabilities in terms of remote sensing of aerosol properties.

- The model identified an alternative class of packaging architectures which is similar in science and cost to the reference packaging architecture, but is more distributed. Scientific synergies are captured by placing the satellites on train configurations, and lifecycle cost is decreased by using the BCP-2000 standard bus for smaller payloads. This more distributed architecture also decreases programmatic and launch risk.
- The initial mission scheduling model that assumed that all the budget for Earth science missions goes for one mission at a time was proved to be inadequate. In this model, a mission takes a number of years to develop that is equal to its cost estimate divided by the yearly budget. When the cost has been covered by the budget, this mission is launched and investment in the next mission starts. This model was not capable of replicating the results of the EOS case study. The main reason was that in reality, investments on several missions occur simultaneously, leading to situations in which two or more missions are launched in a very small time interval. A second scheduling model had to be developed that allows for multi-mission investment.

2. NRC Decadal Survey case study

- The EOSS RBES identified a potential redundancy between some of the eight lidars proposed by the NRC committee. In particular, the altimeter in ICESAT-II could be combined with the DESDYNI altimeter, and the GACM DIAL could potentially be combined with the ACE DIAL.
- Both the ACE and DESDYNI missions fly large radars and lidars on common platforms as per the reference architecture. The model identifies that it is not desirable to fly the radar and the lidar on the same platform, not only because of cost considerations, but also for scientific issues.
- The launch of GPSRO, HYSPIRI, and GACM should be prioritized because these three missions are all highly cost-effective and have high potential to cover data gaps.

3. GEOscan Iridium case study

- The tool confirmed the set of system sensors selected for GEOSCAN, even though the uncertainty in the capabilities of the spectrometer could render it less valuable than a millimeter wave atmospheric sounder.

- A comparison of the science value and cost-effectiveness of the GEOscan program with the Decadal Tier I and Tier II missions has shown that GEOscan as a whole is several times more cost effective than any Decadal mission if the cost of $200M for the GEOscan program is confirmed. GEOscan also provides more scientific and societal value than any single Decadal mission. Individual GEOscan missions are usually not comparable to Decadal missions in terms of absolute science value (with the exception of GPSRO and the gravity mission), but they are definitely comparable in terms of cost-effectiveness.
- Candidates were identified for hosted payloads. The millimeter-wave atmospheric sounder and the GPS in reflectometry mode are particularly promising, although there is some uncertainty into whether they would fit in the GEOscan allotted space for hosted payloads.

In addition to these findings that are specific to each case study, a cross-case study analysis revealed a few additional insights that are general to all EOSS:

- While the decomposition of the problem of EOSS architecting in instrument selection, instrument packaging, and mission scheduling, was adequate for modeling purposes and largely reflects the reality of how EOSS are actually architected, these three problems are far from being uncoupled.
- Couplings from the scheduling problem to the selection and packaging problems are particularly important and tend to be perceived by decision makers as hard constraints for the selection and packaging problems: a certain instrument must be selected, or must be flown on whichever satellite flies first, because it is a heritage instrument, or because it can close a data gap.
- Couplings between the selection and packaging problems are more relevant when standard buses are used than when dedicated buses are developed. When standard buses are used, instruments may be selected or deselected based on the resources available on the bus after placement of the primary payloads. This "opportunity payloads" strategy may bring forth cost savings and increase the technology readiness of certain instruments to be flown in future missions, by increasing bus and launch vehicle packaging efficiency factors.
- Packaging architectural tradespaces (science vs cost) show clusters of iso-science architectures that represent all architectures capturing the same synergies and with the same orbital conflicts. These architectures are different ways of arranging the same instruments on the same orbits (i.e., monolithic spacecraft vs trains). This suggests that once theses synergies have been identified through some preliminary exploration (some of them may not be obvious at the beginning), a simpler science metric can be constructed that counts the number of synergies captured.

- The packaging problem is driven by a few aspects: science synergies, orbit selection, satellite sizing or standard bus selection, and launch vehicle selection. The quality of the results of the packaging problem strongly depends on the fidelity of the models used to size a satellite, select a launch vehicle, and select an orbit for the spacecraft. In particular:
    - Having a set of "generic" launch vehicles (large, medium, small) is not good enough to achieve useful results. Since the differences in packaging architectures are often driven by differences in launch cost, it is important to consider a realistic pool of launch vehicles for each case study.
    - A similar reasoning applies to standard buses.
    - Producing useful results in the packaging problem requires modeling the dependence of scientific performance – not only lifecycle cost - on orbit selection. Orbital parameters affect scientific performance through sensitivity or signal-to-noise ratio, spatial resolution, temporal resolution, and coverage amongst others. This needs to be taken into account because in addition of the effects of orbit selection on cost.
- The mission scheduling problem is coupled to the technology investment problem. While the budget for mission development and the budget for instrument development are separate in most space agencies, both processes are obviously coupled because payloads need to be integrated into spacecraft before launch. Thus, decisions about investments on particular technologies and decisions about investments on particular missions need to be made consistently.

8.3 Discussion

8.3.1 Features of Rule-based System Architecting

The framework, methodology and tools presented in this thesis, referred to with the term "Rule-based system architecting", have a number of features that help improve the system architecting process:

- **Scalability**: The rule-based system architecture framework is architected to be scalable, i.e., to be able to handle larger, knowledge-intensive problems. This means that it is easier to add new knowledge or modify existing knowledge in the database.
- **Transparency and traceability**. The rule-based expert system has a built-in capability to trace the rules that are executed. This can be augmented with domain-specific knowledge to create powerful explanation facilities that support the system architects as they search through the architectural tradespace. Enhanced interaction with the user may potentially result in higher problem-solving performance.

- **Better modeling of interactions and emergence:** The rule-based system architecture framework provides a powerful environment to model interactions between architectural elements and emergent behavior.
- **Fuzzy requirements and capabilities**: The rule-based system architecture framework allows expression of mixed "fuzzy" and numerical requirements and capabilities
- **Supports a broader variety of system architecting problem classes**: The rule-based system architecture framework supports the system architect in the enumeration, evaluation, search, and down-selection phases of the architecting process in a broader variety of system architecting problem classes than state-of-the-art tools
- **Reusability and knowledge "vault":** The rule-based system architecture framework is architected to facilitate storage and sharing of domain-specific and domain-independent knowledge between problems. As a corollary, it can also be used as a knowledge repository to prevent loss of expertise through time.
- **Fully integrated:** The rule-based system architecture framework fully integrates the Jess rules engine, with: a) a full-scope object-oriented programming language such as Java™; b) the modeling and graphical representation power of Matlab™, c) the simple spreadsheet-based interfaces of Excel™ ; d) The STK mission analysis simulation tool, and potentially any other tool that can communicate with Matlab or Java (e.g., any tool that supports .COM).

8.3.2 Inherent Limitations of the Rule-based System Architecting Framework

While the rule-based system architecting framework has several advantages over state-of-the-art tools, there are also certainly some drawbacks, some of which stem from design tradeoff choices:

- **Knowledge elicitation**: The rule-based system architecting framework is based on the use of large quantities of expert knowledge. This knowledge needs to be elicited from a pool of experts unless the user of the tool is also the expert. This process of knowledge elicitation can in some cases be burdensome and become the bottleneck of the project.
- **Computational performance**: The framework achieves higher scalability and traceability than other tools by trading them against some computational performance. Thus, it is not better or worse than state-of-the-art tools, it is just on a different region of the design space that is better suited for large knowledge-intensive problems. As a corollary, it is probably neither necessary nor recommended using the rule-based system architecting framework to solve a relatively small SAP.

- **No control over flow of execution:** Rules engines control most of the flow of execution of the program and only allow the user to partially interact with it, for example, by setting the priority of certain rules. While this is an advantage from the perspective that the user does not need to create new code each time to control the flow of execution, it can also be a disadvantage because adding a new rule in a large rule base may not result in the expected behavior. This risk is partially mitigated by the creation of a good explanation facility that helps the user understand the outcomes of the model.
- **Emergence is a double-edged sword:** While the ability to model emergent behavior is certainly a positive aspect in the context of system architecting, it also brings about additional challenges. In particular, one needs to be careful in the definition of emergence rules, as rules that are "too powerful" in the creation of new facts may excessively increase execution time. This risk can be partially mitigated by setting boundaries on the "amount of emergence allowed". A simple implementation of such boundary utilizes the concept of emergence depth. The level of emergence depth is a property of a fact that is set to zero for all facts present in the beginning of the execution. Every time a new fact is created from application of an emergence rule to a set of facts, the emergence depth of the newly created fact is computed as one plus the maximum emergence depth of the facts in the left hand side of the rule. Thus, constraints can be set on the maximum level of emergence depth allowed on the facts of the left hand side of a rule in order for that rule to fire.
- **Unfamiliar programming language for many systems engineers**: Rules engines are typically functional, declarative languages that resemble CLIPS or PROLOG. These languages are different from the procedural languages with which most engineers are familiar. As a consequence, there might be a learning curve for users that are unfamiliar with such languages. This problem could be mitigating by extending the user interface to allow the user to enter rules and facts in "almost –natural" language (several parts of such interface have already been developed).

8.3.3 Modeling Limitations of the current EOSS Expert System

An expert system for architecting EOSS was created as a result of applying the rule-based system architecting framework to the EOSS domain. Significant effort has been put in the development of this expert system, and the mere knowledge base gathered deserves mention as a valuable contribution. However, the method and tool remain the result of one doctoral thesis, and thus can be improved in many ways. The following is a list of limitations and aspects that could be improved. Note in particular that all the typical limitations of rule-based expert systems discussed in Section 1.6.4 apply.

- **Consistency and completeness of the database:** A traditional critique of RBES consists in the inability in general to guarantee the completeness or even the consistency of the knowledge database. It is indeed possible that the rules in the knowledge base are not a good sampling of the actual body of knowledge. While this problem is also relatively easy to detect provided that a good explanation facility was developed, solving it remains a challenge in the field (Suwa et al., 1984).
- **Sources of uncertainty**: There are several potential sources of uncertainty in the results presented in this thesis: a) the list of measurement capabilities in each instrument's capability rules may be wrong or incomplete; b) the attributes of the same list may be wrong or incomplete; c) the values in the partial satisfaction requirement rules may be wrong or incomplete; d) the relative weights determined in the value aggregation rules may be inaccurate. As explained in the body of the paper, the numerical values used in the case study come from a variety of sources including the Decadal Survey report, interviews with senior scientists at MIT and NASA, and several other publications. While standard interview guidelines and rubrics were used, and significant effort was put into obtaining reasonable values, no formal social science method was utilized to elicit the information from the interviewees.
- **Needs unrelated to measurement requirements and data products**: An underlying assumption of the VASSAR methodology is that the vast majority of scientific and societal needs for EOSS can be expressed in terms of measurement requirements using the attributes defined in the measurement templates. While this is arguably the case for most system requirements that appear on formal mission documents, it is also true that there might be unwritten needs and objectives that are only marginally related to measurement requirements, such as those coming from international partners or other policy elements.
- **Mass budget models:** The rules that compute the mass budget of a spacecraft are based on empirical ratios between payload mass and subsystem mass, which are then completed with complexity penalties. Both ratios and complexity penalties are based on the data provided in the appendices of (Larson & Wertz, 1999b), which concerns a very small number of Earth observing satellites. These ratios and complexity penalties should be updated to reflect a larger number of Earth observing satellites.
- **Cost models**: In a similar note, the cost estimating relationships used in the cost estimation rules are based on data provided in (Apgar et al., 1999) and should be updated to include a larger set of Earth observing satellites.

- **Satellite configuration models**: In addition to computing the satellite mass, selecting a launch vehicle requires estimating the dimensions and a rough configuration of the spacecraft. The rules-of-thumb used to predict the size of the spacecraft are very basic and could be improved to contain a preliminary configuration model that places the instruments, the batteries, and the solar panels in the spacecraft.
- **Mission analysis database**: The mission analysis database contains detailed information about a few orbits in LEO, and the GEO orbit. While most useful inclinations and local times of the ascending nodes are captured, the database could be augmented with orbits at 500km and 700km. Furthermore, information concerning ground track repeat rates for LEO orbits should be incorporated, as the requirement for exact repeat rates appears often in the orbit design of Earth observing missions.
- **Technology investment model**: One of the issues identified in the mission scheduling problem is that this problem is heavily coupled to the technology investment problem. Having a model of the technology investment process that can simulate the increase in maturity of different technologies with time and money would add a lot of value.
- **Better coupling between problems**: This thesis found that there are important couplings between the instrument selection, packaging, and scheduling problems. Since these three problems were solved sequentially, this results in suboptimal algorithm performance. In order to alleviate this problem, iteration was performed manually. In particular, information was shared between problems in the form of additional rules (usually hard constraints). Although this approach was satisfactory for the purpose of this thesis, there might be a better approach to solving the global problem.

8.4 Opportunities for Further Research

This thesis is a first step towards a framework for architecting systems that incorporates a rule-based engine. Substantial progress has been made in demonstrating the utility of this approach. For example, it was shown that it is possible to separate the different bodies of knowledge required to solve a complex knowledge-intensive system architecting problem. This is very likely to result in an increased scalability and transparency of the tool, which was verified in the three case studies conducted in this thesis.

However, no statistical proofs of the superiority of such framework in terms of scalability and transparency were presented. Such proofs would require designing and conducting experiments with groups of people that would use this framework and other frameworks and assess their relative performance. These experiments were left out of the scope of this thesis, and therefore this thesis does not claim the corresponding hypotheses are proved.

In addition to this, several other potential areas of further research are highlighted below.

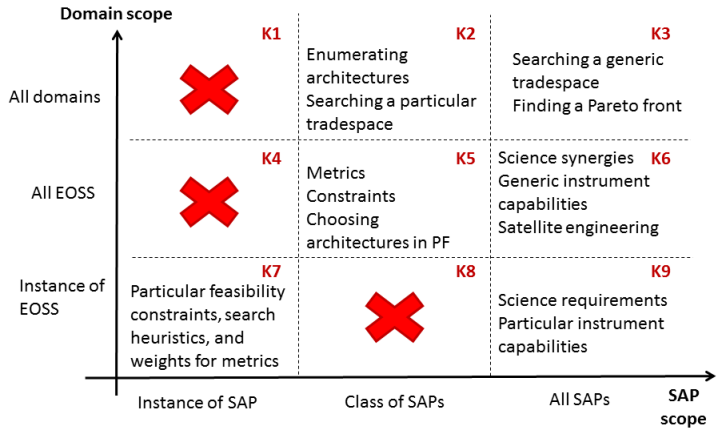
- **Robust expert knowledge elicitation**: One of the main limitations identified for the framework was in the process of knowledge elicitation. The methodology would potentially benefit from the incorporation of a formal social science method such as the Delphi method (Dalkey & Helmer, 1963) to elicit the information from stakeholders. However, the corresponding increase in rigor would have an additional price in consumption of resources, which would make the problem of the knowledge elicitation bottleneck worse.
- **Enhanced RBES with fuzzy set theory**: Furthermore, while our system provides a basic capability to deal with fuzzy attributes through a set of rules that transform back and forth from semi-quantitative to quantitative values, a more formal application of fuzzy set theory as first proposed by Zadeh (L. A. Zadeh, 1965) would certainly be beneficial to treat stakeholder satisfaction.
- **Augmenting the RBES with a machine learning layer**: The development of a machine learning layer on top of the fuzzy rule-based system would significantly improve its performance. This machine learning layer could be used for example to perform multi-attribute regression analysis on the mission analysis database to infer simple relationships that allow for fast prediction of the revisit times of a certain constellation on different regions of the Earth.
- **Modeling emergence in systems architecting using rules**: This thesis suggested that one of the potential advantages of using RBES would be a more natural way of modeling emergence. While this was illustrated for the case of EOSS, it would be interesting to further develop the theory of modeling emergence in systems architecting using rules. In particular, this approach could be compared to alternative ways of modeling emergence such as agent-based models, or cellular automata.
- **Incorporation of policy effects into system architecting using "Policy rules":** It was suggested by some of the interviewees that a policy set of rules be added to the current RBES in order to take into account important rules that might be driving EOSS architecture while being almost independent of science performance. While the VASSAR framework is general enough to accommodate such rules, this could potentially require the definition of new templates.

In addition to the previous points, the performance of any RBES is determined by the quantity and quality of rules in the rule database. Therefore, a major effort going forward will be in the continuous improvement of the different rule bases.

This thesis has opened new avenues of research in the utilization of rule engines to support system architecting. Many opportunities arise that span several engineering disciplines as well as computer science, engineering management, and social science.

9 Appendix: Knowledge database

This thesis separated domain-independent, domain-specific, and instance-specific knowledge as suggested in the diagram below, taken from the discussion Chapter 2.



The following additional materials are provided in this appendix:

1. **Domain-independent, SAP class-specific knowledge (K2)**
    a. Assigning problems
    b. Partitioning and covering problems
    c. Down-selecting problems
    d. Permuting problems
2. **Domain-specific, EOSS-independent SAP-independent knowledge (K6)**
    a. Orbit selection rules
    b. Power budget rules
    c. Complexity-corrected mass budget rules
    d. Launch vehicle selection rules
    e. Standard bus selection rules
3. **EOSS-specific knowledge (K9)**
    a. EOS case study value aggregation rules
    b. Decadal/GEOscan case study value aggregation rules

Requirement satisfaction rules and instrument capability rules for the EOS, Decadal, and GEOscan case studies are not provided in this Appendix, but they can be downloaded in xls format from:

http://web.mit.edu/dselva/www/RBES/

9.1 Domain-independent, SAP class-specific knowledge

9.1.1 Assigning Problems

9.1.1.1 Grammars

Encoding scheme:

```
(deftemplate DECISION (slot name) (multislot options))

(defrule create-architecture-template
    "This rule creates an architecture template that contains slots for each
decision"

    ?c <- (accumulate (bind ?list (create$ ))                          ;; initializer
                (bind ?list (insert$ ?list (+ 1 (length$ ?list)) ?n));; action
                ?list                                          ;; result
                (DECISION (name ?n))) ;; CE
    =>
    (bind ?str "(deftemplate DECISION-ARCH (slot num-decisions) (multislot decision-
names) (multislot sequence) (multislot str)")
    (foreach ?dec ?c
        (bind ?str (str-cat ?str "(slot " ?dec ")")))
    (bind ?str (str-cat ?str ")"))
    (eval ?str))

Example of utilization:
(reset)
(assert (DECISION (name num-boosters) (options (create$ 1 2 4))))
(assert (DECISION (name propellant1) (options (create$ LH2 CH4 RP1))))
(assert (DECISION (name propellant2) (options (create$ LH2 CH4 RP1))))
(run)

   ⇨ "(deftemplate DECISION-ARCH
      (slot num-decisions)
      (multislot decision-names)
      (multislot sequence)
      (multislot str)
      (slot num-boosters)
      (slot propellant1)
      (slot propellant2))"
```

Code 34: Automatic generation of architecture templates in the AsP from decisions and options

Note that the user only needs to assert what the decisions and are, as well as the options for each decision, and this simple rule will automatically generate a template that contains a slot for each architectural decision. In this template, the slot "sequence" contains the representation of the architecture in E_2 while the slot "str" contains the representation of the architecture in E_1.

Enumeration rules:

```
(defrule enumerate-all-architectures
    "This rule enumerates all possible assignments of options to decisions"
    ?arch <- (DECISION-ARCH (decision-names $?list) (num-decisions ?n) (sequence
$?seq) (str $?str) )
    (DECISION (name ?name) (options $?opt))
    (test (< (length$ ?seq) ?n))
    (test (eq ?name (nth$ (+ 1 (length$ ?seq)) ?list)))

    =>
    (retract ?arch)
    (bind ?i 1)
    (foreach ?value ?opt
        (bind ?new-seq (insert$ ?seq (+ 1 (length$ ?seq)) ?i))
        (bind ?new-str (insert$ ?str (+ 1 (length$ ?str)) ?value))
             (printout t ?new-seq crlf)
        (printout t ?new-str crlf)
        (assert (DECISION-ARCH (num-decisions ?n) (decision-names ?list) (sequence
?new-seq)
                (str ?new-str)))
        (++ ?i)
        ))

(defquery get-decisions (DECISION (name ?name)))

(deffunction get-decision-names ()
    (bind ?result (run-query* get-decisions))
    (bind ?list (create$ ))
    (while (?result next)
    (bind ?list (insert$ ?list (+ 1 (length$ ?list)) (?result getString name)))
        )
    (return (map eval ?list))
    )

Example of utilization:

(assert (DECISION-ARCH (num-decisions (count-query-results get-decisions)) (decision-
names (get-decision-names)) (sequence (create$ )) (str (create$ ))))
(run)
```

Code 35: Full factorial enumeration rule for the AsP and example of utilization

Since the size of the tradespace is exponential with the number of decisions, an alternative partial enumeration rule is required that only enumerates a fraction of the tradespace. Such enumeration rule is provided in Code 36. Note that only one command changes, namely the assert command inside the loop. In this enumeration rule, only a fraction of random options are used to populate the architectural tree. This fraction of architectures is controlled through the parameter of the function `rare-randb`. Hence in the example shown in Code 36, this fraction is 50%. For a value of this parameter equal to 100%, this enumeration rule is identical in behavior to the full factorial enumeration rule.

```
(deffunction rare-randb (?p)
    (return (< (random) (* ?p 65536)))
    )

(defrule HARD-CONSTRAINTS::enumerate-some-architectures
    "This rule enumerates a number of random AsP architectures. The number of
architectures can be controlled by changing the value of the parameter of the rare-
randb function."
    ?arch <- (HARD-CONSTRAINTS::DECISION-ARCH (decision-names $?list) (num-decisions
?n) (sequence $?seq) (str $?str) )
    (HARD-CONSTRAINTS::DECISION (name ?name) (options $?opt))
    (test (< (length$ ?seq) ?n))
    (test (eq ?name (nth$ (+ 1 (length$ ?seq)) ?list)))

    =>
    (retract ?arch)
    (bind ?i 1)
    (foreach ?value ?opt
        (bind ?new-seq (insert$ ?seq (+ 1 (length$ ?seq)) ?i))
        (bind ?new-str (insert$ ?str (+ 1 (length$ ?str)) ?value))
             (printout t ?new-seq crlf)
        (printout t ?new-str crlf)
        (if (rare-randb 0.5) then (assert (HARD-CONSTRAINTS::DECISION-ARCH (num-
decisions ?n) (decision-names ?list) (sequence ?new-seq)
                 (str ?new-str))))
        (++ ?i)))
```

Code 36: Partial enumeration rule for the AsP

9.1.1.2 Search heuristics

Local search: In terms of non-informed local search rules, we provide a mutation operator that changes the value of a random decision to a new random value. The code is provided in Code 37.

```
(defquery HARD-CONSTRAINTS::get-options
    (declare (variables ?name))
    (HARD-CONSTRAINTS::DECISION (name ?name) (options ?options)))

(deffunction get-option-values (?dec-name)
    (bind ?result (run-query* HARD-CONSTRAINTS::get-options))
    (while (?result next)
        (bind ?list (?result getSymbol options))
        )
    (return ?list)
    )

(defrule mutation-change-one-assignment
    "This mutation function swaps the value of a single bit"
    ?arch <- (HARD-CONSTRAINTS::DECISION-ARCH (decisions $?dec) (num-decisions ?n))
    =>
    (bind ?dec-name (nth$ (rand-int-1-to-N ?n) (get-decision-names)))
    (bind ?options (get-option-values ?dec-name))
    (bind ?new-val (nth$ (rand-int-1-to-N (length$ ?options)) ?options))
    (eval (str-cat "(duplicate ?arch (" ?dec-name "?new-val))" )))
```

Code 37: A mutation operator for the AsP

A single point crossover operator is also included that combines decisions from the "father" with decisions from the "mother" in order to produce new and hopefully better "children" architectures. The code of such crossover operator is given in Code 38.

```
(deffunction AsP-crossover (?dad ?mom)
    "This function performs a single point crossover between
    two AsP architectures"
    (bind ?m (round (/ (length$ ?dad) 2)))
    (return (create$ (subseq$ ?dad 1 ?m) (subseq$ ?mom (+ 1 ?m) ?m)))
    )

(defrule AsP-single-point-crossover
    "This crossover operator combined assignments from two good architectures
    to create a new and hopefully better architecture by combining decisions
    from the two parent architectures"

    ?dad <- (HARD-CONSTRAINTS::DECISION-ARCH (decisions $?dec) (num-decisions ?n)
(sequence $?seq1)  (improve yes))
    ?mom <- (HARD-CONSTRAINTS::DECISION-ARCH (decisions $?dec) (num-decisions ?n)
(sequence $?seq1) (improve yes))
    =>
    (bind ?new-seq (AsP-crossover ?seq1 ?seq2))
    (assert (HARD-CONSTRAINTS::DECISION-ARCH (num-decisions ?n) (decisions ?dec)
            (sequence ?new-seq)))
    )
```

Code 38: A crossover operator for the AsP

9.1.2 Partitioning and Covering Problems

9.1.2.1 Grammars

Enumeration rules: Under the E_1 encoding scheme, we can implement a full factorial enumeration rule that leverages the recursivity provided by functional languages such as CLIPS:

```
(deftemplate partitioning-architecture (multislot assignments))
(deffunction max$ (?list)
    (bind ?size (length$ ?list))
    (bind ?mx 0)
    (for (bind ?i 1) (<= ?i ?size) (++ ?i)
        (bind ?el (nth$ ?i ?list))
      (if (> ?el ?mx) then (bind ?mx ?el))
       )
    (return ?mx)
    )

(defrule grammar-A-set-partitioning
    "This enumeration rule generates all valid architectures for a generic set
partitioning problem with five elements"
    ?arch <- (partitioning-architecture (assignments $?ass))
      (test (< (length$ ?ass) 5))
    =>
```

```
    (retract ?arch)
    (bind ?n (length$ ?ass))
    (bind ?mx (max$ ?ass)) ; index of current last element of form
    (for (bind ?i 1) (<= ?i (+ ?mx 1)) (++ ?i)
        (bind ?new-ass (insert$ ?ass (+ ?n 1) ?i))
        (assert (partitioning-architecture (assignments ?new-ass)))
        )
    )
(reset)
(assert (partitioning-architecture (assignments (create$ 1))))
(run)
```

Code 39: Efficient full factorial enumeration rule for PaPs

Let R_1 be the enumeration rule defined in Code 39, and let unconstrained grammar $A = \langle E, \phi, R_1 \rangle$. While this rule is very efficient in enumerating all possible set partitions, full enumeration becomes impossible for numbers of elements between 10 and 13 for current computing technology, because Bell numbers grow faster than exponentially. Thus a different grammar is required that allows for non-exhaustive exploration.

The following enumeration rule uses the same encoding scheme as, and is very similar to, the previous one. However, in this enumeration rule, only a certain number of architectures are randomly generated. The number of architectures generated is "controlled" in Code 40 through the variable *?N*. Note that "controlled" in this context means that the number of architectures generated will be exponential with the number of elements, as opposed to faster than exponential in the original case. The base of this exponential will be equal to the average number of architectures generated by iteration, which for a uniform random function will be close to $0.5(N + 1)$.

```
;; Grammar B for a function partitioning architecture problem
(deftemplate TEST::partitioning-architecture (multislot assignments))
(deffunction max$ (?list)
    (bind ?size (length$ ?list))
    (bind ?mx 0)
    (for (bind ?i 1) (<= ?i ?size) (++ ?i)
        (bind ?el (nth$ ?i ?list))
      (if (> ?el ?mx) then (bind ?mx ?el))
        )
    (return ?mx)
    )

(deffunction random_int (?min ?max)
    (bind ?rnd (* (/ (random) 65536) (- (+ 0.5 ?max) (- ?min 0.5)))); random real
between 0 and max - min + 1
    (bind ?x (+ (- ?min 0.5) ?rnd)); random real between min - 0.5 and max + 0.5
    (return (round ?x))
    )
```

```
(defrule TEST::grammar-B-set-partitioning
    "This grammar generates only a random subset of architectures for a generic set
partitioning problem with 10 elements"

    ?arch <- (TEST::partitioning-architecture (assignments $?ass))
      (test (< (length$ ?ass) 10))
    =>
    (bind ?N 4) ; this parameter controls the number of architectures enumerated
    (retract ?arch)
    (bind ?n (length$ ?ass))
    (bind ?mx (max$ ?ass)) ; index of current last element of form
    (for (bind ?i 1) (<= ?i ?N) (++ ?i) ; only loop N times
        (bind ?new-ass (insert$ ?ass (+ ?n 1) (random_int 1 (+ ?mx 1)))); insert
random number between 1 and max + 1
        (assert (TEST::partitioning-architecture (assignments ?new-ass)))
        )
    )
```

Code 40: Controlled partial exploration of the architectural tradespace of PaPs

9.1.2.2 Search heuristics

Tradespace size reduction rules:

- A rule on the maximum number of subsets that a partition can have.

```
(deftemplate TEST::PACK-ARCH (multislot assignments))

(defrule TEST::delete-archs-with-too-many-subsets
    (TEST::MAX-SATS (max-sats# ?mx&~nil))
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    (test (> (max$ ?ass) ?mx))
    =>
    (retract ?arch))
```

Code 41: Tradespace size reduction search heuristic rule for PaPs on max number of subsets in a partition

- A rule on the maximum number of elements that any subset in a partition can have. This avoids the exploration of architectures that would include subsets that are non-sensical for being too large. For example, if the elements are instruments and the subsets are satellites, this avoids looking at architectures that have satellites that cannot be launched by any current launch vehicle.

```
(defrule TEST::delete-archs-with-too-many-elements-in-one-subset
    (TEST::MAX-ELEMS-PER-SUBSET (max-elements-per-subset# ?mx&~nil))
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    (test (> (get_max_elems_per_subset $?ass) ?mx))
    =>
    (retract ?arch))
```

Code 42 Tradespace size reduction search heuristic rule for PaPs on max number of subsets in a partition

- One or more rules to force that an instrument is assigned to a subset of size k (typically, k = 1). This dramatically reduces the size of the tradespace. For k = 1 for example, this is equivalent to considering a PaP with m-1 elements.

```
(defrule TEST::delete-archs-breaking-alone-element-reqs
    (TEST::ALONE-ELEMENTS (elements $?ins))
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    (test (break_alone_req ?ins ?ass))
    =>
    (retract ?arch))
```

Code 43: Tradespace size reduction search heuristic rule for PaPs that requires an element to be assigned to a 1-element subset.

- One or more rules to enforce that two or more elements are grouped in the same subset.

```
(defrule TEST::delete-archs-breaking-together-elems-reqs
    (TEST::TOGETHER-ELEMENTS (elements $?ins))
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    (test (break_together_req ?ins ?ass))
    =>
    (retract ?arch))
```

Code 44: Tradespace size reduction search heuristic rule for PaPs that requires two elements to be assigned to the same subset.

- One or more rules to enforce that two or more instruments are assigned to different subsets.

```
(defrule TEST::delete-archs-breaking-apart-instrument-reqs
    (TEST::APART-INSTRUMENTS (instruments $?ins))
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    (test (break_apart_req ?ins ?ass))
    =>
    (retract ?arch)
    )
```

Code 45: Tradespace size reduction search heuristic rule for PaPs that requires two elements to be assigned to different subsets.

In addition to these rules, decomposition of PaPs can be accomplished using any clustering algorithm. In order to use a clustering algorithm it is necessary to provide a metric for the distance between two elements: two elements that are "close" together using this distance tend to be assigned to the same cluster, and they tend to be assigned to different clusters if this distance is "large". Typically, this distance is the Euclidean distance between the vectors representing the element on a certain space. Such space could be forever a space of architectural attributes of the elements. However, in some PaPs, this may result in poor heuristics, since there is little correlation in general between similarity in the space of architectural attributes and synergies between elements.

Local search rules: Six different local search rules are defined, growing in level of information from completely uninformed random mutation operators to informed specialized operators that utilize the V-DSM. These six rules are described below:

- A mutation operator that changes the position of one random element into a random subset different from its original subset

```
(defrule TEST::mutation-change-one-element
    "This mutation function changes the position of one element"
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    =>
    (bind ?new-arch (mutate_swap_one_element ?ass))
    (duplicate ?arch (assignments ?new-arch))
    )
```

Code 46: Mutation operator for PaPs that changes the position of a random element into a new random subset.

- A mutation operator that swaps the positions of two random elements taken from different random subsets.

```
(defrule TEST::mutation-swaps-two-elements
    "This mutation function swaps the positions of two elements"
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    =>
    (bind ?new-arch (mutate_swap_two_elements ?ass))
    (duplicate ?arch (assignments ?new-arch))
)
```

Code 47: Mutation operator for PaPs that changes the position of two random elements from different subsets.

- A mutation operator that breaks a random big subset into two smaller subsets.

```
(defrule TEST::mutation-break-one-big-subset
    "This mutation function breaks one big random subset"
    ?arch <- (TEST::PACK-ARCH (assignments $?ass) (mutate yes))
    =>
    (bind ?new-arch (mutate_break_big_subset ?ass))
    (duplicate ?arch (assignments ?new-arch))
    )
```

Code 48: Mutation operator for PaPs that breaks one random large subset in a partition into two smaller subsets.

- A mutation operator that combines two random "small" subsets to create a larger subset.

```
(defrule TEST::mutation-combine-two-small-subsets
    "This mutation function combines two small random subsets"
    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    =>
    (bind ?new-arch (mutate_combine_small_subsets ?ass))
    (duplicate ?arch (assignments ?new-arch))
    )
```

Code 49: Mutation operator for PaPs that combines two random small subsets to create a new larger subset.

- Improve by adding synergies: This heuristic rule identifies the missing synergies in an architecture, selects one of these missing synergies randomly, and swaps the position of two elements in order to capture that synergy. This rule is the informed version of *mutation-combine-two-small-subsets*.

```
(defrule TEST::improve-by-adding-synergy
    "This heuristic search rule improves a packaging architecture by
identifying a synergy that is not captured and rearranging the
assignments so that the synergy is captured"

    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    =>
    (bind ?new-arch (matlabf improve_add_synergy ?ass))
    (duplicate ?arch (assignments ?new-arch))
    )
```

Code 50: Informed local search operator for PaPs that changes the position of two elements from different subsets intelligently in order to capture a missing synergy.

1. Improve by eliminating interference: This heuristic rule identifies a current interference in an architecture, and swaps the position of two elements in order to break that interference. Note that this rule is the informed version of `mutation-break-one-big-subset.`

```
(defrule TEST::improve-by-removing-interference
    "This heuristic search rule improves a packaging architecture by
identifying an existing interference and removing it"

    ?arch <- (TEST::PACK-ARCH (assignments $?ass))
    =>
    (bind ?new-arch (matlabf improve_remove_interference ?ass))
    (bind ?new-str (matlabf get_str_from_arch ?new-arch))
    (duplicate ?arch (assignments ?new-arch) (improve no) (str ?new-
str))
    )
```

Code 51: Informed local search operator for PaPs that changes the position of two elements from different subsets intelligently in order to capture a missing synergy.

The last informed search heuristic that we provide for PaPs is a crossover operator. Crossover operators combine characteristics of two architectures to create a new architecture. This crossover operator takes some subsets from one of the parents and then assigns the rest of the elements to the other. Its CLIPS code is provided below.

```
%%% Matlab PACK_crossover.m
function xoverKids  =
PACK_crossover(parents,options,GenomeLength,FitnessFcn,unused,thisPopulation)
nKids = length(parents)/2;
linCon = options.LinearConstr;
constr = ~isequal(linCon.type,'unconstrained');
xoverKids = zeros(nKids,GenomeLength);
index = 1;
```

```
for i=1:nKids
    % get parents
    papa = thisPopulation(parents(index),:);
    index = index + 1;
    mama = thisPopulation(parents(index),:);
    index = index + 1;
    %% Compute children from 2 parents in sats form
    first_half = papa(1:ceil(GenomeLength/2));
    first_half_papa = PACK_arch2sats(first_half);%these are length
(first_half_papa) sats
    nsat1 = length(first_half_papa);
    second_half =  mama(ceil(GenomeLength/2)+1:end);
    second_half_mama =
PACK_arch2sats(PACK_fix(second_half),ceil(GenomeLength/2));
    second_half =  PACK_fix(second_half) + max(first_half);
    existing_sat = mama(ceil(GenomeLength/2)+1:end) <=
max(mama(1:ceil(GenomeLength/2))); %existing_sat(i) = 1 means that instrument
i is in a satellite that was created in the first half
    nsat2 = length(second_half_mama);
    % Try to combine satellites of second half with satellite of first half
    kid_sat = [first_half_papa;second_half_mama];
    for j = 1:nsat2 % foreach satellite from mum
        sat = second_half_mama{j}; % satellite
        for k = 1:length(sat) % loop through its instruments
            instr = sat(k);% instr
            if existing_sat(instr-ceil(GenomeLength/2)) % if this instrument
in mum belongs to a satellite of the 1st half
                % lump this satellite with a random satellite from first half
                % (which comes from dad)
                kid_sat = merge_sats(kid_sat,randi(nsat1),j+nsat1);% merge
satellites random and this one
                break;% go to next satellite in loop over j
            end
        end
    end
    xoverKids(i,:) = PACK_fix(PACK_sats2arch(kid_sat));
    % Make sure that offspring are feasible w.r.t. linear constraints
    if constr
        feasible  =
isTrialFeasible(xoverKids(i,:)',linCon.Aineq,linCon.bineq,linCon.Aeq, ...
            linCon.beq,linCon.lb,linCon.ub,sqrt(options.TolCon));
        if ~feasible % Kid is not feasible
            % Children are arithmetic mean of two parents (feasible w.r.t
            % linear constraints)
            alpha = rand;
            xoverKids(i,:) = round(alpha*thisPopulation(r1,:) + ...
                (1-alpha)*thisPopulation(r2,:));
        end
    end
end
```

Code 52: A crossover operator for PaPs.

Repair operators: A repair operator is provided that projects a non-sensical architecture representation into the set of sensical architecture representations. The Matlab code for a generic repair operator for PaPs is provided below.

```
function new = PACK_fix(seq)
new = seq;

%% relabel subset indices in increasing order starting from 1
pivs = java.util.HashMap;
for i = 1:length(seq)
    piv = seq(i);
    if pivs.containsKey(piv)
        new(i) = pivs.get(piv);
    else
        n = pivs.size;
        pivs.put(piv,n + 1);
        new(i) = n+1;
    end
end

%% remove empty satellites
for i = 2:length(new)
    if(new(i) > max(new(1:i-1)) + 1)
        new(i) = max(new(1:i-1)) + 1;
    end
end
end
```

Code 53: Matlab code for a generic repair operator for PaPs.

9.1.3 Down-selecting Problems

9.1.3.1 Grammars

Enumeration rules: Given this encoding scheme E_1, a simple set of enumeration rules R for the unconstrained DsP is provided in Code 54.

```
(deftemplate TEST::SEL-ARCH (multislot sequence)); template for encoding scheme

(defrule TEST::enumerate-all-subsets
    "This rule enumerates all possible subsets in a set of m elements"
    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
      (test (< (length$ $?seq) 5))
    =>
    (retract ?arch)
    (bind ?n (length$ $?seq))
    (bind ?new-seq (insert$ ?seq (+ ?n 1) 0))
    (assert (TEST::SEL-ARCH  (sequence ?new-seq)))
    (bind ?new-seq (insert$ ?seq (+ ?n 1) 1))
    (assert (TEST::SEL-ARCH  (sequence ?new-seq)))
    )
```

Code 54: Enumeration rule for the unconstrained DsP

Beyond 30 elements, full factorial enumeration becomes impossible and an alternative grammar that enumerates only a fraction of the tradespace becomes necessary. Such grammar is provided in Code 55.

```
(deffunction rare-randb ()
    (bind ?P 1000)
    (return (< (random) ?P)))
(defrule TEST::enumerate-some-subsets
    "This rule enumerates all possible subsets in a set of m elements"
    ?arch <- (TEST::SEL-ARCH (sequence $?seq) (stop-tree FALSE))
      (test (< (length$ $?seq) 15))
    =>
    (retract ?arch)
    (bind ?n (length$ $?seq))
    (bind ?new-seq (insert$ ?seq (+ ?n 1) 0))
    (assert (TEST::SEL-ARCH  (sequence ?new-seq) (stop-tree (rare-randb))))
    (bind ?new-seq (insert$ ?seq (+ ?n 1) 1))
    (assert (TEST::SEL-ARCH  (sequence ?new-seq) (stop-tree (rare-randb)))))
(defrule TEST::remove-incomplete-arch
    "This rule enumerates all possible subsets in a set of m elements"
    (declare (salience -10))
    ?arch <- (TEST::SEL-ARCH (sequence $?seq) (stop-tree 1));
      (test (< (length$ $?seq) 15))
    =>
    (retract ?arch)   )
```

Code 55: CLIPS code for a partial enumeration grammar for the DsP

Enumeration constraints: We now consider four types of generic enumeration constraints for the DsP, divided in two opposite pairs:

- $OR(\{e_i\})$ enumeration constraints: this enumeration rule eliminates all subsets in which **none** of elements in $\{e_i\}$ are selected. One possible natural language equivalent of this rule is "AT LEAST ONE FROM". In logic, this is called a disjunction. It is useful to define a situation in which elements belong to groups, and at least one element of each group is required.
- $NAND\{e_i\}$ enumeration constraints: this enumeration rule eliminates all subsets in which **all** of elements in $\{e_i\}$ are selected. Note that: $NAND = \neg OR$. One possible natural language equivalent of this rule is "NOT ALL OF". In logic, this is called an alternative denial. It is useful to define a situation in which two elements are redundant.
- $XOR(\{e_i\})$ enumeration constraints: this enumeration rule eliminates all subsets in which **either more or less than one** elements in $\{e_i\}$ are selected. Note that for m = 2 only: $XOR = OR \wedge NAND$. One possible natural language equivalent of this rule is "EXACTLY ONE FROM". This is useful to define a situation in which elements belong to groups, and at least one element of each group is required, but it makes no sense to take more than 1 element from each group.

- $XNOR(\{e_i\})$ enumeration constraints: this enumeration rule eliminates all subsets **except those in which either all or none** of elements in $\{e_i\}$ are selected. Note that for m = 2 only: $XNOR = \neg XOR$. One possible natural language equivalent of this rule is "EITHER ALL OR NONE OF". This is useful to define a situation in which it really only makes sense to select certain elements as a group, so that the decision to make is whether to select the group or not. Note that this constraint reduces the size of the problem n -

The implementation of these constraints in CLIPS is provided below. Note that a different template is defined for each type of enumeration constraint. The code for a few supporting functions is also shown.

```
(deftemplate TEST::OR-CONSTRAINT (multislot elements))
(deftemplate TEST::NAND-CONSTRAINT (multislot elements))
(deftemplate TEST::XOR-CONSTRAINT (multislot elements))
(deftemplate TEST::XNOR-CONSTRAINT (multislot elements))

(deffunction bit-and$ (?list1 ?list2)
    (bind ?i 0)
    (bind ?res (create$ ))
    (foreach ?bit1 ?list1
        (bind ?res (insert$ ?res (++ ?i) (bit-and ?bit1 (nth$ ?i ?list2)))))
    (return ?res)
    )

(deffunction bit-or$ (?list1 ?list2)
    (bind ?i 0)
    (bind ?res (create$ ))
    (foreach ?bit1 ?list1
        (bind ?res (insert$ ?res (++ ?i) (bit-or ?bit1 (nth$ ?i ?list2)))))
    (return ?res)
    )

(deffunction +$ (?list)
    (if (eq (length$ ?list) 1) then (return (nth$ 1 ?list))
        else (return (+ (nth$ 1 ?list) (+$ (rest$ ?list)))))
    )

(deffunction get-mask (?elems ?N)
    (bind ?res (create$ ))
    (for (bind ?i 1) (<= ?i ?N) (++ ?i)
        (if (numberp (member$ ?i ?elems)) then
            (bind ?res (insert$ ?res ?i 1))
        else
            (bind ?res (insert$ ?res ?i 0)))
        )
    (return ?res)
    )
(defrule TEST::enforce-OR-constraints
    "If an architecture does not contain at least one of the elements in an or
constraint
    then it should be deleted"
```

```
    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
    (TEST::OR-CONSTRAINT (elements $?elems))
    (test (< (+$ (bit-and$ ?seq (get-mask $?elems 4))) 1))
    =>

    (retract ?arch)
    )

(defrule TEST::enforce-NAND-constraints
    "If an architecture contains all of the elements in a NAND constraint
    then it should be deleted"

    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
    (TEST::NAND-CONSTRAINT (elements $?elems))
    (test (eq (+$ (bit-and$ ?seq (get-mask $?elems 4))) (length$ ?elems)))
    =>

    (retract ?arch)
    )

(defrule TEST::enforce-XOR-constraints
    "If an architecture does not contain exactly one of the elements in a XOR
constraint
    then it should be deleted"

    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
    (TEST::XOR-CONSTRAINT (elements $?elems))
    (test (neq (+$ (bit-and$ ?seq (get-mask $?elems 4))) 1))
    =>

    (retract ?arch)
    )

(defrule TEST::enforce-XNOR-constraints
    "If an architecture contains a number of the elements in a XNOR constraint
    that is > 0 and less than all of them, then it should be deleted"

    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
    (TEST::XNOR-CONSTRAINT (elements $?elems))
    (test (> (+$ (bit-and$ ?seq (get-mask $?elems 4))) 0))
    (test (< (+$ (bit-and$ ?seq (get-mask $?elems 4))) (length$ ?elems)))
    =>

    (retract ?arch)
    )
```

Code 56: Canonical enumeration constraints for DsPs

Note that this list of enumeration constraints is not exhaustive. One could consider for instance an "EXACTLY N OUT OF K" constraint in which exactly N elements from a subset of K elements need to be selected. For K = 2 and N = 1 this is equivalent to the XOR rule. Another useful rule could be "AT LEAST N OUT OF K". The implementation of such constraints from combinations or variations of the others is trivial and shown below:

```
(deftemplate TEST::EXACTLY-N-OUT-OF-K-CONSTRAINT (multislot elements) (slot N))
(deftemplate TEST::AT-LEAST-N-OUT-OF-K-CONSTRAINT (multislot elements) (slot N))

(defrule TEST::enforce-EXACTLY-N-OUT-OF-K-constraints
    "If an architecture contains a number of the elements in an EXACTLY-N-OUT-OF-K
constraint
    that is not equal to N, then it should be deleted"

    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
    (TEST::EXACTLY-N-OUT-OF-K-CONSTRAINT (elements $?elems) (N ?N))
    (test (neq (+$ (bit-and$ ?seq (get-mask $?elems 4))) ?N))
    =>

    (retract ?arch))

(defrule TEST::enforce-AT-LEAST-N-OUT-OF-K-constraints
    "If an architecture contains a number of the elements in an AT-LEAST-N-OUT-OF-K
constraint
    that is less than N, then it should be deleted"

    ?arch <- (TEST::SEL-ARCH (sequence $?seq))
    (TEST::AT-LEAST-N-OUT-OF-K-CONSTRAINT (elements $?elems) (N ?N))
    (test (< (+$ (bit-and$ ?seq (get-mask $?elems 4))) ?N))
    =>

    (retract ?arch))
```

Code 57: Additional enumeration constraints for the DsP

9.1.3.2 Search heuristics

Local search rules: We propose one non-informed local search rule in the form of a mutation operator that swaps the value of a bit randomly.

```
(defrule TEST::mutation-swap-one-bit
    "This mutation function generates N new architectures from the original one by
swapping the value of a single random bit"
    ?arch <- (TEST::SEL-ARCH (sequence ?orig))
    =>
    (bind ?N 5)
    (for (bind ?i 0) (< ?i ?N) (++ ?i)
          (bind ?new-seq (mutate_one_bit ?orig))
      (duplicate ?arch (sequence ?new-seq))
    )
```

Code 58: A non-informed mutation operator for the DsP

Note that the variable *N* allows the definition of the number of new architectures that are generated each time this mutation function is executed.

In addition to this non-informed mutation operator, we provide two informed local search rules that utilize the B-DSM defined in section 9.1.2.2: the first one attempts to improve the subset by completing it with an element that has high synergies with the elements already in the subset; the second one attempts to improve it by removing a redundant element from the subset.

```
(defrule TEST::improve-by-completing-architecture
    "This rule identifies the shortcomings of an architecture and
    modifies it by adding an instrument that covers some of the gaps
    "
     ?arch <- (TEST::SEL-ARCH (sequence ?orig))
    =>
    (bind ?new-seq (matlabf complete_arch ?orig)); this is an ArrayList of objectives
    (if (neq ?new-seq ?orig) then (duplicate ?arch (sequence ?new-seq))
    )

(defrule TEST::improve-by-removing-redundancy
    "This rule identifies the potential redundancies in an architecture and
eliminates
    one of them"

     ?arch <- (TEST::SEL-ARCH (sequence ?orig))
    =>
    (bind ?new-seq (matlabf remove_redundancy_from_arch ?orig)); this is an ArrayList
of objectives
    (duplicate ?arch (sequence ?new-seq))
    )
```

Code 59: Two informed mutation operators for the DsP that utilize the B-DSM

9.1.4 Permuting Problems

9.1.4.1 Grammars

Encoding scheme: A straightforward encoding scheme for the PeP is an array of m integers between 1 and m:

$$E: O_i \rightarrow [p_1, p_2, \dots, p_m]$$

$$p_j \in [1, m]$$

Two different encoding schemes can be defined depending on the meaning of the p_j:

- $E_1: p_j = k$ means that element $e_j$is assigned to position k in O_i
- $E_2: p_j = k$ means that element $e_k$is assigned to position j in O_i

Both encoding schemes are used in this thesis, since some rules are easier to implement under E_1, while others are easier to implement under E_2. Hence, there is a need for an operator that changes from $E_1$to E_2. The code for such operator is given below, together with the template that uses the two different encodings. E_1 was arbitrarily named the sequence, and E_2 the ordering.

```
(deftemplate TEST::PERMUTING-ARCH (multislot sequence) (multislot ordering))

(deffunction sequence-to-ordering (?seq)
    (bind ?ord (create$ ))
    (foreach ?x ?seq (bind ?ord (insert$ ?ord (+ (length$ ?ord) 1) 0)))
    (for (bind ?i 1) (<= ?i (length$ ?seq)) (++ ?i)
        (bind ?j (nth$ ?i ?seq))
    (bind ?ord (replace$ ?ord ?j ?j ?i))
        )
    (return ?ord)
    )
```

Code 60: Template for a permuting architecture and operator to change between E1 (sequence of mission) and E2 (mission orders) in PeP

In order to differentiate between these two different encoding schemes, we will arbitrarily use O_i to indicate the representation of a permutation in E_1, while $O_i{}'$ will indicate the representation of the same permutation in E_2. For instance, if $O_i = [2,4,5,1,3]$ then $O_i{}' = [4,1,5,2,3]$. In this permutation, the sequence is element 2 – element 4 – element 5 – element 1 – element 3. If we want to know the position of e_j in the sequence we can also look at the jth entry in $O_i{}'$.

Enumeration rules: Given $E_1$and E_2, the enumeration rule given in Code 61 can be used to enumerate all possible permutations of m elements.

Since *factorial(m)* grows very fast with *m*, it is required to define an alternative grammar that only enumerates part of the tradespace. This alternative grammar is presented in Code 62.

```
(deftemplate TEST::PERMUTING-ARCH (multislot sequence))

(defrule TEST::enumerate-all-permutations
    "This rule enumerates all possible permutations of a set of m elements"
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (test (< (length$ ?seq) 5))
    =>
    (retract ?arch)
    (bind ?n (length$ ?seq))
    (for (bind ?i 1) (<= ?i 5) (++ ?i)
        (if (eq (member$ ?i ?seq) FALSE) then
            (bind ?new-seq (insert$ ?seq (+ ?n 1) ?i))
        (assert (TEST::PERMUTING-ARCH (sequence ?new-seq)))
            ))
    )
```

Code 61: A grammar for full factorial enumeration of all permutations in the unconstrained PeP

```
(deffunction rare-randb (?p)
    (return (< (random) (* ?p 65536)))
    )

(defrule TEST::enumerate-some-permutations
    "This rule enumerates some random permutations of a set of m elements"
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (test (< (length$ ?seq) 8))
    =>
    (retract ?arch)
    (bind ?n (length$ ?seq))
    (for (bind ?i 1) (<= ?i 8) (++ ?i)
        (if (eq (member$ ?i ?seq) FALSE) then
            (bind ?new-seq (insert$ ?seq (+ ?n 1) ?i))
        (if (rare-randb 0.9) then (assert (TEST::PERMUTING-ARCH (sequence ?new-
seq))))
            )
        )
    )
```

Code 62: A grammar for partial enumeration of some permutations in the unconstrained PeP

Enumeration constraints: In the planning instances of PePs in which, in addition to sequence, dates are assigned to tasks or events, other constraints may appear that make explicit reference to dates instead of relative positions. These rules require an operator that transforms a sequence of tasks with costs into a sequence of dates, given a certain budget profile. Such operator is provided below.

```
(deftemplate TEST::BUDGET (multislot budgets) (multislot years))
(deftemplate TEST::COSTS (multislot elements) (multislot costs))

(defquery TEST::search-budget
    (TEST::BUDGET (budgets ?budgets) (years ?years))
    )

(defquery TEST::search-costs
    (TEST::COSTS (elements ?elems) (costs ?costs))
    )

(deffunction get-budgets ()
    (bind ?result (run-query* TEST::search-budget))
    (?result ?next)
    (return (?result getSymbol budgets))
    )

(deffunction get-years ()
    (bind ?result (run-query* TEST::search-budget))
    (?result ?next)
    (return (?result getSymbol years))
    )

(deffunction get-costs ()
    (bind ?result (run-query* TEST::search-costs))
    (?result ?next)
```

```
    (return (?result getSymbol costs))
    )
(deffunction sequence-launch-dates (?seq)
    (bind ?budgets (get-budgets))
    (bind ?years (get-years))
    (bind ?costs (get-costs))
    (bind ?dates (create$ ))
    (bind ?start-date (nth$ 1 ?years))
    (bind ?date ?start-date)
    (for (bind ?i 1) (< ?i (length$ ?costs)) (++ ?i)
        (bind ?date (+ ?date (/ (nth$ ?i ?costs) (nth$ (round (- ?date ?start-date))
?budgets))))
        (bind ?dates (insert$ ?dates ?i ?date))
        )
    (return ?dates)
    )
```

Code 63: Operator that computes dates from budget and cost data for the PeP problem

This function assumes that readiness date for element *i + 1* is equal to readiness date for element *i* plus the cost of the element *i* divided by the budget at time t.

We provide seven possible hard enumeration constraints in generic PePs: BEFORE, AFTER, BETWEEN, NOT-BETWEEN, CONTIGUOUS, NON-CONTIGUOUS, and SUBSEQUENCE constraints. All of these utilize a few generic functions to check precedence between elements in a sequence or ordering. These functions are provided in Code 64. Note that these hard constraints can also be used as search rules with the purpose of reducing the size of the tradespace. Furthermore, equivalent rules can be defined that look at relative dates of elements instead of looking at positions. The code of these rules is almost identical to the rules shown below and therefore it is not included.

```
(deffunction check-precedence-in-ordering-binary (?el1 ?el2 ?ord)
    "returns TRUE if ?el1 goes before ?el2 and FALSE otherwise"
    (return (< (nth$ ?el1 ?ord) (nth$ ?el2 ?ord))))

(deffunction check-precedence-in-sequence-binary (?el1 ?el2 ?seq)
    "returns TRUE if ?el1 goes before ?el2 and FALSE otherwise"
    (bind ?ord (sequence-to-ordering ?seq))
    (return (check-precedence-in-ordering-binary ?el1 ?el2 ?ord)))

(deffunction check-succession-in-ordering-binary (?el1 ?el2 ?ord)
    "returns TRUE if ?el1 goes after ?el2 and FALSE otherwise"
    (return (not (check-precedence-in-ordering-binary ?el1 ?el2 ?ord))))

(deffunction check-succession-in-sequence-binary (?el1 ?el2 ?seq)
    "returns TRUE if ?el1 goes after ?el2 and FALSE otherwise"
    (return (not (check-precedence-in-sequence-binary ?el1 ?el2 ?seq))))

(deffunction check-precedence-in-ordering (?el1 ?list ?ord)
    "returns TRUE if ?el1 goes before all elements in ?el2 and FALSE otherwise"
    (if (eq (listp ?list) FALSE) then
```

```
        (return (check-precedence-in-ordering-binary ?el1 ?list ?ord)))
    (if (eq (length$ ?list) 1) then
        (bind ?el2 (nth$ 1 ?list) )
        (return (check-precedence-in-ordering-binary ?el1 ?el2 ?ord))
        else
        (bind ?el2 (nth$ 1 ?list))
        (bind ?res (check-precedence-in-ordering-binary ?el1 ?el2 ?ord))
        (if (eq ?res FALSE ) then (return FALSE) else
            (return (check-precedence-in-ordering ?el1 (rest$ ?list) ?ord)))
        ))

(deffunction check-precedence-in-sequence (?el1 ?list ?seq)
    "returns TRUE if ?el1 goes before ?el2 and FALSE otherwise"
    (bind ?ord (sequence-to-ordering ?seq))
    (return (check-precedence-in-ordering  ?el1 ?list ?ord)))

(deffunction check-succession-in-ordering (?el1 ?list ?ord)
    "returns TRUE if ?el1 goes after all elements in ?el2 and FALSE otherwise"
    (if (eq (listp ?list) FALSE) then
        (return (check-succession-in-ordering-binary ?el1 ?list ?ord)))
    (if (eq (length$ ?list) 1) then
        (bind ?el2 (nth$ 1 ?list) )
        (return (check-succession-in-ordering-binary ?el1 ?el2 ?ord))
        else
        (bind ?el2 (nth$ 1 ?list))
        (bind ?res (check-succession-in-ordering-binary ?el1 ?el2 ?ord))
        (if (eq ?res FALSE ) then (return FALSE) else
            (return (check-succession-in-ordering ?el1 (rest$ ?list) ?ord)))
        ))

(deffunction check-succession-in-sequence (?el1 ?list ?seq)
    "returns TRUE if ?el1 goes after ?el2 and FALSE otherwise"
    (bind ?ord (sequence-to-ordering ?seq))
    (return (check-succession-in-ordering  ?el1 ?list ?ord)))
```

Code 64: Operators to check precedency in the PeP

The seven types of hard constraints for tradespace size are discussed below.

- $BEFORE(e_i, \{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $e_i$does not precede all the elements in $\{e_j\}$. It is useful to define pre-conditions between tasks or events.

```
(deftemplate TEST::BEFORE-CONSTRAINT (slot element) (multislot before))

(defrule TEST::enforce-BEFORE-constraints
    "If an architecture does not does not satisfy a before constraint
    then it should be deleted"
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (TEST::BEFORE-CONSTRAINT (element ?el) (before $?elems))
    (test (eq (check-precedence-in-sequence ?el ?elems ?seq) FALSE))
    =>
        (retract ?arch))
```

Code 65: The BEFORE constraint in the PeP

- $AFTER(e_i, \{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $e_i$does not succeed all the elements in $\{e_j\}$. This is the opposite rule of $BEFORE(e_i, \{e_j\})$. It is useful to define post-conditions between tasks or events.

```
(deftemplate TEST::AFTER-CONSTRAINT (slot element) (multislot after))
(defrule TEST::enforce-AFTER-constraints
   "If an architecture does not does not satisfy an AFTER constraint
   then it should be deleted"
   ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
   (TEST::AFTER-CONSTRAINT (element ?el) (after $?elems))
   (test (eq (check-succession-in-sequence ?el ?elems ?seq) FALSE))
   =>
   (retract ?arch))
```

Code 66: The AFTER constraint in the PeP

- $BETWEEN(e_i, \{e_j, e_k\})$ constraints: this enumeration rule eliminates all permutations in which $e_i$does not appear between e_j and e_k. Note that this constraint does NOT enforce the subsequence e_j, e_i, e_k since: a) the order between e_j and e_k can be reversed; there may be other elements inside the interval e_j, e_k. Hence for example, $O_1 = [e_j, e_i, e_m, e_k]$ and $O_1 = [e_k, e_l, e_i, e_m, e_j]$ would both satisfy the constraint $BETWEEN(e_i, \{e_j, e_k\})$. It is useful to define simultaneous pre- and post-conditions between tasks or events.

```
(deftemplate TEST::BETWEEN-CONSTRAINT (slot element) (multislot between))

(defrule TEST::enforce-BETWEEN-constraints
    "If an architecture does not does not satisfy a BETWEEN constraint
    then it should be deleted"

    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (TEST::BETWEEN-CONSTRAINT (element ?el) (between $?elems))
    (or (test (eq (check-succession-in-sequence ?el (first$ ?elems) ?seq) FALSE))
    (test (eq (check-precedence-in-sequence ?el (rest$ ?elems) ?seq) FALSE)) )
    =>
    (retract ?arch)
    )
```

Code 67: The BETWEEN constraint in the PeP

- $NOT\_BETWEEN(e_i, \{e_j, e_k\})$ constraints: this enumeration rule eliminates all permutations in which e_i appears between e_j and e_k. This is the opposite rule of $BETWEEN(e_i, \{e_j, e_k\})$.

```
(deftemplate TEST::NOT-BETWEEN-CONSTRAINT (slot element) (multislot not-between))

(defrule TEST::enforce-BETWEEN-constraints
    "If an architecture does not does not satisfy a BETWEEN constraint
    then it should be deleted"
```

```
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (TEST::BETWEEN-CONSTRAINT (element ?el) (between $?elems))
    (or (test (eq (check-succession-in-sequence ?el (first$ ?elems) ?seq) FALSE))
    (test (eq (check-precedence-in-sequence ?el (rest$ ?elems) ?seq) FALSE)) )
    =>
    (retract ?arch)
    )
```

Code 68: The NOT-BETWEEN constraint in the PeP

- $CONTIGUOUS(\{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $\{e_j\}$ are not contiguous. Note that there are several possible orderings that satisfy contiguity of a subset of elements.

```
(deffunction min$ (?list)
    (return (eval (str-cat "(min " (implode$ ?list) ")" )))
    )

(deffunction sort$ (?list)
    (if (eq (length$ ?list) 1) then (return (nth$ 1 ?list)))
    (bind ?mn (min$ ?list))
      (bind ?index (member$ ?mn ?list))
    (bind ?shorter-list (replace$ ?list ?index ?index (create$ )))
    (return (create$ ?mn (sort$ ?shorter-list)))
    )

(deffunction check-contiguity-in-sequence (?elems ?seq)
    (if (eq (length$ ?elems) 1) then (return TRUE))
    (bind ?ord (sequence-to-ordering ?seq))
    ;(printout t ?ord crlf)
    (bind ?indexes (create$ ))
    (for (bind ?i 1) (<= ?i (length$ ?elems)) (++ ?i)
        (bind ?indexes (insert$ ?indexes ?i (nth$ (nth$ ?i ?elems) ?ord)))
        )
    (bind ?indexes (sort$ ?indexes))
    (printout t ?indexes crlf)
    (for (bind ?i 1) (< ?i (length$ ?elems)) (++ ?i)
        (if (neq (- (nth$ (+ ?i 1) ?indexes) (nth$ (+ ?i) ?indexes)) 1) then (return
FALSE))) (return TRUE)
    )

(defrule TEST::enforce-CONTIGUITY-constraints
    "If an architecture does not does not satisfy a CONTIGUITY constraint
    then it should be deleted"

    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (TEST::CONTIGUITY-CONSTRAINT (elements $?elems) )
    (test (eq (check-contiguity-in-sequence ?elems ?seq) FALSE))
    =>
    (retract ?arch))
```

Code 69: The CONTIGUITY constraint in the PeP

- $NON\_CONTIGUOUS(\{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $\{e_j\}$ are contiguous. This is the opposite rule of $CONTIGUOUS(\{e_j\})$.

```
(defrule TEST::enforce-NON-CONTIGUITY-constraints
    "If an architecture does not does not satisfy a NON CONTIGUITY constraint
    then it should be deleted"
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (TEST::NON-CONTIGUITY-CONSTRAINT (elements $?elems) )
    (test (eq (check-contiguity-in-sequence ?elems ?seq) TRUE))
    =>
    (retract ?arch))
```

Code 70: NON-CONTIGUITY constraint in the PeP

- $SUBSEQUENCE(O_j)$ constraints: this enumeration rule eliminates all permutations which do not contain O_j as a subsequence, i.e. all permutations O_i such that $O_j \nsubseteq O_i$.

```
(deffunction is-subsequence (?subseq ?seq)
      (return (numberp (str-index (implode$ ?subseq) (implode$ ?seq))))
    )

(defrule TEST::enforce-SUBSEQUENCE-constraints
    "If an architecture does not does not satisfy a SUBSEQUENCE constraint
    then it should be deleted"

    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (TEST::SUBSEQUENCE-CONSTRAINT (subsequence $?elems) )
    (test (eq (is-subsequence ?elems ?seq) FALSE))
    =>
    (retract ?arch)
    )
```

Code 71: SUBSEQUENCE constraint in the PeP

9.1.4.2 Search heuristics

Tradespace size reduction rules: First, we note that, as mentioned earlier, it is possible to use the seven enumeration rules as search hard constraints. In addition to these seven rules, we propose one rule to fix the position of a certain element FIX-POSITION:

- $FIX\_POSITION(e_i, j)$ constraints: this tradespace size reduction rule eliminates all permutations in which element *i* does not appear in position *j*.

```
(deffunction is-in-position (?elem ?pos ?seq)
    (return (eq (nth$ ?elem (sequence-to-ordering ?seq)) ?pos))
    )

(defrule TEST::enforce-fix-position-constraint
    (TEST::FIX-POSITION-CONSTRAINT (element ?elem) (position ?pos))
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (test (eq (is-in-position ?elem ?pos ?seq) FALSE))
```

```
    =>
    (retract ?arch)
    )
```

Code 72: FIX-POSITION constraint in the PeP

Instead of deterministic rules, a few "fuzzy" constraints can also be applicable to some instances of the PeP. These fuzzy constraints are softer in nature because instead of enforcing a certain position for an element, they enforce a range of positions.

- $BY\_BEGINNING(\{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $\{e_j\}$ do not appear in "the beginning" of the sequence. Note that there are several possible orderings that satisfy contiguity of a subset of elements.

```
(deffunction by-beginning-binary (?elem ?seq)
    (if (<= (nth$ ?elem (sequence-to-ordering ?seq)) (round (/ (length$ ?seq) 3)))
        then (return TRUE) else (return FALSE))
    )

(deffunction by-beginning (?elems ?seq)
    (if (not (listp ?elems)) then (return (by-beginning-binary ?elems ?seq)))
    (if (eq (length$ ?elems) 1) then (return (by-beginning-binary (nth$ 1 ?elems)
?seq))
        else
        (if (eq (by-beginning-binary (nth$ 1 ?elems) ?seq) FALSE) then (return FALSE)
            else (return (by-beginning  (rest$ ?elems) ?seq))))
    )


(defrule TEST::enforce-by-beginning-constraint
    (TEST::BY-BEGINNING-CONSTRAINT (elements $?elems))
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (test (eq (by-beginning ?elems ?seq) FALSE))
    =>
    (retract ?arch)
    )
```

Code 73: BY-BEGINNING constraint in the PeP

- $BY\_MIDDLE(\{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $\{e_j\}$ do not appear in "the beginning" of the sequence. Note that there are several possible orderings that satisfy contiguity of a subset of elements.

```
(deffunction by-middle-binary (?elem ?seq)
    (if (and
            (> (nth$ ?elem (sequence-to-ordering ?seq)) (round (/ (length$ ?seq) 3)))
            (<= (nth$ ?elem (sequence-to-ordering ?seq)) (round (* 2 (/ (length$
?seq) 3))))
            )
        then (return TRUE) else (return FALSE)))
```

```
(deffunction by-middle (?elems ?seq)
    (if (not (listp ?elems)) then (return (by-middle-binary ?elems ?seq)))
    (if (eq (length$ ?elems) 1) then (return (by-middle-binary (nth$ 1 ?elems) ?seq))
        else
        (if (eq (by-middle-binary (nth$ 1 ?elems) ?seq) FALSE) then (return FALSE)
            else (return (by-middle  (rest$ ?elems) ?seq))))
    )

(defrule TEST::enforce-by-middle-constraint
    (TEST::BY-MIDDLE-CONSTRAINT (elements $?elems))
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (test (eq (by-middle ?elems ?seq) FALSE))
    =>
    (retract ?arch))
```

Code 74: BY-MIDDLE constraint in the PeP

- $BY\_END(\{e_j\})$ constraints: this enumeration rule eliminates all permutations in which $\{e_j\}$ do not appear in "the beginning" of the sequence. Note that there are several possible orderings that satisfy contiguity of a subset of elements.

```
(deffunction by-end-binary (?elem ?seq)
    (if (> (nth$ ?elem (sequence-to-ordering ?seq)) (round (* 2 (/ (length$ ?seq)
3))))
        then (return TRUE) else (return FALSE))
    )

(deffunction by-end (?elems ?seq)
    (if (not (listp ?elems)) then (return (by-end-binary ?elems ?seq)))
    (if (eq (length$ ?elems) 1) then (return (by-end-binary (nth$ 1 ?elems) ?seq))
        else
        (if (eq (by-end-binary (nth$ 1 ?elems) ?seq) FALSE) then (return FALSE)
            else (return (by-end  (rest$ ?elems) ?seq))))
    )

(defrule TEST::enforce-by-end-constraint
    (TEST::BY-END-CONSTRAINT (elements $?elems))
    ?arch <- (TEST::PERMUTING-ARCH (sequence $?seq))
    (test (eq (by-end ?elems ?seq) FALSE))
    =>
    (retract ?arch)
    )
```

Local search rules: Amongst the non-informed local search rules, we define a mutation operator that swaps the positions of two elements. Its CLIPS code is given below.

```
(deffunction rand-int-1-to-N (?N)
    (bind ?x (+ 1 (round (* (- ?N 1) (/ (random) 65536)))))
    )

(deffunction mutate-two-positions-in-sequence (?seq)
```

```
    (bind ?N (length$ ?seq))
    (bind ?pos1 (rand-int-1-to-N ?N))
    (bind ?ok FALSE)
    (while (eq ?ok FALSE) (bind ?pos2 (rand-int-1-to-N ?N)) (bind ?ok (neq ?pos1
?pos2)))

    (bind ?tmp (nth$ ?pos1 ?seq))
    (bind ?seq (replace$ ?seq ?pos1 ?pos1 (nth$ ?pos2 ?seq)))
    (bind ?seq (replace$ ?seq ?pos2 ?pos2 ?tmp))
    (return ?seq)
    )
(defrule SEARCH-HEURISTICS::mutation-swap-two-positions
    "This mutation function swaps the value of a single bit"
    ?arch <- (HARD-CONSTRAINTS::PERMUTING-ARCH (sequence ?orig) (mutate yes))
    =>
    (bind ?N 5)
    (for (bind ?i 0) (< ?i ?N) (++ ?i)
          (bind ?new-seq (mutate-two-positions-in-sequence ?orig))
          (bind ?new-str (explode$ (matlabf SCHED_arch_to_str ?new-seq)))
      (duplicate ?arch (sequence ?new-seq) (mutate no) (str ?new-str))
       (modify ?arch (mutate no))))
```

Code 75: A mutation operator (non-informed local search rule) for the PeP

Concerning informed local search rules, we define a crossover operator, given in Code 76.

```
(deffunction PeP-crossover (?dad ?mom); e.g. dad = (1 4 5 2 3), mom = (2 4 5 1 3)
    (bind ?n (length$ ?dad))
    (bind ?m (round (/ ?n 2)))
    (bind ?child (subseq$ ?dad 1 ?m)); first half of sequence from dad: (1 4 5)
    (for (bind ?i 1) (<= ?i ?n) (++ ?i); respect order from mom's second half: (2 3)
    (bind ?el (nth$ ?i ?mom))
        (if (not (subsetp (create$ ?el) ?child)) then (bind ?child (insert$ ?child (+
(length$ ?child) 1) ?el))))
    (return ?child)
    )

(defrule TEST::PeP-crossover-operator
    "This rule performs a crossover between two individuals of a PeP population"

     ?dad <- (TEST::PERMUTING-ARCH (sequence $?seq-dad) (improve yes))
     ?mom <- (TEST::PERMUTING-ARCH (sequence $?seq-mom) (improve yes))
    =>
    (bind ?new-seq (PeP-crossover ?seq-dad ?seq-mom));
    (assert (TEST::PERMUTING-ARCH (sequence ?new-seq) (improve no)))
    (modify ?dad (improve no))
    (modify ?mom (improve no))
    )
```

Code 76: A crossover operator (informed local search rule) for the PeP

9.2 Domain-specific, EOSS-independent knowledge

9.2.1 Orbit selection rules

The orbit selection strategy works as follows. The RBES is initialized, and all possible instruments and orbits are asserted, i.e. added into working memory. A mission fact is asserted carrying a certain number of instruments, without any orbit assigned. A rule is added that from this initial fact, will assert all possible orbit assignments to all allowed orbits, i.e. all possible combinations of altitude, inclination, and LTAN (circular orbits are assumed). This rule is shown below.

```
(defrule ORBIT-SELECTION::assert-all-possible-mission-orbits
    (declare (salience 20))
    ?orig <- (MANIFEST::Mission (Name ?name) (select-orbit yes) (in-orbit ?nil)
(orbit-altitude# nil) (orbit-type nil) (orbit-RAAN nil) (orbit-inclination nil)
(instruments $?payload))
    =>
    (foreach ?typ (create$ SSO LEO)
        (foreach ?h (create$ 275 400 600 800 1300)
            (foreach ?i (create$ polar SSO near-polar)
                (foreach ?raan (create$ AM PM DD NA)
                    (if (valid-orbit ?typ ?h ?i ?raan) then (duplicate ?orig (in-
orbit (str-cat ?typ "-" ?h "-" ?i "-" ?raan)) (orbit-altitude# ?h) (orbit-type ?typ)
(orbit-RAAN ?raan) (orbit-inclination ?i) )))
        )))
    (retract ?orig)
    )
```

Code 77: Rule to assert all possible orbits for a mission.

The main orbit selection rules are described below, classified according to the orbital parameter they concern. Only orbit altitude, inclination, and right ascension of the ascending node are considered, since most LEO have circular orbits and therefore eccentricity and argument of the perigee are trivial.

Orbit altitude: Different instruments perform better when they fly at different altitudes. High power active instruments such as lasers or SAR are preferably flown at low altitudes to minimize power requirements. This is because the radar equation has a term in R^{-4} which means that the backscattered power received at the antenna decreases with the fourth power of the distance. This is the case of ESA's future explorer ADM-Aeolus which carries a Doppler wind LIDAR and will fly at 400km. A rule capturing this is shown in the Appendix, on Code 78. On the other hand, passive optical imagers usually prefer to be in higher orbits that provide better swath, coverage and revisit time. This is the case of the SPOT series for instance, which flew at 820km. This other piece of knowledge is captured in Code 79. Hence, when a lidar or a SAR are combined with an optical imager, whichever the altitude is the solution will be suboptimal from the point of view of at least one instrument.

This is the case of Envisat for example, in which the SAR (ASAR) and the radar altimeter (RA-2) had to fly at 800km thus increasing their individual power requirements. Had the SAR flown at 400km for instance, it would have needed 12dB less of power (a factor of 16 less). On the other hand in EE/EarthCare which flies at 400km there is a MSI with a swath of 150km and a spatial resolution of 500m. The same instrument flying at 800km would have twice the swath and hence twice better coverage although spatial resolution would also double from 500m to 1000m. Note also that if there is a requirement for repeatability of ground traces or very stringent revisit time requirements, only a few altitudes will be able to satisfy all these constraints.

```
(defrule ORBIT-SELECTION::no-lidars-beyond-500km-p20 "Because of power
considerations"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name) (penalty-var ?var)  (is-type ?typ) (h ?h&:(> ?h 500)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int))
    (or
        (test (eq ?int "Laser altimeters"))
        (test (eq ?int "Elastic lidar"))
        (test (eq ?int "Differential Absorption Lidars"))
        (test (eq ?int "Doppler Wind Lidars"))
        )
    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 20) )")))

(defrule ORBIT-SELECTION::no-radars-beyond-600km-except-oceanography-p7
"Because of power considerations"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type ?typ) (h ?h&:(> ?h 600)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int))
    (or
        (test (eq ?int "Imaging MW radars (SAR)"))
        (test (eq ?int "Cloud profile and rain radars"))
        )
    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 7) )")))
```

Code 78: Orbit selection rules that favor low orbits for spacecraft carrying high energy instruments

```
(defrule ORBIT-SELECTION::passive-imagers-want-to-fly-high-b6 "Because of
coverage considerations"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type ?typ) (h ?h&:(> ?h 600)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int))
    (or
        (test (eq ?int "Imaging multi-spectral radiometers -passive optical-
"))
        (test (eq ?int "Imaging multi-spectral radiometers -passive MW-"))
        (test (eq ?int "High resolution optical imagers"))
        )
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 6) )")))

(defrule ORBIT-SELECTION::passive-imagers-want-to-fly-high-b3 "Because of
```

```
coverage considerations"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type ?typ) (h ?h&:(> ?h 400)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int))
    (or
        (test (eq ?int "Imaging multi-spectral radiometers -passive optical-
"))
        (test (eq ?int "Imaging multi-spectral radiometers -passive MW-"))
        (test (eq ?int "High resolution optical imagers"))
        )
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 3) )")))
```

Code 79: Orbit selection rules that favor high orbits for spacecraft carrying passive imagers

Oceanography instruments have historically been flown in higher orbits around 1000km in order to maximize coverage. This is captured through the following rule in Code 80.

```
(defrule ORBIT-SELECTION::oceanography-missions-want-1000km-b20 "To get low
drag penalties in better orbit determination"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type LEO) (h 1300) (i near-polar))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Illumination ?illum)
(average-power# ?p) (Concept ?c))
    (test (neq (str-index "ocean" ?c) FALSE))

    (or
      (test (eq ?int "Imaging MW radars (SAR)"))
       (test (eq ?int "Radar scatterometer"))
       (test (eq ?int "Radar altimeter"))
       )
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 20) )"))
    )
```

Code 80: Orbit selection rules that favors higher 1000km orbits for oceanography missions

Finally, geodynamic sensors usually fly at low or very low altitudes to achieve better accuracies in the measurement of the Earth's gravity or magnetic field. A modern example is the case of ESA Earth Explorers GOCE (263km) and Swarm (400km). Such requirement is captured in the following rule in Code 81.

```
(defrule ORBIT-SELECTION::no-grav-or-magn-instruments-beyond-400km-p9 "Because
need the Earth to be close"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type ?typ) (h ?h&:(> ?h 600)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int))
    (or
        (test (eq ?int "Magnetic field"))
        (test (eq ?int "Gravity instruments"))
        )
    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 9) )")))
```

```
(defrule ORBIT-SELECTION::nothing-below-400-except-grav-magn-p9 "Because of
drag issues"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type ?typ) (h ?h&:(< ?h 400)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int))

       (test (neq ?int "Magnetic field"))
    (test (neq ?int "Gravity instruments"))

    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 9) )")) )
```

Code 81: Orbit selection rules that favor very low orbits for spacecraft carrying geodynamic sensors and penalize those orbits for all other sensors

- Inclination: The choice of orbit inclination is typically set by two factors: 1) whether the users would like to sample diurnal cycles of the phenomena of interest or they would rather avoid diurnal cycles; 2) whether the user requires global coverage or they would rather focus on lower inclinations. Most remote sensing instruments require sun-synchronous orbits (SSO) in order to get rid of diurnal effects. A notorious exception is oceanography. Oceanography missions are typically flown in non SSO in order to capture diurnal sampling and thus avoid tidal aliasing. This is captured by the rule shown in Code 82.

```
(defrule ORBIT-SELECTION::no-SSO-for-oceanography-instruments-p4 "Because SSO
bring tidal aliasing problems"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type SSO))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) )
    (or
        (test (eq ?int "Radar scatterometer"))
        (test (eq ?int "Radar altimeter"))
        )
    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 4) )"))
    )
(defrule ORBIT-SELECTION::passive-imagers-want-SSO-not-POL-p3 "To ensure
illumination characteristics that are as similar as possible every pass"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type ?type&:(neq ?type SSO)))
    (DATABASE::Instrument (Name ?ins) (Concept ?c))
    (test (eq (str-index "cryosph" ?c) FALSE))
    (test (eq (str-index "oceanograph" ?c) FALSE))
    (test (eq (str-index "altimet" ?c) FALSE))
    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 3) )"))
    )
```

Code 82: Orbit selection rules that favor SSO for generic instruments and penalize SSO for oceanography missions

Finally, cryospheric missions prefer true polar inclinations in order to maximize coverage of the polar regions. This is captured by the following rule given in Code 83.

```
(defrule ORBIT-SELECTION::true-polar-preferred-for-cryospheric-instruments-b4
"Because better coverage of the poles"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-mission
?name)(penalty-var ?var)  (is-type LEO) (i ?i&:(eq i polar)))
    (DATABASE::Instrument (Name ?ins) (Concept ?c) )
    (or
        (test (neq (str-index "cryospher" ?c) FALSE))
        (test (neq (str-index "ice" ?c) FALSE))
        (test (neq (str-index "snow" ?c) FALSE))
        )
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 4) )"))    )
```

Code 83: Orbit selection rule that favors true polar orbits for cryospheric missions

- Right ascension or local time of the ascending node for Sun-synchronous orbits: in the case of SSO, the choice of the local time of the ascending node is extremely important because the local time at which measurements are taken is a constant for a given latitude and almost the same for all non-polar latitudes as shown in Figure 94.

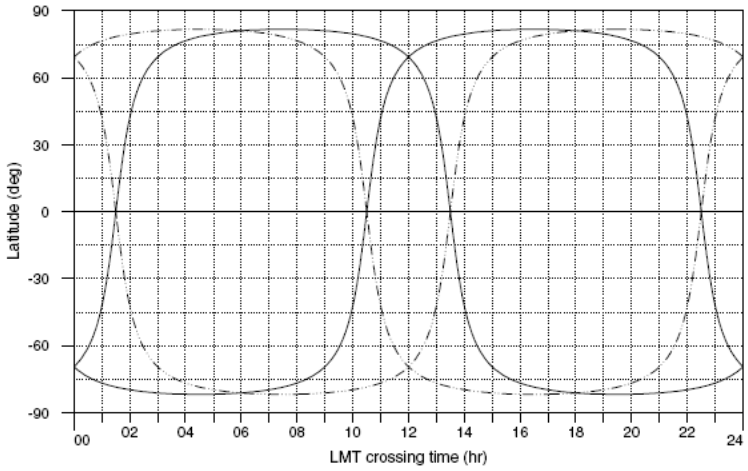


Figure 94: LTM crossing time vs latitude for 800km orbits with RAAN at 10:30, 13:30, 22:30,01:30[24]

In practice the choice of RAAN is chosen as a compromise between various soft constraints:

- To obtain the best solar lighting conditions for the regions observed; this typically implies the selection of an AM orbit, as captured by the rules on Code 84.

```
(defrule ORBIT-SELECTION::most-instruments-want-raan-AM-b2 "Better lighting
conditions in the morning than in the afternoon"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-
mission ?name) (penalty-var ?var)  (is-type SSO) (raan AM))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Concept ?c)
(Illumination Passive))
    =>
```

[24] Image credit: Miguel Aguirre (ESA/ESTEC)

```
    (eval (str-cat  "(bind " ?var " (- " ?var " 2) )"))

(defrule ORBIT-SELECTION::passive-VNIR-instruments-want-raan-AMorPM-p7
"Because passive VIS/NIR instruments need a source of light"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-
mission ?name)(penalty-var ?var)  (is-type SSO) (raan ?raan&:(eq ?raan
DD)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Spectral-region ?sr)
(Illumination Passive))
    (or
        (test (eq ?sr UV)) (test (eq ?sr VIS))
        (test (eq ?sr NIR))(test (eq ?sr VNIR))
        (test (eq ?sr UV+VNIR))
        )
    =>
    (eval (str-cat  "(bind " ?var " (+ " ?var " 7) )")))
```

Code 84: Orbit selection rules favoring the selection of AM orbits to optimize lighting conditions

- To maximize sensitivity to a certain phenomenon that has a diurnal cycle. For example, photosynthetic activity is known to be maximal in the early afternoon. Thus, it is preferable to fly vegetation instruments on PM orbits. This is taken into account by the following rule given in Code 85.

```
(defrule ORBIT-SELECTION::vegetation-instruments-want-raan-PM-b3 "Because
photosynthetic activity is maximal early PM"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-
mission ?name)(penalty-var ?var)  (is-type SSO) (raan ?raan&:(eq ?raan
PM)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Concept ?c)
(Illumination Passive))
 (or
        (test (neq (str-index "vegetation" ?c) FALSE))
        (test (neq (str-index "photosynthe" ?c) FALSE))
        )

    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 3) )"))
    )
```

Code 85: Orbit selection rule that favors PM orbits for spacecraft carrying vegetation instruments

- To reduce the risk of antisolar or specular reflection that could blind the sensor: this occurs when the position of the Earth, the satellite and the Sun are so that the sunlight reflected by the Earth, which can be much more intense that the radiation coming from the observed region goes directly into the sensor. This typically precludes the utilization of SSO with RAAN around noon.
- To take regional meteorological factors into account: typically, avoid clouds that form in the tropics around mid-afternoon, as captured by the rule provided in Code 86.

```
(defrule ORBIT-SELECTION::other-instruments-want-raan-AM-b2 "Because there
is less cloudiness in the morning than in the afternoon"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-
mission ?name) (penalty-var ?var)  (is-type SSO) (raan AM))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Concept ?c)
(Illumination Passive))
    (test (eq (str-index "vegetation" ?c) FALSE))
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 2) )"))
    )
```

Code 86: Orbit selection rule that favors AM orbits for spacecraft carrying generic, passive instruments to avoid cloudiness

Similarly, pollution is maximal early afternoon, as shown in Code 87.

```
(defrule ORBIT-SELECTION::chemistry-instruments-want-raan-PM-b3 "Because
pollution is maximal in early PM"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-
mission ?name)(penalty-var ?var)  (is-type SSO) (raan ?raan&:(eq ?raan
PM)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Concept ?c)
(Illumination Passive))
 (or
        (test (neq (str-index "chemistry" ?c) FALSE))
        (test (neq (str-index "pollut" ?c) FALSE))
        )
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 3) )"))    )
```

Code 87: Orbit selection rule that favors PM orbits for spacecraft carrying tropospheric chemistry instruments

- To take into account the crossing time of another SSO satellite or sets of satellites in order to make data co-registration easier. For example, the local time of the post EPS satellites will be 09:30 to maximize the synergy with the MetOp mission. This is currently the main constraint when the instrument belongs to a series, such as the A-train.
- To limit periods of solar eclipse: Dawn-dusk orbits, with a local time around 6:00 or 18:00, have very few eclipses whereas AM or PM orbits have almost constant duration eclipses of 35 min (approximately a third of the orbital period). Figure 95 illustrates these effects.

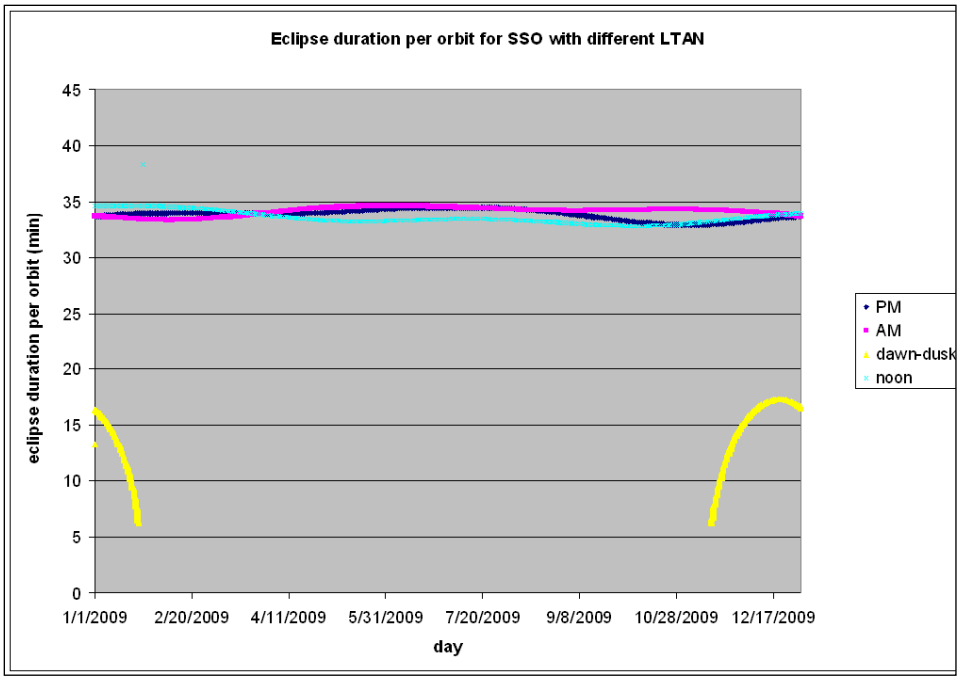


Figure 95: Eclipse duration time per orbit for SSO with different LTAN[25]

This is taken into account by the following rule, provided in Code 88:

```
(defrule ORBIT-SELECTION::high-energy-instruments-want-raan-DD-b8 "Because
DD orbits are more favorable in terms of energy"
    ?o <-(ORBIT-SELECTION::orbit (orb ?miss) (of-instrument ?ins) (in-
mission ?name)(penalty-var ?var)  (is-type SSO) (raan ?raan&:(eq ?raan
DD)))
    (DATABASE::Instrument (Name ?ins) (Intent ?int) (Illumination ?illum)
(average-power# ?p))
    (or
        (test (eq ?illum Active))
             (test (eq ?int "Imaging MW radars (SAR)"))
        (test (eq ?int "Cloud profile and rain radars"))
        (test (eq ?int "Laser altimeters"))
        (test (eq ?int "Elastic lidar"))
        (test (eq ?int "Differential Absorption Lidars"))
        (test (eq ?int "Doppler Wind Lidars"))
        (test (and (neq ?p nil) (> ?p 1000)))
        )
    =>
    (eval (str-cat  "(bind " ?var " (- " ?var " 8) )"))
    )
```

Code 88: Orbit selection rule that favors DD orbits for spacecraft carrying high energy instruments

[25] The data for this plot was calculated using STK with the following satellites: Terra (AM), Aqua (PM), QuikSCAT (dawn-dusk) and Orbview (noon)

- To limit thermal variations during each revolution: In SSO, the angle between the direction of the sunlight and the direction perpendicular to the solar panels varies describing a cone. As we see in Figure 96, this cone is narrower for PM orbits than for AM orbits, which simplifies the design of the power subsystem. This reinforces the rule presented in Code 88.

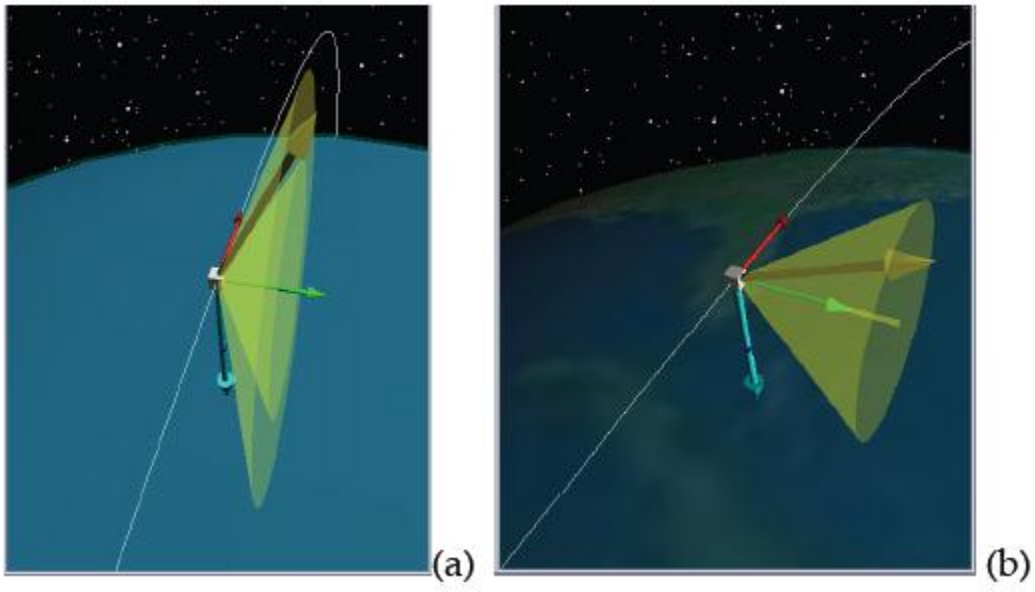


Figure 96: Solar cone for two satellites with different local times: a) AM b) dawn-dusk[26]

Given these constraints, each kind of instrument has a "preferred" RAAN depending on the application. As an example, Table 51 provides the RAAN of several Earth observation satellites:

[26] Image credit: Miguel Aguirre (ESA/ESTEC)

Table 51: Local Time of Ascending Node for various Earth observation satellites[27]

Sun-synchronous satellite	τ_{AN}	Sun-synchronous satellite	τ_{AN}
Landsat-1	21:30	IRS-1A	22:25
Landsat-2	21:30	IRS-1B	22:25
Landsat-3	21:30	IRS-1C	22:30
Landsat-4	21:45	IRS-1D	22:30
Landsat-5	21:45	IRS-P2	22:40
Landsat-7	22:00	IRS-P3	22:30
EO-1	22:01	IRS-P6	22:30
SAC-C	22:15	TES	22:30
SPOT-1	22:30	Cartosat-1	22:30
SPOT-2	22:30	Cartosat-2	22:30
SPOT-3	22:15	CBERS-1	22:30
SPOT-4	22:30	CBERS-2	22:30
SPOT-5	22:30	TMSat	22:20
Hélios-1A	13:17	OrbView-3	22:30
Hélios-1B	13:16	OrbView-4	22:30
Pléiades-1	22:15	QuickBird-2	22:20
Pléiades-2	22:15	Ikonos-2	22:30
ERS-1	22:15	EarlyBird-1	22:30
ERS-2	22:30	QuikTOMS	22:30
Envisat	22:00	BIRD	22:30
EarthCARE	22:30	Terra (EOS-AM-1)	22:30
MOS-1	22:25	Aqua (EOS-PM-1)	13:30
MOS-1B	22:30	CloudSat	13:31
JERS-1	22:30	Calipso	13:31
ADEOS-1	22:30	PARASOL	13:33
ADEOS-2	22:30	Aura (EOS-Chem-1)	13:38
ALOS	22:30	OCO	13:15
EROS-A1	21:45	Rocsat-2	21:45
Kompsat-1	22:50	Tan Suo-1	23:00
Resource21-01	22:30	Diamant-1	23:30
Resource21-02	22:30	Diamant-2	23:30
Resurs-O1-4	22:15	RapidEye-1	12:00
TechSat-1B	22:15	NEMO	10:30
FaSat-2	22:20	SSR-1/ss	09:30

Four groups of satellites can be identified:

- Those flying in AM orbits (local time between 09:30/21:30 and 10:30/22:30): the vast majority of Earth observation satellites are launched in AM orbits in order to optimize sunlight conditions. Note from Figure 94 that a RAAN around 22:30 AM yields optimal lighting conditions for latitudes in the Northern hemisphere, whereas a RAAN around 10:30 yields optimal conditions for observation of the Southern hemisphere. Therefore most satellites are launched with RAAN = 22:30 and observe during the descending part of their orbits.

[27] This table is taken from (Capderou, 2005).

- Those flying in PM orbits (local time around 01:30/13:30): some optical and microwave passive instruments are launched in PM orbits that are symmetrical to the AM orbits around noon. PM orbits may allow avoiding systematic cloud cover in certain regions of the Earth during the afternoon. Furthermore, some instruments are put in two satellites, one with an AM orbit and the other with a PM orbit so that revisit time is halved assuming perfect data registration. This is the case of CERES and MODIS which are both flown both in EOS/Terra (AM) and EOS/Aqua (PM). Note from Figure 94 that a RAAN around 13:30 AM yields optimal lighting conditions for latitudes in the Northern hemisphere, whereas a RAAN around 01:30 yields optimal conditions for observation of the Southern hemisphere. An interesting example of satellites in PM orbits is the A-train, which consists of five satellites with RAAN spaced by only a few minutes: EOS/Aqua (13:30), CloudSat (13:31), Calipso (13:31), PARASOL (13:33) and EOS/Aura (13:38).
- Those flying in dawn-dusk orbits: we find in this group high energy instruments such as LIDARs, SARs and scatterometers which require short eclipse durations to avoid large batteries. For example, Radarsat-1, a Canadian satellite launched in 1995 which carried a SAR, flies in a SSO orbit with an ascending node at 18:00. Other missions flying in this kind of orbit for similar reasons are QuikSCAT (NASA scatterometer, 1999), TerraSAR-X1 and –L1 or the aforementioned GOCE and ADM. Note that dawn-dusk orbits also minimize the effects of the ionosphere.
- Those flying in noon-midnight orbits which are rare because of the aforementioned risk of sunlight reflections. However, these orbits are perfectly suitable when the intention is to measure the sunlight reflected by the observable. This is typically the case of ocean color sensors like SeaWiFS in SeaStar (renamed Orbivew-2) which flew at 12:20 descending node, or OCS in Haiyan-1 (a.k.a. Ocean-1) which flies around 12:00.

9.2.2 Power budget rules

The rules that compute a simplified power budget are taken from Chapter 11 in Space Mission Analysis and Design (Larson & Wertz, 1999a).

```
(deffunction get-dod (?orbit)
    (bind ?type (matlabf get_orbit_type ?orbit))
    (bind ?raan (matlabf get_orbit_raan ?orbit))
    (if (eq ?type GEO) then (bind ?dod 0.8)
        elif (and (eq ?type SSO) (eq ?raan DD)) then (bind ?dod 0.6)
        else (bind ?dod 0.4)
        )
    (return ?dod))
```

```
(defrule EPS-DESIGN::design-EPS
    ?miss<- (MANIFEST::Mission (payload-power# ?p&~nil) (EPS-mass# nil) (in-orbit
?orb)
        (payload-mass# ?m) (satellite-BOL-power# nil)(lifetime ?life))
    =>
    (bind ?orbit-info (get-orbit-info ?orb))
    (bind ?frac (nth$ 1 ?orbit-info))
    (bind ?angle (nth$ 2 ?orbit-info))
    (bind ?T (nth$ 3 ?orbit-info))
    (bind ?dod (get-dod ?orb))
    (bind ?epsm (matlabf mass_EPS ?p ?p ?frac ?angle ?T ?life ?m ?dod))
    (bind ?pow (matlabf power_EPS ?p ?frac ?angle ?T ?life))
    (modify ?miss (EPS-mass# ?epsm) (satellite-BOL-power# ?pow))
    )
```

Code 89: Rule computing a simple satellite power budget

9.2.3 Complexity-corrected mass budget rules

The mass budgets are calculated in a 3-step process: first, penalties are calculated from the characteristics of the instruments in the spacecraft; second, the mass of each subsystem is calculated using information from these penalties to decide which subsystem-to-payload mass ratios are appropriate; third, complexity penalties are applied in the appropriate cases. Rules are used in each of these three steps. These rules are described in the following paragraphs.

Mechanical interactions: Instruments affect each other from the mechanical point of view in many ways:

- Moving parts of spacecraft such as scanning instruments create vibrations that can excite the natural frequencies of the structure. This needs to be taken into account in the dynamic behavior study of the satellite in order to avoid harm to the platform or to the instruments. For example, the Aqua's AMSR-E is a 300kg scanning parabolic antenna; one can imagine that such a massive device with a diameter of 1.6m in continuous rotation at 40rpm induces heavy perturbations on its co-passengers. This can be modeled as a mass complexity penalty, as shown in the rule in Code 90. Note that a single scanning instrument activates this penalty for the whole spacecraft in which it is flown, regardless of the other instruments.

```
(defrule MASS-BUDGET::scanning-penalty-check
    "This rule finds out whether there is any instrument in the mission
    with a scanning requirement and updates scanning-penalty boolean value"
    (declare (salience 20))
    ?miss <- (MANIFEST::Mission (scanning-penalty nil) (instruments $?payload))
    =>
    (bind ?penalty 0)
    (foreach ?instr ?payload
        (bind ?scan (get-scanning ?instr))
        (if (and (neq ?scan "no-scanning") (neq ?scan nil) (neq ?penalty 1)) then
             (modify ?miss (scanning-penalty 1))
```

```
                (bind ?penalty 1)
            )
        )
    (if (eq ?penalty 0) then (modify ?miss (scanning-penalty 0))
        elif (eq ?penalty 1) then (modify ?miss (scanning-penalty 1)))
    )
```

Code 90: Rule computing the scanning complexity penalty in satellite mass budgets

- Mechanisms: mechanisms are very often used in satellites to deploy very large solar arrays or antennae. This way some of the problems pointed out in the section concerning limited resources on the spacecraft can be overcome. An example of a relatively simple mechanism is NASA's Aquarius mission with its complex instrument featuring one scatterometers and 3 microwave radiometers. An animation of the deployment of the instrument in orbit can be found in the NASA's Aquarius website[28]. The deployment consists of four independent movement or steps and each of the steps needs to be successful in order for the satellite to work correctly.

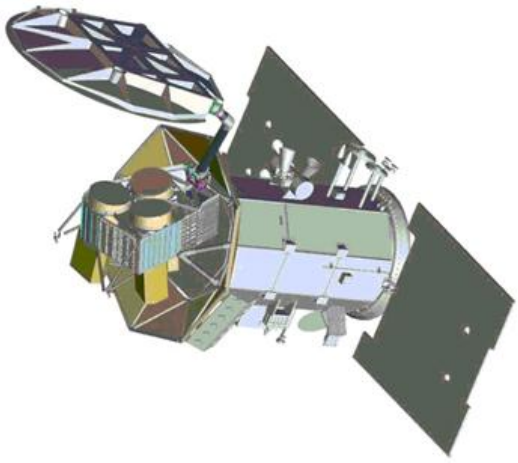


Figure 97: NASA's Aquarius spacecraft featuring one scatterometer and 3 radiometers (Image credit: NASA)

Probably one of the most complex mechanisms never to be deployed is NASA's James Webb Space Telescope. An astonishing animation of the deployment of the James Webb Space telescope can be found in the project's website[29]. It is intuitive to understand that such complex mechanisms involving a large number of steps can substantially lower the reliability of the mission. As a numerical example let us assume that the probability of each individual step being successful is 95%. Then the reliability of the mechanism is 95% ^30 steps = 21%. While the hope is that the value of 95% is highly pessimistic, this calculation illustrates the danger of mechanisms.

[28] http://www.esr.org/aquarius_sat/AQU_Animation_050624_0811.mpg

[29] http://jwst.gsfc.nasa.gov/videos/09jwsta_depedit_720p_4mbps

A rule computing a complexity penalty related to the presence of mechanisms is shown below.

```
(defrule MASS-BUDGET::mechanisms-penalty-check
    "This rule finds out whether there is any instrument in the mission
    with a deployment mechanism and updates mechanism-penalty boolean value"
    (declare (salience 20))
    ?miss <- (MANIFEST::Mission (mechanisms-penalty nil) (instruments
$?payload))
    =>
    (bind ?penalty 0)
    (foreach ?instr ?payload
        (bind ?dep (get-deployment-mechanism ?instr))
        (if (and (eq ?dep "yes") (neq ?penalty 1)) then
            (modify ?miss (mechanisms-penalty 1))
               (bind ?penalty 1)
            )
        )
    (if (eq ?penalty 0) then (modify ?miss (mechanisms-penalty 0)))
    )
```

Code 91: Rule computing the mechanism complexity penalty in satellite mass budgets

- The same moving parts perturb instruments with high pointing accuracy requirements or high integration times. An example of this is the case of IASI, an infrared sounder on the Metop platform. During the testing phase of Metop, the IA&T team noticed that the performance of the instrument was much lower than expected because the instrument was highly sensitive to the micro-vibrations induced by the other instruments on the platform. Eventually it was necessary to add dampers to the instrument to improve its performance. This was obviously not without cost. This kind of problem is modeled through the rule shown in Code 92.

```
(defrule MASS-BUDGET::ADCS-penalty-check
    "This rule finds out whether there is any instrument in the mission
    with high pointing requirements and updates ADCS-penalty boolean value"
    (declare (salience 20))
    ?miss <- (MANIFEST::Mission (ADCS-penalty nil) (instruments $?payload))
    =>
    (bind ?penalty 0)
    (foreach ?instr ?payload
        (bind ?pen (get-pointing-requirements ?instr))
        (if (and (eq ?pen "High") (neq ?penalty 1)) then
            (modify ?miss (ADCS-penalty 1))
               (bind ?penalty 1)
            )
        )
    (if (eq ?penalty 0) then (modify ?miss (ADCS-penalty 0)))
    )
```

Code 92: Rule computing the pointing complexity penalty in satellite mass budgets

Thermal interactions: In Earth observation, thermal control is usually integrated in the instrument. Some sensors require to be cooled down to extremely low temperatures in order to achieve functional SNRs. This is typically the case of short wave infrared sensors like ADEOS/GLI, MIPAS and AATSR from Envisat and ASTER, MOPITT, AIRS and HIRDLS from EOS. Temperatures of up to 180K can be achieved by thermoelectric coolers, but to achieve lower temperatures it is necessary to use mechanical coolers such as Stirling coolers, which induce vibrations on the platform. Cryocoolers are usually part of the instruments, so that the spacecraft's thermal subsystem does not have to provide that much cold. However, the design of the thermal subsystem may be more complicated in the presence of cryocoolers because extremely low temperatures can harm other spacecraft components, typically batteries, for which the lowest functional temperature is around -40 deg C. Hence, heaters and radiators may be needed in order to assure that the temperature of the batteries is maintained over -40deg C. A rule capturing this effect is provided in Code 93.

```
(defrule MASS-BUDGET::thermal-penalty-check
    "This rule finds out whether there is any instrument in the mission
    requiring active cryo-cooling and updates thermal-penalty boolean value"
    (declare (salience 20))
    ?miss <- (MANIFEST::Mission (thermal-penalty nil) (instruments $?payload))
    =>
    (bind ?penalty 0)
    (foreach ?instr ?payload
        (bind ?pen (get-thermal-control ?instr))
        (if (and (eq ?pen "Active-cryocooler") (neq ?penalty 1)) then
           (modify ?miss (thermal-penalty 1))
                (bind ?penalty 1)
            )
        )
    (if (eq ?penalty 0) then (modify ?miss (thermal-penalty 0)))
)
```

Code 93: Rule computing the thermal complexity penalty in satellite mass budgets

Even in the case of purely passive thermal control using radiators, another problem of thermal origin appears: all the instruments "fight" to get a good view of cold space which makes the configuration design process much more complicated. For example in the case of Envisat the cold face was the top of the satellite, which explains the accumulation of instruments in that zone. In addition to that, instruments are sources of heat. If two instruments are put next to each other on a platform, there will generally be an interchange of heat with between the instruments and the platform and between the instruments. This needs to be taken into account by the thermal specialist.

Electromagnetic interactions: Virtually all remote sensing sensors emit and/or receive electromagnetic radiation in a certain part of the spectrum. Consequently a number of problems appear:

- Active instruments may jam passive instruments or tracking, telemetry and command equipment working on the same frequency band, and even those that are in different bands because of harmonics and inter-modulation products.
- High RF power instruments can induce currents in nearby electronic devices. This is solved by adequate shielding of all electrical wires.

The consequence of these problems is that the architect needs to think carefully about the configuration of the satellite in order to avoid interferences between instruments. In some cases, it is enough with setting the instruments in opposite sides of the satellite for example, but when this is not possible, structures such as long booms are necessary to protect passive sensitive instruments from the electromagnetic environment in the platform. This is the case of many satellites using precise magnetometers to sense the Earth's magnetic field, such as Swarm, or the GOES 3rd generation. Sometimes booms are also used to isolate TT&C equipment from the rest of the spacecraft like in the case of Landsast-4 or EOS/Terra.

In addition, EMC issues are extremely hard to model and thus to predict. Consequently, extensive EMC testing is required to ensure that instruments will not interfere with each other during flight. Naturally, the number and cost of the tests that are necessary in the case of multi-instrument platforms is much higher than in the case of dedicated satellites.

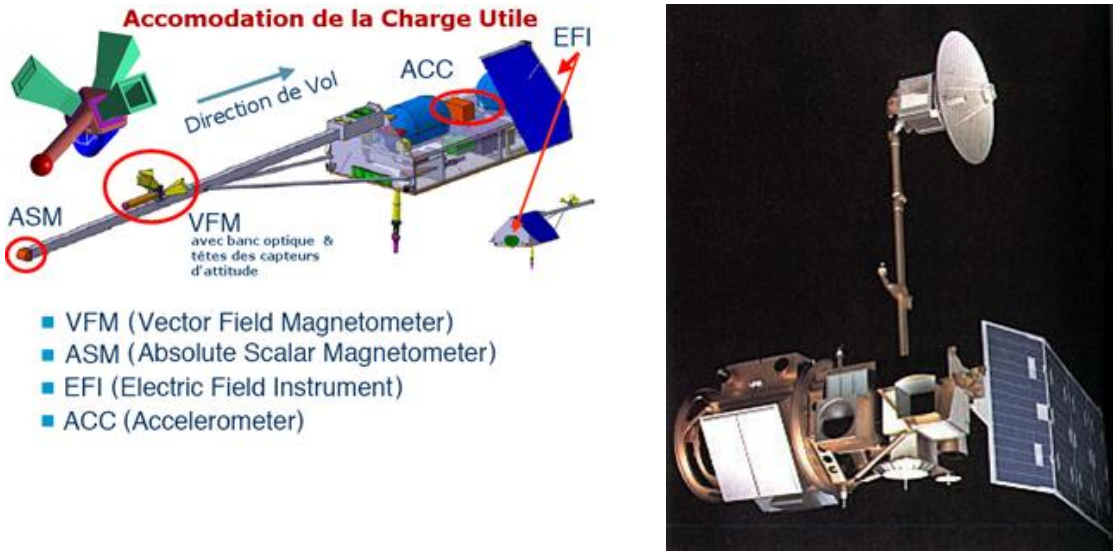


Figure 98: Examples of long booms in the EE/Swarm (left) and Landsat-4 (right) satellites

A rule capturing the EMC interaction between instruments is presented below.

```
(defrule MASS-BUDGET::EMC-penalty-check
    "This rule finds out whether there are any pair of instruments in the mission
    in which one is active, the other passive, and both use the same MW band"
    (declare (salience 20))
```

```
    ?miss <- (MANIFEST::Mission (EMC-penalty nil) (instruments $?payload))
    (test (> (length$ ?payload) 0))
    (DATABASE::Instrument (Name ?n1) (Illumination Active) (band ?band&~nil))
    (DATABASE::Instrument (Name ?n2) (Illumination Passive) (band ?band&~nil))
    (test (eq (sub-string 1 2 ?band) MW))
    (test (integerp (member$ ?n1 ?payload)))
    (test (integerp (member$ ?n2 ?payload)))

    =>
    (modify ?miss (EMC-penalty 1)))
```

Code 94: Rule computing the EMC penalty in the satellite mass budget

Optical interactions: The purpose of most instruments on Earth observation satellites is by definition to observe the Earth. This implies that they all require a view of the Earth which means that in most multi-instrument platforms most instruments will be in the nadir looking face. As a consequence, the available surface for instruments on the satellite is far from being the whole satellite surface. Furthermore, instruments have very different viewing concepts and the configuration needs to be so that none of them interfere with each other.

Finally, a last complexity penalty is applied to spacecraft for which the total data rate requirement is large. Indeed, although one can sometimes trade complexity amongst subsystems, there are limitations that prevail due to the state of technology for certain components. Data rate is a good example. State-of-the-art downlink data rates are in X-band (8-12 GHz) which provides around 400-600Mbps in LEO with a reasonable ground station using an antenna on the satellite of reasonable size and emitting a reasonably small power. In order to achieve higher data rates, complex Ka-band or even optical links are being studied. This penalty is captured in the rule shown in Code 95.

```
(defrule MASS-BUDGET::datarate-penalty-check
    "This rule finds out whether there is any instrument in the mission
    with a scanning requirement and updates scanning-penalty boolean value"
    (declare (salience 20))
    ?miss <- (MANIFEST::Mission (datarate-penalty nil) (instruments $?payload))
    =>
    (bind ?penalty 0)
    (foreach ?instr ?payload
        (bind ?rb (get-data-rate ?instr))
        (if (and (> ?rb 100) (neq ?penalty 1)) then
            (modify ?miss (datarate-penalty 1))
                (bind ?penalty 1)
            )
        )
    (if (eq ?penalty 0) then (modify ?miss (datarate-penalty 0)))
    )
```

Code 95: Data rate penalty for satellite mass budgets

Once these penalties have been computed, rules are used to decide which subsystem-to-payload mass ratios to use depending on these penalties, and mass penalties are applied depending on the penalties. Most of the penalties are applied to the structures subsystem, as shown in Code 16. This is so because it is assumed that all the work related to the spacecraft configuration is assigned to the structures team.

```
(defrule MASS-BUDGET::design-structure-subsystem
    "Computes structure subsystem mass using rules of thumb"
    (declare (salience 10))
    ?miss <- (MANIFEST::Mission (structure-mass# nil) (mechanisms-penalty ?mech-pen)
        (payload-mass# ?m) (thermal-penalty ?th-pen) (EMC-penalty ?emc-pen)
        (scanning-penalty ?sc-pen))
    =>
    (if (eq ?mech-pen nil) then (bind ?mech-pen 0))
    (if (eq ?th-pen nil) then (bind ?th-pen 0))
    (if (eq ?emc-pen nil) then (bind ?emc-pen 0))
    (if (eq ?sc-pen nil) then (bind ?sc-pen 0))
    (bind ?struct-mass (* 0.75 ?m))
    (bind ?struct-mass (+ ?struct-mass (* ?mech-pen 0.10 ?m)))
    (if (eq ?emc-pen nil) then (bind ?emc-pen 0))
    (bind ?struct-mass (+ ?struct-mass (* ?emc-pen 0.05 ?m)))
    (bind ?struct-mass (+ ?struct-mass (* ?sc-pen 0.10 ?m)))
    (bind ?struct-mass (+ ?struct-mass (* ?th-pen 0.05 ?m)))
    (modify ?miss (structure-mass# ?struct-mass))
    )
```

Code 96: Rule to predict the structure mass in complexity-corrected mass budgets

The communications subsystem mass takes into account the data rate penalty, as shown in Code 97.

```
(defrule MASS-BUDGET::design-comm-subsystem
    "Computes comm subsystem mass using rules of thumb"
    (declare (salience 10))
    ?miss <- (MANIFEST::Mission (comm-OBDH-mass# nil) (datarate-penalty ?pen)
(payload-mass# ?m))
    =>
    (if (eq ?pen nil) then (bind ?pen 0))
    (if (eq ?pen 1) then (bind ?comm-mass-coeff 0.44)
        else (bind ?comm-mass-coeff 0.22)
        )
    (bind ?comm-mass (* ?m ?comm-mass-coeff))
    (modify ?miss (comm-OBDH-mass# ?comm-mass))
    )
```

Code 97: Rule to predict the communications subsystem mass in complexity-corrected mass budgets

As for the ADCS, its mass depends on the pointing requirements, which are set by the presence of the pointing penalty. The implementation of such rule is shown in Code 98.

```
(defrule MASS-BUDGET::design-ADCS-subsystem
    "Computes ADCS subsystem mass using rules of thumb"
    (declare (salience 10))
    ?miss <- (MANIFEST::Mission (ADCS-mass# nil) (ADCS-penalty ?pen) (payload-mass#
?m))
```

```
    =>
    (if (eq ?pen nil) then (bind ?pen 0))
    (if (eq ?pen 1) then (bind ?adcs-mass-coeff 0.44)
        else (bind ?adcs-mass-coeff 0.22))
    (bind ?adcs-mass (* ?m ?adcs-mass-coeff))
    (modify ?miss (ADCS-mass# ?adcs-mass))
    )
```

Code 98: Rule to predict the ADCS mass in complexity-corrected mass budgets

The mass of the thermal subsystem essentially depends on the presence of the thermal penalty as illustrated in Code 99.

```
(defrule MASS-BUDGET::design-thermal-subsystem
    "Computes thermal subsystem mass using rules of thumb"
    (declare (salience 10))
    ?miss <- (MANIFEST::Mission (thermal-mass# nil) (thermal-penalty ?pen) (payload-
mass# ?m))
    =>
    (if (eq ?pen nil) then (bind ?pen 0))
    (if (eq ?pen 1) then (bind ?thermal-mass-coeff 0.22)
        else (bind ?thermal-mass-coeff 0.037)
        )
    (bind ?thermal-mass (* ?m ?thermal-mass-coeff))
    (modify ?miss (thermal-mass# ?thermal-mass))
    )
```

Code 99: Rule to predict the thermal subsystem mass in complexity-corrected mass budgets

Finally, the mass of the propulsion subsystem is assumed to be essentially dependent on payload mass as shown in Code 100.

```
 (defrule MASS-BUDGET::design-propulsion-subsystem
    "Computes propulsion subsystem mass using rules of thumb"
    (declare (salience 10))
    ?miss <- (MANIFEST::Mission (propulsion-mass# nil) (payload-mass# ?m))
    =>

    (bind ?prop-mass (* ?m 0.14))
    (modify ?miss (propulsion-mass# ?prop-mass))
    )
```

Code 100: Rules computing a complexity-corrected satellite mass budget

The complete complexity-corrected satellite mass budget is thus given by the sum of all the subsystem masses, as shown in Code 101.

```
(defrule MASS-BUDGET::add-subsystem-masses
    "Computes the sum of subsystem masses"
    ?miss <- (MANIFEST::Mission (propulsion-mass# ?prop-mass&~nil) (structure-mass#
?struct-mass&~nil)
        (comm-OBDH-mass# ?comm-mass&~nil) (ADCS-mass# ?adcs-mass&~nil) (EPS-mass#
?eps-mass&~nil)
        (thermal-mass# ?thermal-mass&~nil) (payload-mass# ?payload&~nil) (satellite-
mass# nil))
```

```
    =>
    (bind ?sat-mass (+ ?prop-mass ?struct-mass ?eps-mass ?adcs-mass ?comm-mass
?payload ?thermal-mass))
    (modify ?miss (satellite-mass# ?sat-mass))
    )
```

Code 101: Rule computing total spacecraft mass in complexity-corrected mass budgets

9.2.4 Launch vehicle selection rules

The rules that take into account the knowledge from this section are described below. A function is created to determine whether a certain launch vehicle class has enough performance to put a certain mass in a certain orbit.

```
(deffunction sufficient-performance-Soyuz (?m ?type ?h ?i)
    ;; SSO
    (if (eq ?type SSO) then
        (bind ?perf (- 5260 (* 1.26667 (- ?h 400)))); SSO perf for Soyuz in kg
        (if (> ?perf (* ?m 1.3)) then ; 30% margin
            (return 1)
        else (return 0))
        ;; LEO polar
     elif ( eq ?type LEO ) then
        (bind ?perf (+ 5232 (* 0.0869 ?h) (* ?h ?h -0.00024))); LEO polar perf for
Soyuz in kg from user manual page 2-12
        (if (> ?perf (* ?m 1.3)) then ; 30% margin
            (return 1)
        else (return 0))
        ;; GTO
     elif (eq ?type GTO) then
        (bind ?perf 3060); SSO perf for Soyuz in kg
        (if (> ?perf (* ?m 1.3)) then ; 30% margin
            (return 1)
        else (return 0))
    else  (return 0))
  )

(deffunction sufficient-performance-Ariane5 (?m ?type ?h ?i)
    ;; SSO
    ;;(printout t ?m ?type ?h ?i crlf)
    (if (eq ?type SSO) then
        (bind ?perf 10000); SSO perf for Ariane5 in kg
        (if (> ?perf (* ?m 1.3)) then ; 30% margin
            (return 1)
        else (return 0))

        ;; LEO polar
     elif (eq ?type LEO) then
        (bind ?perf (- 20000 (* (/ 10000 400) (- ?h 400)))); LEO polar perf for
Ariane5 in kg from user manual page 2-12
        ;;(printout t ?perf crlf)
        (if (> ?perf (* ?m 1.3)) then ; 30% margin
            (return 1)
        else (return 0))
```

```
        ;; GTO
     elif (eq ?type GTO) then
        (bind ?perf 10000); GTO perf for Ariane in kg
        (if (> ?perf (* ?m 1.3)) then ; 30% margin
            (return 1)
        else (return 0))
    else (return 0))
  )
```

Code 102: Function that checks whether a certain launch vehicle class has enough performance to launch a certain payload into a certain orbit

These functions are used by a rule that eliminates launcher options that do not have enough performance.

```
 (deffunction sufficient-performance (?lv ?m ?type ?h ?i)
    (if (eq ?lv Ariane5-class) then (return (sufficient-performance-Ariane5 ?m ?type
?h ?i))
        elif (eq ?lv Soyuz-class) then (return (sufficient-performance-Soyuz ?m ?type
?h ?i))
     elif (eq ?lv Vega-class) then (return (sufficient-performance-Vega ?m ?type ?h
?i))
     elif (eq ?lv Pegasus-class) then (return (sufficient-performance-Pegasus ?m
?type ?h ?i))
        else (return 1)
        )
    )

(defrule LV-SELECTION::insufficient-performance
    "Eliminate options for which performance is not sufficient with margin"
    ?f <- (MANIFEST::Mission (Name ?miss) (launch-vehicle ?lv&~nil) (payload-mass#
?m&~nil)
        (orbit-type ?typ&~nil) (orbit-altitude# ?h&~nil) (orbit-inclination ?i&~nil))
    (test (neq (sufficient-performance ?lv (* 4 ?m) ?typ ?h ?i) 1)); assume payload-
to-bus mass ratio of 4
    =>
    (retract ?f)
    )
```

Code 103: Rule eliminating launchers that do not have enough performance

The same principle is applied to height, and diameter.

```
(deffunction large-enough-height (?lv ?dim); ?dim = (max-diam area height)
    (bind ?fairing-dimensions (get-launch-fairing-dimensions ?lv)); (diam height)
    (bind ?diam (nth$ 1 ?dim))
    (if (eq ?diam nil) then (return 0) else
        (if (> (nth$ 2 ?fairing-dimensions) (* 0.8 ?diam)) then
        (return 1)
        else (return 0)
        )))

(deffunction large-enough-area (?lv ?dim); ?dim = (max-diam area height)
    (bind ?fairing-dimensions (get-launch-fairing-dimensions ?lv)); (diam height)
    ;(bind ?diam (nth$ 1 ?dim))
    (bind ?area (nth$ 2 ?dim))
```

```
    (if (eq ?area nil) then (return 0))
    (if (> (* (nth$ 1 ?fairing-dimensions) (nth$ 2 ?fairing-dimensions)) (* 0.8
?area)) then
        (return 1)
        else (return 0)
        )
    )

 (defrule LV-SELECTION::insufficient-fairing-height
    "Eliminate options for which fairing height is not sufficient with margin"
    ?f <- (MANIFEST::Mission (Name ?miss) (launch-vehicle ?lv&~nil) (payload-
dimensions# $?dim))
    (test (neq (large-enough-height ?lv ?dim) 1))
    =>
    ;(printout t "Insufficient diameter of " ?lv " for mission " ?miss crlf)
    (retract ?f)
    )

 (defrule LV-SELECTION::insufficient-fairing-area
    "Eliminate options for which fairing area is not sufficient with margin"
    ?f <- (MANIFEST::Mission (Name ?miss ) (launch-vehicle ?lv&~nil) (payload-
dimensions# $?dim))
    (test (neq (large-enough-area ?lv ?dim) 1))
    =>
    (retract ?f)
    )
```

Code 104: Rules eliminating launchers that do not have enough volume

Finally, when more than one launcher option is available, the most expensive options are discarded, as shown below.

```
 (defrule LV-SELECTION::eliminate-more-expensive-launchers
    "From all feasible options, eliminate the most expensive ones"
    (declare (salience -5))
    ?m1 <- (MANIFEST::Mission (Name ?name) (launch-vehicle ?lv1&~nil) (launch-cost#
?c1&~nil))
    ?m2 <- (MANIFEST::Mission (Name ?name) (launch-vehicle ?lv2&~nil) (launch-cost#
?c2&~nil))
    (test (neq ?lv1 ?lv2))
    =>
    (if (< ?c1 ?c2) then (retract ?m2)
        else (retract ?m1) )
    )
```

Code 105: Rules eliminating the most expensive launch vehicles

9.2.5 Standard bus selection rules

Standard bus selection rules were incorporated in order to allow for the acquisition of commercially available buses instead of the development of new ad-hoc buses. This results in general in cost savings due to reduced bus development cost. These rules were only utilized for the EOS case study. Three buses were considered: TRW's T-330, Ball Aerospace's BCP-2000, and Orbital Science's Pegastar.

The rules that select the optimal standard bus for a given payload are provided below.

```
(deffunction get-bus-cost (?bus)
    (if (eq ?bus T330-class) then (return 140000)
        elif (eq ?bus BCP2000-class) then (return 40000)
        elif (eq ?bus Pegastar-class) then (return 15000)
        else (return 1000000)))

(deffunction enough-mass (?bus ?m)
    (if (eq ?bus Pegastar-class) then (if (< ?m 70) then (return TRUE) else (return
FALSE)))
    (if (eq ?bus BCP2000-class) then (if (< ?m 300) then (return TRUE) else (return
FALSE)))
    (if (eq ?bus T330-class) then (if (< ?m 1300) then (return TRUE) else (return
FALSE)))
    (if (eq ?bus dedicated-class) then (return TRUE)) (return FALSE))

(deffunction enough-power (?bus ?p)
    (if (eq ?bus Pegastar-class) then (if (< ?p 70) then (return TRUE) else (return
FALSE)))
    (if (eq ?bus BCP2000-class) then (if (< ?p 300) then (return TRUE) else (return
FALSE)))
    (if (eq ?bus T330-class) then (if (< ?p 1300) then (return TRUE) else (return
FALSE)))
    (if (eq ?bus dedicated-class) then (return TRUE)) (return FALSE))

(defrule BUS-SELECTION::standard-bus-not-enough-payload-mass
    "Eliminate options for which performance is not sufficient with margin"
    ?f <- (MANIFEST::Mission (Name ?miss) (standard-bus ?bus&~nil) (payload-mass#
?m&~nil)
        (payload-power# ?p&~nil) (payload-data-rate# ?h&~nil) (payload-dimensions#
$?dim))
    (test (neq (enough-mass ?bus ?m) TRUE));
    =>
    (retract ?f))

(defrule BUS-SELECTION::standard-bus-not-enough-payload-power
    "Eliminate options for which performance is not sufficient with margin"
    ?f <- (MANIFEST::Mission (Name ?miss) (standard-bus ?bus&~nil) (payload-mass#
?m&~nil)
        (payload-power# ?p&~nil) (payload-data-rate# ?h&~nil) (payload-dimensions#
$?dim))
    (test (neq (enough-power ?bus ?p) TRUE));
    =>
    (retract ?f))
```

Code 106: Standard bus selection rules

These rules are completed by a rule that eliminates the more expensive buses.

9.2.6 Synergy rules

Synergy rules create new measurments and data products from combinations of measurements and data products. Synergy rules are at the core of this methodology because they model emergent behavior in the value delivery loop, which is a key feature of a system architecting model.

The following synergy rules model the application of the most common data processing algorithms: resampling in time and space, and disaggregation schemes in the space, time, and spectral domains.

```
(defrule SYNERGIES::spatial-disaggregation "A frequent coarse spatial resolution
measurement can be combined with a sparse high spatial resolution measurement to
produce a frequent high spatial resolution measurement with average accuracy"

    ?m1 <- (REQUIREMENTS::Measurement (Parameter ?p&~nil) (Temporal-resolution
?tr1&~nil) (Horizontal-Spatial-Resolution ?hsr1&~nil) (Accuracy ?a1&~nil) (Id ?id1)
(taken-by ?ins1) (synergy-level# ?s1&:(< ?s1 1)))
    ?m2 <- (REQUIREMENTS::Measurement (Parameter ?p&~nil) (Temporal-resolution
?tr2&~nil) (Horizontal-Spatial-Resolution ?hsr2&~nil) (Accuracy ?a2&~nil) (Id ?id2)
(taken-by ?ins2) (synergy-level# ?s2&:(< ?s2 1)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    (test (neq ?id1 ?id2))
      =>
      (duplicate ?m1 (Parameter ?p) (Temporal-resolution (eval (fuzzy-max Temporal-
resolution ?tr1 ?tr2)))
            (Horizontal-Spatial-Resolution (eval (fuzzy-max Horizontal-Spatial-
Resolution ?hsr1 ?hsr2)))
            (Accuracy (eval (fuzzy-avg ?a1 ?a2))) (synergy-level# (+ (max ?s1 ?s2)
1))
            (Id (str-cat ?id1 "-disaggregated" ?id2))
            (taken-by (str-cat ?ins1 "-" ?ins2 "-disaggregated")))

    (duplicate ?m2 (Parameter ?p) (Temporal-resolution (eval (fuzzy-max Temporal-
resolution ?tr1 ?tr2)))
            (Horizontal-Spatial-Resolution (eval (fuzzy-max Horizontal-Spatial-
Resolution ?hsr1 ?hsr2)))
            (Accuracy (eval (fuzzy-avg ?a1 ?a2))) (synergy-level# (+ (max ?s1 ?s2)
1))
            (Id (str-cat ?id2 "-disaggregated" ?id1))
            (taken-by (str-cat ?ins2 "-" ?ins1 "-disaggregated"))))

(defrule SYNERGIES::spatial-disaggregation-hyperspectral "A hyperspectral coarse
spatial resolution measurement can be combined with a multispectral high spatial
resolution measurement to produce a high spatial resolution hyperspectral measurement
with lower accuracy"

    ?m1 <- (REQUIREMENTS::Measurement (Parameter ?p&~nil) (Temporal-resolution
?tr1&~nil) (Spectral-sampling Multispectral-10-100-channels) (Horizontal-Spatial-
Resolution ?hsr1&~nil) (Accuracy ?a1&~nil) (Id ?id1) (taken-by ?ins1) (synergy-level#
```

```
?s1&:(< ?s1 1)))
    ?m2 <- (REQUIREMENTS::Measurement (Parameter ?p&~nil) (Temporal-resolution
?tr2&~nil) (Spectral-sampling Hyperspectral-100-channels-or-more) (Horizontal-
Spatial-Resolution ?hsr2&~nil) (Accuracy ?a2&~nil) (Id ?id2) (taken-by ?ins2)
(synergy-level# ?s2&:(< ?s2 1)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    (test (neq ?id1 ?id2))
      =>
    (duplicate ?m1 (Temporal-resolution (eval (fuzzy-max Temporal-resolution ?tr1
?tr2)))
            (Horizontal-Spatial-Resolution (eval (fuzzy-max Horizontal-Spatial-
Resolution ?hsr1 ?hsr2)))
            (Accuracy (eval (fuzzy-avg ?a1 ?a2))) (synergy-level# (+ (max ?s1 ?s2)
1))
            (Spectral-sampling Hyperspectral-100-channels-or-more)
            (Id (str-cat ?id1 "-hyp-disaggregated-" ?id2))
            (taken-by (str-cat ?ins1 "-" ?ins2 "-hyp-disaggregated"))))

(defrule SYNERGIES::spatial-averaging "any image can be averaged out in space to
provide a new, better accuracy, coarser resolution image"
    ?m <- (REQUIREMENTS::Measurement (Parameter ?p&~nil)  (Horizontal-Spatial-
Resolution ?hsr1&~nil) (Accuracy ?a2&~nil) (Id ?id1) (taken-by ?ins1) (synergy-level#
?s1&:(< ?s1 1)))
    (test (>= (SameOrBetter Horizontal-Spatial-Resolution ?hsr1 Low-1km-10km) 0))
    (test (>= (SameOrBetter Accuracy High ?a2) 1))
      =>
    (duplicate ?m (Id (str-cat ?id1 "-space-averaged")) (Horizontal-Spatial-
Resolution (eval (Worsen Horizontal-Spatial-Resolution ?hsr1))) (Accuracy (eval
(Improve Accuracy ?a2))) (taken-by (str-cat ?ins1 "-space-averaged")) (synergy-level#
(+ ?s1 1))))

(defrule SYNERGIES::time-averaging "any image can be averaged out in time to provide
a new, better accuracy, sparser temporal resolution image"
    ?m <- (REQUIREMENTS::Measurement (Parameter ?p&~nil)  (Temporal-resolution#
?tr1&~nil)
        (rms-variable-measurement# ?rms&:(> ?rms )) (Id ?id1) (taken-by ?ins1)
(synergy-level# ?s1&:(< ?s1 1)))
    (test (>= (SameOrBetter Temporal-resolution ?tr1 Low-3days-1-week) 0))
    (test (>= (SameOrBetter Accuracy High ?rms) 1))
      =>
    (duplicate ?m (Id (str-cat ?id1 "-time-averaged")) (Temporal-resolution (eval
(Worsen Temporal-resolution ?tr1))) (Accuracy (eval (Improve Accuracy ?a2))) (taken-
by (str-cat ?ins1 "-time-averaged")) (synergy-level# (+ ?s1 1))))
```

Code 107: Synergy rules modeling major data processing and assimilation schemes

Other synergy rules capture algorithms concerning a specific discipline. The following rules concern solid Earth measurements.

```
(defrule SYNERGIES::ocean-mass-distribution-from-gravity "Ocean mass distribution can
be inferred from precise gravity measurements"
        ?grav <- (REQUIREMENTS::Measurement (Parameter "5.1.1 Geoid and gravity field
variations") (Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id1))
```

```
    =>
    (duplicate ?grav (Id (str-cat ?id1 "-syn")) (Parameter "3.2.6 Ocean mass
distribution")))


(defrule SYNERGIES::seafloor-topography
    "Seafloor topography measurements can be inferred from sea level height and ocean
mass distribution measurements"
    ?slh <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height") (Id
?id1))
    ?grv <- (REQUIREMENTS::Measurement (Parameter "3.2.6 Ocean mass distribution")
(Id ?id2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>
    (duplicate ?slh (Id (str-cat ?id1 "-syn-" ?id2)) (Parameter "3.2.2 seafloor
topography")))
```

Code 108: Synergy rules from the solid Earth community

The following rules concern ocean altrimetry measurements.

```
(defrule altimetry-err-budget-POD
    "If an instrument providing precise orbit determinations is sharing a platform
with an altimeter, then the POD contribution in the altimetry error budget is
decreased to 2 cm if the orbit is high enough to have low drag, and 4 cm if it is a
lower orbit, i.e. 800km or less"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height")
        (orbit-altitude# ?h) (rms-system-POD# 10.0) (taken-by ?ra))
    (REQUIREMENTS::Measurement (Parameter "A9.Precise Orbit Determination") (taken-by
?gps) (Id ?id2))
    (CAPABILITIES::Manifested-instrument (Intent "Orbitographers") (Name ?gps))
    (SYNERGIES::cross-registered-instruments (instruments $?ins) (degree-of-cross-
registration spacecraft))
    (test (member$ ?gps $?ins))
    (test (member$ ?ra $?ins))
    =>
    (if (>= ?h 800) then (bind ?new-rms-POD 2.0) else (bind ?new-rms-POD 4.0))
    (modify ?meas (rms-system-POD# ?new-rms-POD) (taken-by (str-cat ?ra "-POD-"
?gps))))

(defrule altimetry-err-budget-tropo-wet-correction
    "If an atmospheric humidity measurement is cross-correlated with an altimetry
measurement, the tropospheric correction contribution in the altimetry error budget
is decreased by a factor that is directly proportional to the accuracy of the
humidity measurement"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height") (taken-
by ?ra) (rms-system-tropoH2O# ?rms-tropo&:(>= ?rms-tropo 1)) (Id ?id1))
    (REQUIREMENTS::Measurement (Parameter "1.8.1 H2O") (Id ?id2) (taken-by ?mwr)
```

```
(rms-system-instrument# ?rms-mwr &~nil));
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    =>
    (bind ?new-rms-tropo (* ?rms-mwr 0.6)); 0.6cm per K of rms error in the MWR
(christensen et al 93)
    (modify ?meas (rms-system-tropoH2O# ?new-rms-tropo) (taken-by (str-cat ?ra "-wet-
" ?mwr)))
    )

(defrule altimetry-err-budget-ionospheric-correction
    "If a measurement of inospheric total electron content is cross-correlated with
an altimetry measurement, the ionospheric correction contribution in the altimetry
error budget is decreased by a factor that is directly proportional to the accuracy
of the inospheric instrument"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height") (taken-
by ?ra) (rms-system-ionosphere# ?rms-iono&:(>= ?rms-iono 1)) (Id ?id1))
   (REQUIREMENTS::Measurement (Parameter "A8.Total electron content in ionosphere")
(Accuracy ?acc) (Id ?id2) (taken-by ?io))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    =>
    (if (eq ?acc High) then (bind ?new-rms-iono (/ ?rms-iono 5)) else (bind ?new-rms-
iono (/ ?rms-iono 3))); assume for now that the effect of instrument is to divide
contribution to error by 3
    (modify ?meas (rms-system-ionosphere# ?new-rms-iono) (taken-by (str-cat ?ra "-
iono-" ?io))))

(defrule SYNERGIES::altimetry-variable-error
   "If several independent measurements are gathered for example in a constellation,
    the variable ensemble measurement error decreases as 1/sqtr(2+N), assuming 2
altimeters are already being flown (conversation w/ Dr Sarah Gille, UCSD)"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height")
         (rms-variable-measurement# ?rms-var&4.0) (num-of-planes# ?np&~nil) (taken-by
?ra)
        (num-of-sats-per-plane# ?ns&~nil))

    =>
    (bind ?new-rms-var (/ ?rms-var (sqrt (+ 2 (* ?np ?ns))))); assume that there are
2 other altimeters
    (modify ?meas (rms-variable-measurement# ?new-rms-var)))

(defrule SYNERGIES::altimetry-variable-error-when-nil
   "If several independent measurements are gathered for example in a constellation,
    the variable ensemble measurement error decreases as 1/sqtr(2+N), assuming 2
altimeters are already being flown (conversation w/ Dr Sarah Gille, UCSD)"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height")
```

```
        (rms-variable-measurement# ?rms-var&4.0) (num-of-planes# ?np&nil) (taken-by
?ra) (num-of-sats-per-plane# ?ns&nil))

    =>
    (bind ?new-rms-var (/ ?rms-var (sqrt 3))); assume that there are 2 other
altimeters
    (modify ?meas (rms-variable-measurement# ?new-rms-var)))


(defrule altimetry-err-budget-tropo-dry-correction
    "If an atmospheric pressure measurement is cross-correlated with an altimetry
measurement, the dry tropospheric correction contribution in the altimetry error
budget is decreased by a factor that is directly proportional to the accuracy of the
pressure measurement"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height") (taken-
by ?ra) (rms-system-tropo-dry# ?rms-tropo&:(>= ?rms-tropo 0.5)) (Id ?id1))
    (REQUIREMENTS::Measurement (Parameter "1.3.4 Atmospheric pressure") (Id ?id2)
(taken-by ?dry) (rms-system-instrument# ?rms-mwr &~nil))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    =>
    (bind ?new-rms-tropo (* ?rms-mwr 0.2)); 0.2cm per K of rms error in the MWR
    (modify ?meas (rms-system-tropo-dry# ?new-rms-tropo) (taken-by (str-cat ?ra "-
dry-" ?dry))))

(defrule SYNERGIES::altimetry-tidal-error
   "If several independent measurements are gathered for example in a constellation,
    the variable ensemble measurement error decreases as 1/sqtr(N)"

    (declare (no-loop TRUE))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height")
         (rms-system-tides# 10.0) (orbit-type ?orb&~nil) (taken-by ?ra))
    =>
    (if (eq ?orb SSO) then (bind ?rms-tide 4.0) else (bind ?rms-tide 1.0))
    (modify ?meas (rms-system-tides# ?rms-tide)))

(defrule SYNERGIES::altimetry-err-total
    "If an instrument providing precise orbit determinations is sharing a platform
with an altimeter, then the POD contribution in the altimetry error budget is
decreased from 10-15cm to 2-5 cm"

    (declare (salience -30) (no-loop TRUE) )
    ?meas <- (REQUIREMENTS::Measurement (Parameter "3.2.1 Sea level height")
         (rms-system-POD# ?rms-POD)  (rms-system-tropoH2O# ?rms-tropo) (rms-system-
ionosphere# ?rms-iono)
        (rms-system-instrument# ?rms-ins) (rms-variable-measurement# ?rms-var)
        (rms-system-tides# ?rms-tides) (rms-system-tropo-dry# ?rms-tropo-dry) (rms-
total# 100.0))

    =>
    (bind ?rms-total (sqrt (+ (** ?rms-POD 2) (** ?rms-iono 2) (** ?rms-tropo 2)  (**
?rms-ins 2)
```

```
       (** ?rms-tides 2) (** ?rms-tropo-dry 2)  (** ?rms-var 2))))
    (modify ?meas (rms-total# ?rms-total)))
```

Code 109: Synergy rules from the ocean altimetry community

The following rules concern scatterometry measurements.

```
(defrule SYNERGIES::sea-surface-winds-combining-SAR-and-scatterometer
    "sea surface winds from sar and scatterometers can be combined to increase
sensitivity. See Monaldo et al, TGRS 2004 Vol 42, Iss 2"

    (declare (no-loop TRUE))
    ?m1 <- (REQUIREMENTS::Measurement (Parameter "3.4.1 Ocean surface wind speed")
(taken-by ?sar) (rms-variable-measurement# ?err1&~nil))
    ?m2 <- (REQUIREMENTS::Measurement (Parameter "3.4.1 Ocean surface wind speed")
(taken-by ?scat&:(neq ?scat ?sar)) (rms-variable-measurement# ?err2&~nil))
    (CAPABILITIES::Manifested-instrument (Intent "Imaging MW radars -SAR-") (Name
?sar))
    (CAPABILITIES::Manifested-instrument (Intent "Radar scatterometer") (Name ?scat))
    (SYNERGIES::cross-registered-instruments (instruments $?ins) (degree-of-cross-
registration spacecraft))
    (test (member$ ?sar $?ins))
    (test (member$ ?scat $?ins))
    =>
    (modify ?m2 (rms-variable-measurement# ?sens (* 0.75 (min ?err1 ?err2))))
    )

(defrule SYNERGIES::sea-surface-winds-with-high-winds
    "C-band scatterometry offers better sensitivity than Ka/Ku at high winds.
    See Decadal Survey report on XOVWM mission"

    (declare (no-loop TRUE))
    ?m <- (REQUIREMENTS::Measurement (Parameter "3.4.1 Ocean surface wind speed")
(taken-by ?ins) (sensitivity-in-high-winds nil))
    (CAPABILITIES::Manifested-instrument (Intent "Radar scatterometer") (Name ?ins)
(Spectral-region ?sp))
    =>
    (if (eq ?sp MW-C) then (bind ?sens High) else (bind ?sens Low))
    (modify ?m (sensitivity-in-high-winds ?sens)))

(defrule SYNERGIES::ocean-wind-vector-sensitivity-high-winds
    "If a Ku-band measurement of ocean winds is combined with a C-band measurement
    then the sensitivity at high speed improves"

    (declare (salience 10) (no-loop TRUE))
    ?C <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter ?p)
(Horizontal-Spatial-Resolution Very-low-10-100km) (sensitivity-in-high-winds High))
    ?Ku <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter ?p)
(Horizontal-Spatial-Resolution Low-1km-10km) (sensitivity-in-high-winds Low))
    (test (meas-group ?p 3.4.0))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>
```

```
    (duplicate ?Ku (sensitivity-in-high-winds High) (taken-by (str-cat ?ins1 "-syn-"
?ins2)) (Id (str-cat ?id1 "-syn-" ?id2 ))))

(defrule SYNERGIES::ocean-wind-vector-complete
    "If a Ku-band measurement of ocean winds is combined with a C-band measurement
    and a passive measurement then this is the optimal wind measurement"

    (declare (salience 10) (no-loop TRUE))
    ?C <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter ?p)
(Horizontal-Spatial-Resolution Very-low-10-100km) (sensitivity-in-high-winds High))
    ?Ku <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter ?p)
(Horizontal-Spatial-Resolution Low-1km-10km) (sensitivity-in-high-winds Low))
    ?X <- (REQUIREMENTS::Measurement (Id ?id3) (taken-by ?ins3) (Parameter ?p)
(Horizontal-Spatial-Resolution Very-low-10-100km) (sensitivity-in-rain High))
    (test (meas-group ?p 3.4.0))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m)) (test (member$ ?id2 $?m)) (test (member$ ?id3 $?m))
    (not (REQUIREMENTS::Measurement (Parameter ?p) (Horizontal-Spatial-Resolution
Low-1km-10km) (sensitivity-in-high-winds High) (sensitivity-in-rain High) (rms-
system-tropoH2O# Low)))

    =>

    (duplicate ?Ku (sensitivity-in-high-winds High) (sensitivity-in-rain High) (rms-
system-tropoH2O# Low) (taken-by (str-cat ?ins1 "-syn-" ?ins2 "-syn-" ?ins3)) (Id
(str-cat ?id1 "-syn-" ?id2 "-syn-" ?id3))))

(defrule SYNERGIES::ocean-wind-vector-sensitivity-in-rain
    "If a Ku-band measurement of ocean winds is combined with a C-band measurement
    then the sensitivity at high speed improves"
    (declare (salience 10) (no-loop TRUE))
    ?X <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter ?p)
(Horizontal-Spatial-Resolution Very-low-10-100km) (sensitivity-in-rain High))
    ?Ku <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter ?p)
(Horizontal-Spatial-Resolution Low-1km-10km) (sensitivity-in-rain Low))
    (test (meas-group ?p 3.4.0))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (duplicate ?Ku (sensitivity-in-rain High) (taken-by (str-cat ?ins1 "-syn-"
?ins2)) (Id (str-cat ?id1 "-syn-" ?id2 )))
    )

(defrule SYNERGIES::ocean-wind-vector-atmospheric-correction
    "If a Ku-band measurement of ocean winds is combined with a X-band passive
measurement then the accuracy of the retrieval improves"

    (declare (salience 10) (no-loop TRUE))
    ?X <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter ?p)
(Horizontal-Spatial-Resolution Very-low-10-100km) (rms-system-tropoH2O# Low))
    ?Ku <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter ?p)
```

```
(Horizontal-Spatial-Resolution Low-1km-10km) (rms-system-tropoH2O# High))
    (test (meas-group ?p 3.4.0))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (duplicate ?Ku (rms-system-tropoH2O# Low) (taken-by (str-cat ?ins1 "-syn-"
?ins2)) (Id (str-cat ?id1 "-syn-" ?id2 ))))
```

Code 110: Synergy rules from the scatterometry community

The following rules concern ocean biology measurements.

```
(defrule SYNERGIES::river-plumes-from-ocean-color "River plumes and sediment fluxes
can be measured with an ocean color measurement of sufficient horizontal spatial
resolution"

        ?oc <- (REQUIREMENTS::Measurement (Parameter "3.1.1 Ocean color - 410-680nm
(Chlorophyll absorption and fluorescence, pigments, phytoplankton, CDOM)")
(Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id1) (Temporal-resolution
?tr))
    (test (SameOrBetter Horizontal-Spatial-Resolution ?hsr High-10-100m))
    (test (SameOrBetter Temporal-resolution ?tr High-12h-24h))
    (REQUIREMENTS::Measurement (Parameter "3.1.2 Extended ocean color - UV (enhanced
DOC, CDOM)") (Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (duplicate ?oc (Id (str-cat ?id1 "-syn")) (Parameter "3.2.5 river plumes/sediment
fluxes")))
```

Code 111: Synergy rules from the ocean biology community

The following rules concern land topography measurements.

```
 (defrule SYNERGIES::add-multi-angular-capability
    "If a multi-angular radiometer is combined with another imager
    then the common measurement combines the characteristics of the two"

    (declare (no-loop TRUE) (salience 5))
    ?no <- (REQUIREMENTS::Measurement (Parameter ?p) (taken-by ?ins1) (Id ?id1)
(ThreeD ?td1 &~ Full-3D) (synergy-level# ?s1&:(< ?s1 1)))
    (REQUIREMENTS::Measurement (Parameter ?p) (taken-by ?ins2) (Id ?id2) (ThreeD ?td2
&~ No-3D &~ N-A &~ nil) (synergy-level# ?s2&:(< ?s2 1)))
    (or (test (meas-group ?p 1.0.0)) (test (meas-group ?p 2.0.0)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    (test (> (SameOrBetter ThreeD ?td2 ?td1) 0))

    =>
```

```
     (duplicate ?no (ThreeD ?td2) (Id (str-cat ?id1 "-multiang-" ?id2))  (taken-by
(str-cat ?ins1 "-multiang-" ?ins2 )) (synergy-level# (+ 1 (max ?s1 ?s2))))
    )

(defrule SYNERGIES::HSR-TR-ThreeD-combination-scheme
    "Combination of a MODIS-like instrument with an ASTER like instrument with a MISR
like instrument"

    (declare (no-loop TRUE) (salience 10))
    ?MODIS <- (REQUIREMENTS::Measurement (Parameter ?p) (Accuracy High) (Horizontal-
Spatial-Resolution Medium-100m-1km) (Temporal-resolution Medium-1day-3days) (taken-by
?ins1) (Id ?id1) (ThreeD ?td1 &~ Full-3D))
    ?ASTER <- (REQUIREMENTS::Measurement (Parameter ?p) (Accuracy High) (Horizontal-
Spatial-Resolution High-10-100m) (Temporal-resolution Very-low-1-3-weeks) (taken-by
?ins2) (Id ?id2) (ThreeD ?td2 &~ No-3D &~ N-A &~ nil))
    ?MISR <- (REQUIREMENTS::Measurement (Parameter ?p) (Accuracy High) (Horizontal-
Spatial-Resolution Medium-100m-1km) (Temporal-resolution Very-low-1-3-weeks) (taken-
by ?ins3) (Id ?id3) (ThreeD Full-3D))
    (or (test (meas-group ?p 1.0.0)) (test (meas-group ?p 2.0.0)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    (test (member$ ?id3 $?m))

    =>

     (duplicate ?MODIS (ThreeD Full-3D) (Horizontal-Spatial-Resolution High-10-100m)
(Id (str-cat ?id1 "-disagg-" ?id2 "-multiang-" ?id3))  (taken-by (str-cat ?ins1 "-
disaggr-" ?ins2 "-multiang-" ?ins3))))
```

Code 112: Synergy rules from the land topography community

The following rules concern hydrology measurements.

```
(defrule SYNERGIES::groundwater-storage-from-gravity "Groundwater storage can be
inferred from precise gravity measurements"

        ?grav <- (REQUIREMENTS::Measurement (Parameter "5.1.1 Geoid and gravity field
variations") (Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id1))

    =>

    (duplicate  ?grav  (Id  (str-cat  ?id1  "-syn"))  (Parameter  "2.7.3  groundwater
storage")))
```

Code 113: Synergy rules from the hydrology community

The following rules concern cryospheric measurements.

```
 (defrule SYNERGIES::glacier-mass-balance-from-gravity "Glacier mass balance
measurements can be inferred from precise gravity measurements using ice topography
measurements"

        ?grav <- (REQUIREMENTS::Measurement (Parameter "5.1.1 Geoid and gravity field
variations") (Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (taken-by ?ins1)
(Id ?id1))
```

```
      ?topo <- (REQUIREMENTS::Measurement (Parameter "4.1.5 Ice Sheet topography")
(Id ?id2) (taken-by ?ins2))
      (SYNERGIES::cross-registered (measurements $?m))
      (test (member$ ?id1 $?m))
      (test (member$ ?id2 $?m))

    =>

    (duplicate ?topo (Id (str-cat ?id1 "-syn" ?id2)) (taken-by (str-cat ?ins1 "-syn"
?ins2)) (Parameter "4.1.3 glacier mass balance")))

(defrule SYNERGIES::cross-instrument-calibration-heritage-SCLP
    "If the SCLP radiometer is flown together with the SAR, then the calibration of
the SAR is better"

    ?SAR <- (REQUIREMENTS::Measurement (Parameter "4.2.1 snow-water equivalence")
(Illumination Active) (On-board-calibration ?obc&~High) (taken-by ?ins1) (Id ?id1))
    (REQUIREMENTS::Measurement (Parameter "4.2.1 snow-water equivalence")
(Illumination Passive) (taken-by ?ins2) (Id ?id2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (modify ?SAR (On-board-calibration High) (taken-by (str-cat ?ins1 "-syn-" ?ins2))
         (Id (str-cat ?id1 "-syn-" ?id2 ))))
```

Code 114: Synergy rules from the cryosphere community

The following rules concern atmospheric measurements.

```
(defrule SYNERGIES::visible-atmospheric-plume-from-aerosols "Visible atmospheric
plumes can be measured from high temporal and spatial resolution multispectral
aerosol measurements"

        ?ae <- (REQUIREMENTS::Measurement (Parameter "1.1.1 aerosol height/optical
depth") (Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Temporal-resolution
?tr) (Id ?id1))
    (test (SameOrBetter Horizontal-Spatial-Resolution ?hsr High-10-100m))
    (test (SameOrBetter Temporal-resolution ?tr High-12h-24h))
    =>
    (duplicate ?ae (Id (str-cat ?id1 "-syn")) (Parameter "1.8.16 Visible atmospheric
plumes")))

 (defrule SYNERGIES::black-carbon-from-aerosols "Black carbon can be measured from
polarimetric aerosol measurements"

        ?ae <- (REQUIREMENTS::Measurement (Parameter "1.1.2 aerosol shape,
composition, physical and chemical properties") (Polarimetry yes) (Id ?id1))

    =>

    (duplicate ?ae (Id (str-cat ?id1 "-syn")) (Parameter "1.8.13 Black carbon and
other polluting aerosols")))
```

```
(defrule SYNERGIES::surface-composition-in-all-spectrum
    "Surface composition measurements in different regions of the spectrum can be
combined"

    ?VNIR <- (REQUIREMENTS::Measurement (Parameter ?p) (Spectral-region opt-
VNIR+SWIR) (Temporal-resolution ?tr1&~nil) (Spectral-sampling ?ss1&~nil) (Id ?id1)
(taken-by ?ins1))
    ?TIR <- (REQUIREMENTS::Measurement (Parameter ?p) (Spectral-region opt-TIR)
(Temporal-resolution ?tr2&~nil) (Spectral-sampling ?ss2&~nil)  (Id ?id2) (taken-by
?ins2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    (or (test (meas-group ?p 2.0.0)) (test (meas-group ?p 3.7.0)))

    =>

    (duplicate ?TIR (Spectral-region opt-VNIR+SWIR+TIR) (Temporal-resolution (fuzzy-
min Temporal-resolution ?tr1 ?tr2)) (Spectral-sampling (fuzzy-max Spectral-sampling
?ss1 ?ss2)) (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (Id (str-cat ?id1 "-syn-"
?id2))))

(defrule SYNERGIES::MW-and-IR-sounders-tropo-sensitivity
    "A MW sounder increases the sensitivity in the troposphere of the IR sounder by
providing all weather capability. This rule comes from AIRS-AMSU in EOS"

    ?IR <- (REQUIREMENTS::Measurement (Parameter "1.2.1 Atmospheric temperature
fields")
        (Spectral-region ?sr1) (sensitivity-in-low-troposphere-PBL ?sensIR)
        (Id ?id1) (taken-by ?ins1) (synergy-level# ?s1&:(< ?s1 1)))
    ?MW <- (REQUIREMENTS::Measurement (Parameter "1.2.1 Atmospheric temperature
fields")
        (Spectral-region ?sr2) (Id ?id2) (taken-by ?ins2) (synergy-level# ?s2&:(< ?s2
1)))
    (test (integerp (str-index MW ?sr1)))
    (test (integerp (str-index opt ?sr2)))
    (test (neq ?sensIR High))

    =>

    (duplicate ?IR (sensitivity-in-low-troposphere-PBL High)
        (Id (str-cat ?id1 "-syn-" ?id2))
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 ))
        (synergy-level# (+ (max ?s1 ?s2) 1))
        ))

(defrule SYNERGIES::add-all-weather-capability
    "If two measurements are of the same parameter and one has all weather capability
    and the other does not, we assume that we can assimilate them and have an all
     weather capability measurement"

    (declare (no-loop TRUE) (salience -50))
    ?no <- (REQUIREMENTS::Measurement (Parameter ?p) (All-weather no) (Id ?id1)
(taken-by ?ins1) (synergy-level# ?s1&:(< ?s1 3)))
```

```
    (REQUIREMENTS::Measurement (Parameter ?p) (All-weather yes) (Id ?id2) (taken-by
?ins2) (synergy-level# ?s2&:(< ?s2 1)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (duplicate ?no (All-weather yes) (Id (str-cat ?id1 "-syn-" ?id2))
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (synergy-level# (+ (max ?s1 ?s2)
1))))

(defrule SYNERGIES::add-cloud-mask
    "If an imager is cross-registered with another instrument that does not provide
cloud mask info, the imager can be used to obtain cloud-cleared images of the second
instrument"

    (declare (no-loop TRUE) (salience 10))
    ?no <- (REQUIREMENTS::Measurement (Parameter ?p) (cloud-cleared no) (Id ?id1)
(taken-by ?ins1) (synergy-level# ?s1&:(< ?s1 2)))
    (REQUIREMENTS::Measurement (Parameter "1.5.4 cloud mask") (Id ?id2) (taken-by
?ins2) (synergy-level# ?s2&:(< ?s2 1)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (modify ?no (cloud-cleared yes) (Id (str-cat ?id1 "-mask-" ?id2)) (taken-by (str-
cat ?ins1 "-mask-" ?ins2 )) (synergy-level# (+ 1 (max ?s1 ?s2))))
    (assert (SYNERGIES::cross-registered (measurements (insert$ $?m (+ 1 (length$
$?m)) (str-cat ?id1 "-mask-" ?id2 )))))
    )

(defrule SYNERGIES::add-sensitivity-in-cirrus-from-mmw-measurement
    "If two measurements are of the same parameter and one has all weather capability
    and the other does not, we assume that we can assimilate them and have an all
     weather capability measurement"
    (declare (no-loop TRUE) (salience -50))
    ?no <- (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-in-cirrus ?s&~High)
(Id ?id1) (taken-by ?ins1) (synergy-level# ?s1&:(< ?s1 2)))
    (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-in-cirrus High) (Id ?id2)
(taken-by ?ins2) (synergy-level# ?s2&:(< ?s2 1)))
    (not (REASONING::stop-improving (Measurement ?p)))

    =>

    (duplicate ?no (sensitivity-in-cirrus High) (Id (str-cat ?id1 "-syn-" ?id2))
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (synergy-level# (+ 1 (max ?s1
?s2)))))

(defrule SYNERGIES::add-sensitivity-in-convective-clouds
    "If two measurements are of the same parameter and one has high sensitivity over
convective clodus and the other does not, we assume that we can assimilate them and
have a better measurement with  sensitivity in convective clouds"
```

```
    (declare (no-loop TRUE) (salience -50))
    ?no <- (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-in-convective-
clouds ?s&~High) (Id ?id1) (taken-by ?ins1) (synergy-level# ?s1&:(< ?s1 2)))
    (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-in-convective-clouds High)
(Id ?id2) (taken-by ?ins2) (synergy-level# ?s2&:(< ?s2 1)))
    (not (REASONING::stop-improving (Measurement ?p)))

    =>

    (duplicate ?no (sensitivity-in-convective-clouds High) (Id (str-cat ?id1 "-syn-"
?id2))
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (synergy-level# (+ 1 (max ?s1
?s2)))))

(defrule SYNERGIES::column-vs-profile-chemistry-measurements
    "If we have a profile measurement in addition to a column measurement, both are
improved as a result because of two independents measurements of the total column."

    ?col <- (REQUIREMENTS::Measurement (Parameter ?p) (Vertical-Spatial-Resolution
nil)
     (Id ?id1) (taken-by ?ins1) (Accuracy ?acc1&~nil) (synergy-level# ?s1&:(< ?s1
2)))
    (REQUIREMENTS::Measurement (Parameter ?p) (Vertical-Spatial-Resolution ?vsr&~nil)
     (Id ?id2) (taken-by ?ins2) (Accuracy ?acc2) (synergy-level# ?s2&:(< ?s2 1)))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (duplicate ?col (Accuracy (eval (Improve Accuracy ?acc1))) (Id (str-cat ?id1 "-
syn-" ?id2))
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (Vertical-Spatial-Resolution ?vsr)
(synergy-level# (+ 1 (max ?s1 ?s2)))))

(defrule SYNERGIES::tropo-vs-strato-chemistry-measurements
    "If we have a chemistry measurement with high sensitivity in the troposphere,
    and another one with high sensitivity in the stratosphere, we have the complete
    atmosphere."

    (declare (no-loop TRUE) (salience -50))
    ?tro <- (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-in-low-
troposphere-PBL High) (synergy-level# ?s1&:(< ?s1 2))
    (sensitivity-in-upper-troposphere-and-stratosphere ?sr&~High) (Id ?id1) (taken-by
?ins1) (Accuracy ?acc1&~nil))
    ?str <- (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-in-low-
troposphere-PBL ?tr&~High) (synergy-level# ?s2&:(< ?s2 2))
     (sensitivity-in-upper-troposphere-and-stratosphere High) (Id ?id2) (taken-by
?ins2) (Accuracy ?acc2))

    =>

    (duplicate ?tro (Id (str-cat ?id1 "-syn-" ?id2)) (sensitivity-in-upper-
troposphere-and-stratosphere High)
```

```
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (synergy-level# (+ 1 (max ?s1
?s2))))
    (duplicate ?str (Id (str-cat ?id1 "-syn-" ?id2)) (sensitivity-in-low-troposphere-
PBL High)
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (synergy-level# (+ 1 (max ?s1
?s2)))))

(defrule SYNERGIES::sensitivity-over-oceans
    "If we have two measurements and one has good sensitivity over oceans we can
combine it with another one with lower sensitivity to create a new data product"

    ?no <- (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-over-oceans no)
    (Id ?id1) (taken-by ?ins1) (Accuracy ?acc1&~nil) (synergy-level# ?s1&:(< ?s1 3)))
    (REQUIREMENTS::Measurement (Parameter ?p) (sensitivity-over-oceans High)
     (Id ?id2) (taken-by ?ins2) (Accuracy ?acc2) (synergy-level# ?s2&:(< ?s2 1)))

    =>

    (duplicate ?no (Id (str-cat ?id1 "-syn-" ?id2)) (sensitivity-over-oceans High)
        (taken-by (str-cat ?ins1 "-syn-" ?ins2 )) (synergy-level# (+ 1 (max ?s2
?s1)))))

(defrule SYNERGIES::CO2-temperature-error
    "If a laser CO2 measurement error is accompanied by a passive temperature
measurement then the accuracy of the CO2 retrieval improves"

    ?CO2 <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter "1.8.3
CO2") (rms-system-tropoH2O# High))
    ?T <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter "1.2.1
Atmospheric temperature fields") (Spectral-region opt-SWIR))

    =>

    (modify ?CO2 (rms-system-tropoH2O# Low) (taken-by (str-cat ?ins1 "-syn-" ?ins2))
(Id (str-cat ?id1 "-syn-" ?id2 ))))

(defrule SYNERGIES::CO2-pressure-error
    "If a laser CO2 measurement error is accompanied by a passive O2 measurement
    then the accuracy of the CO2 retrieval improves"

    ?CO2 <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter "1.8.3
CO2") (rms-system-tropo-dry# High))
    ?T <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter "1.8.6
O2") (Spectral-region opt-SWIR))

    =>

    (modify ?CO2 (rms-system-tropo-dry# Low) (taken-by (str-cat ?ins1 "-syn-" ?ins2))
(Id (str-cat ?id1 "-syn-" ?id2 ))))
```

Code 115: Synergy rules from the atmospheric chemistry and physics community

The following rules concern societal applications of remote sensing data products.

```
(defrule SYNERGIES::pointing-capability
```

```
    ?m<- (REQUIREMENTS::Measurement (Parameter ?p) (Temporal-resolution ?tr1)
(Pointing-capability High) (orbit-altitude# ?h&:(> ?h 450)) (Id ?id1) (taken-by
?ins1) (synergy-level# 0))
    (or (test (eq ?tr1 Low-3days-1-week)) (test (eq ?tr1 Very-low-1-3-weeks)))

    =>

    (duplicate ?m (Temporal-resolution Medium-1day-3days) (synergy-level# 1)))

(defrule SYNERGIES::fire-monitoring-bands
    ?TIR <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter "A2.Fire
Monitoring") (Temporal-resolution ?tr1&~nil) (Spectral-sampling ?ss1&~nil)  (Accuracy
?acc&~Low) (Spectral-region opt-TIR))
    ?SWIR <- (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter
"A2.Fire Monitoring") (Temporal-resolution ?tr2&~nil) (Spectral-sampling ?ss2&~nil)
(Accuracy ?acc2&~Low) (Spectral-region opt-VNIR+SWIR))
    (test (neq ?ins1 ?ins2))

   =>

    (duplicate ?TIR (Accuracy High) (Spectral-region opt-VNIR+SWIR+TIR) (Temporal-
resolution (fuzzy-min Temporal-resolution ?tr1 ?tr2)) (Spectral-sampling (fuzzy-max
Spectral-sampling ?ss1 ?ss2))
         (taken-by (str-cat ?ins1 "-syn-" ?ins2)) (Id (str-cat ?id1 "-syn-" ?id2 ))))

(defrule SYNERGIES::hydrocarbon-reservoir-monitoring-from-surface-deformation
"Hydrocarbon reservoirs can be monitored by measuring surface deformation and surface
composition"

        ?sd <- (REQUIREMENTS::Measurement (Parameter "2.2.1 surface deformation")
(Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id1))
    (test (SameOrBetter Horizontal-Spatial-Resolution ?hsr High-10-100m))
    (REQUIREMENTS::Measurement (Parameter "2.6.5 surface composition") (Horizontal-
Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    =>

    (duplicate ?sd (Id (str-cat ?id1 "-syn-" ?id2)) (Parameter "2.6.4 hydrocarbon
reservoir monitoring")))

(defrule SYNERGIES::flood-monitoring-from-hires-topography "Hi res topography with 5m
horizontal spatial resolution and 10cm accuracy allows flood monitoring"

        ?topo <-  (REQUIREMENTS::Measurement (Parameter "2.2.2 Hi-res topography")
(Horizontal-Spatial-Resolution ?hsr & :(neq ?hsr nil)) (Id ?id1))
    (test (SameOrBetter Horizontal-Spatial-Resolution ?hsr Very-high-1-10m))

    =>

    (duplicate ?topo (Id (str-cat ?id1 "-syn")) (Parameter "2.6.4 hydrocarbon
reservoir monitoring")))
```

```
(defrule SYNERGIES::mmw-sounders-rain-rates-hurricanes
    "If there is a cloud liquid water and precipitation measurement and the
instrument has H2O bands in the MMW then it can measure rain rate, hurricanes, etc"

    ?cl<-  (REQUIREMENTS::Measurement (Parameter "1.7.1 Cloud liquid water and
precipitation rate") (taken-by ?ins))
    (CAPABILITIES::Manifested-instrument (Name ?ins) (Spectral-region MW-submm))
    (not (REQUIREMENTS::Measurement (Parameter "1.7.3 Rain rate, tropical storms, and
hurricanes") (taken-by ?ins)))

    =>

    (duplicate ?cl (Parameter "1.7.3 Rain rate, tropical storms, and hurricanes")))

(defrule SYNERGIES::fire-monitoring
    "If there is a multispectral disaster monitoring measurement, then we can do fire
monitoring"

    ?this <- (REQUIREMENTS::Measurement (Id ?id1) (taken-by ?ins1) (Parameter "2.6.3
disaster monitoring") (Spectral-sampling ?ss &~Multispectral-10-100-channels
&~Hyperspectral-100-channels-or-more))
    (REQUIREMENTS::Measurement (Id ?id2) (taken-by ?ins2) (Parameter "1.8.3 CO2")
(Spectral-sampling Multispectral-10-100-channels))

    =>

    (duplicate ?this (Parameter "A2.Fire Monitoring") (Id (str-cat ?id1 "-syn-" ?id2
)) (Accuracy Medium)
          (taken-by (str-cat ?ins1 "-syn-" ?ins2)) (Spectral-sampling Multispectral-
10-100-channels)))
```

Code 116: Synergy rules from the societal applications

The following rules concern measurements for numerical weather prediction.

```
(defrule count-num-soundings-per-day
    "Computes number of soundings per day from number of satellites carrying
    GPS receivers, based on paper Research on the Number and Distribution of GPS
Occultation Events for Orbit Selection for Global/Regional Observation, RAST 2007"

    ?m <- (REQUIREMENTS::Measurement (Parameter "1.3.3 GPS radio occultation")
(taken-by ?tk)
        (num-soundings-per-day# nil) (num-of-planes# ?np&~nil) (num-of-sats-per-
plane# ?ns&~nil) )

    =>

    (bind ?ns (* 450 (* ?ns ?np))); 450 soundings/day per satellite
    (modify ?m (num-soundings-per-day# ?ns))

    )

(defrule count-num-soundings-per-day-when-nil
    "Computes number of soundings per day from number of satellites carrying
    GPS receivers, based on paper Research on the Number and Distribution of GPS
```

```
Occultation Events for Orbit Selection for Global/Regional Observation, RAST 2007"

   ?m <-  (REQUIREMENTS::Measurement (Parameter "1.3.3 GPS radio occultation")
(taken-by ?tk) (num-soundings-per-day# nil) (num-of-planes# nil) (num-of-sats-per-
plane# nil) )

    =>

    (bind ?ns 450); 450 soundings/day per satellite
    (modify ?m (num-soundings-per-day# ?ns))
    )

(defrule num-soundings-per-day-add
    "Computes number of soundings per day from number of satellites carrying
    GPS receivers, based on paper Research on the Number and Distribution of GPS
Occultation Events for Orbit Selection for Global/Regional Observation, RAST 2007"

    ?m1 <- (REQUIREMENTS::Measurement (Parameter "1.3.3 GPS radio occultation") (num-
soundings-per-day# ?ns1&~nil) (taken-by ?tk1))
    ?m2 <- (REQUIREMENTS::Measurement (Parameter "1.3.3 GPS radio occultation") (num-
soundings-per-day# ?ns2&~nil) (taken-by ?tk2))
    (test (neq ?m1 ?m2))

    =>

    (retract ?m1)
    (modify ?m2 (num-soundings-per-day# (+ ?ns1 ?ns2)) (taken-by (str-cat ?tk1
?tk2)))
    )
```

Code 117: Synergy rules from the numerical weather prediction community

The following rules concern Earth radiation budget measurements.

```
(defrule SYNERGIES::clouds-and-radiation2
    "For EOS, it is convenient to express requirements in terms of number of CERES
flying"

    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (Name ?n1&~ACRIM) )
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (Name ?n2&~?n1&~ACRIM) )
    (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW radiance -
0.3-2um-") (Id ?id1) (taken-by ?ins1))
    ?clouds <- (REQUIREMENTS::Measurement (Parameter "1.5.3 Cloud amount/distribution
-horizontal and vertical-") (Id ?id2) (taken-by ?ins2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))

    (not (REQUIREMENTS::Measurement (Parameter "A4.Clouds and radiation") (num-of-
indep-samples# 2)))

    =>
```

```
    (duplicate ?clouds (Parameter "A4.Clouds and radiation") (num-of-indep-samples#
2)
        (Id (str-cat ?id1 "-syn-" ?id2))  (taken-by (str-cat ?ins1 "-syn-" ?ins2 ))))

(defrule SYNERGIES::clouds-and-radiation3
    "For EOS, it is convenient to express requirements in terms of number of CERES
flying"

    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (Name ?n1&~ACRIM))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (Name ?n2&~?n1&~ACRIM))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (Name ?n3&~?n2&~?n1&~ACRIM))
    (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW radiance -
0.3-2um-") (Id ?id1) (taken-by ?ins1))
    ?clouds <- (REQUIREMENTS::Measurement (Parameter "1.5.3 Cloud amount/distribution
-horizontal and vertical-") (Id ?id2) (taken-by ?ins2))
    (SYNERGIES::cross-registered (measurements $?m))
    (test (member$ ?id1 $?m))
    (test (member$ ?id2 $?m))
    (not (REQUIREMENTS::Measurement (Parameter "A4.Clouds and radiation") (num-of-
indep-samples# 3)))

    =>

    (duplicate ?clouds (Parameter "A4.Clouds and radiation") (num-of-indep-samples#
3)
        (Id (str-cat ?id1 "-syn-" ?id2))  (taken-by (str-cat ?ins1 "-syn-" ?ins2 "-
syn" ?n1 "-syn" ?n2 "-syn" ?n3)))
    )

(defrule SYNERGIES::CERES-sampling-multiple-angles-SW
    "For CERES, it is required to have one instrument in a cross-track scanning
configuration to get good spatial sampling and another one in azimuth scanning
configuration in order to  sample all possible angles"

    (declare (no-loop TRUE))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning cross-track) (Name ?n1))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning biaxial) (Name ?n2&:(neq ?n1 ?n2)))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW
radiance -0.3-2um-") (rms-variable-angular-sampling# ?ang))
    (test (eq (numberp ?ang) FALSE));; essentially nil

    =>

    (modify ?m (rms-variable-angular-sampling# 1.2)) ;; 1sigma, in W/m2 averaged over
30 days, Wielicki et al 1995
    )

(defrule SYNERGIES::CERES-sampling-multiple-angles-LW
    "For CERES, it is required to have one instrument in a cross-track scanning
configuration to get good spatial sampling and another one in azimuth scanning
```

```
configuration in order to sample all possible angles"

    (declare (no-loop TRUE))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning cross-track) (Name ?n1))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning biaxial) (Name ?n2&:(neq ?n1 ?n2)))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.2 Spectrally resolved IR
radiance -200-2000cm-1-") (rms-variable-angular-sampling# ?ang))
    (test (eq (numberp ?ang) FALSE));; essentially nil

    =>

    (modify ?m (rms-variable-angular-sampling# 1.2)) ;; 1sigma, in W/m2 averaged over
30 days, Wielicki et al 1995
    )


(defrule SYNERGIES::CERES-sampling-single-angle-SW
    "For CERES, it is required to have one instrument in a cross-track scanning
configuration  to get good spatial sampling and another one in azimuth scanning
configuration in order to  sample all possible angles"
    (declare (no-loop TRUE))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning cross-track) (Name ?n1))
    (not (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning biaxial) ))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW
radiance -0.3-2um-") (rms-variable-angular-sampling# ?ang))
    (test (eq (numberp ?ang) FALSE));; essentially nil

    =>

    (modify ?m (rms-variable-angular-sampling# 4.8)) ;; 1sigma, in W/m2 averaged over
30 days, Wielicki et al 1995
    )

(defrule SYNERGIES::CERES-sampling-single-angle-LW
    "For CERES, it is required to have one instrument in a cross-track scanning
configuration to get good spatial sampling and another one in azimuth scanning
configuration in order to  sample all possible angles"

    (declare (no-loop TRUE))
    (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning cross-track) (Name ?n1))
    (not (CAPABILITIES::Manifested-instrument (Intent "Earth radiation budget
radiometers") (scanning biaxial) ))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.2 Spectrally resolved IR
radiance -200-2000cm-1-") (rms-variable-angular-sampling# ?ang))
    (test (eq (numberp ?ang) FALSE));; essentially nil

    =>

    (modify ?m (rms-variable-angular-sampling# 4.8)) ;; 1sigma, in W/m2 averaged over
30 days, Wielicki et al 1995
```

```
    )

(defrule SYNERGIES::CERES-sampling-time-SW2
    "For CERES, it is required to have one instrument in a cross-track scanning
configuration  to get good spatial sampling and another one in azimuth scanning
configuration in order to sample all possible angles"

    (declare (no-loop TRUE) (salience -3))
    (REQUIREMENTS::Measurement (Parameter "A4.Clouds and radiation") (num-of-indep-
samples# ?n&2))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW
radiance -0.3-2um-") (rms-variable-time-sampling# ?tim))
    (test (eq (numberp ?tim) FALSE));; essentially nil

    =>

    (if (eq ?n 1) then (bind ?x 10.45) elif (eq ?n 2) then (bind ?x 2.3) elif (eq ?n
3) then (bind ?x 0.47) else (bind ?x 15.0))
    (modify ?m (rms-variable-time-sampling# ?x)) ;; 1sigma, in W/m2 averaged over 30
days, Wielicki et al 1995
    )

(defrule SYNERGIES::CERES-sampling-time-LW2
    "For CERES, error due to sampling time decreases with  the number of independent
samples as  described in Wielicki et al 1995"

    (declare (no-loop TRUE) (salience -3))
    (REQUIREMENTS::Measurement (Parameter "A4.Clouds and radiation") (num-of-indep-
samples# ?n&2))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.2 Spectrally resolved IR
radiance -200-2000cm-1-") (rms-variable-time-sampling# ?tim))
    (test (eq (numberp ?tim) FALSE));; essentially nil

    =>

    (if (eq ?n 1) then (bind ?x 10.45) elif (eq ?n 2) then (bind ?x 2.3) elif (eq ?n
3) then (bind ?x 0.47) else (bind ?x 15.0))
    (modify ?m (rms-variable-time-sampling# ?x)) ;; 1sigma, in W/m2 averaged over 30
days, Wielicki et al 1995
    )

(defrule SYNERGIES::CERES-sampling-time-SW3
    "For CERES, it is required to have one instrument in a cross-track scanning
configuration to get good spatial sampling and another one in azimuth scanning
configuration in order to sample all possible angles"

    (declare (no-loop TRUE) (salience -3))
    (REQUIREMENTS::Measurement (Parameter "A4.Clouds and radiation") (num-of-indep-
samples# ?n&3))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW
radiance -0.3-2um-") (rms-variable-time-sampling# ?tim))
    (test (eq (numberp ?tim) FALSE));; essentially nil

    =>
```

```
    (if (eq ?n 1) then (bind ?x 10.45) elif (eq ?n 2) then (bind ?x 2.3) elif (eq ?n
3) then (bind ?x 0.47) else (bind ?x 15.0))
    (modify ?m (rms-variable-time-sampling# ?x)) ;; 1sigma, in W/m2 averaged over 30
days, Wielicki et al 1995
    )

(defrule SYNERGIES::CERES-sampling-time-LW3
    "For CERES, error due to sampling time decreases with  the number of independent
samples as described in Wielicki et al 1995"

    (declare (no-loop TRUE) (salience -3))
    (REQUIREMENTS::Measurement (Parameter "A4.Clouds and radiation") (num-of-indep-
samples# ?n&3))
    ?m <- (REQUIREMENTS::Measurement (Parameter "1.9.2 Spectrally resolved IR
radiance -200-2000cm-1-") (rms-variable-time-sampling# ?tim))
    (test (eq (numberp ?tim) FALSE));; essentially nil

    =>

    (if (eq ?n 1) then (bind ?x 10.45) elif (eq ?n 2) then (bind ?x 2.3) elif (eq ?n
3) then (bind ?x 0.47) else (bind ?x 15.0))
    (modify ?m (rms-variable-time-sampling# ?x)) ;; 1sigma, in W/m2 averaged over 30
days, Wielicki et al 1995
    )

(defrule SYNERGIES::radiation-budget-err-total-SW
    "Computes total rms error for a radiation budget mission. Has 3 components,
instrument error, angular sampling error, and time sampling error. See Wielicki et al
95 about CERES for more info"

    (declare (no-loop TRUE) (salience -5))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "1.9.3 Spectrally resolved SW
radiance -0.3-2um-") (rms-variable-angular-sampling# ?ang)
         (rms-variable-time-sampling# ?tim) (rms-system-instrument# ?ins) (rms-total#
100.0))

    =>

    (if (eq ?ins nil) then (bind ?ins 0))
    (if (eq ?ang nil) then (bind ?ang 0))
    (if (eq ?tim nil) then (bind ?tim 0))
    (bind ?rms-total (sqrt (+ (** ?ang 2) (** ?ins 2) (** ?tim 2) )))
    (modify ?meas (rms-total# ?rms-total))
    )

(defrule SYNERGIES::radiation-budget-err-total-LW
    "Computes total rms error for a radiation budget mission. Has 3 components,
instrument error, angular sampling error, and time sampling error. See Wielicki et al
95 about CERES for more info"

    (declare (no-loop TRUE) (salience -5))
    ?meas <- (REQUIREMENTS::Measurement (Parameter "1.9.2 Spectrally resolved IR
radiance -200-2000cm-1-") (rms-variable-angular-sampling# ?ang)
         (rms-variable-time-sampling# ?tim) (rms-system-instrument# ?ins) (rms-total#
100.0))
```

```
    =>
     (if (eq ?ins nil) then (bind ?ins 0))
    (if (eq ?ang nil) then (bind ?ang 0))
    (if (eq ?tim nil) then (bind ?tim 0))
    (bind ?rms-total (sqrt (+ (** ?ang 2) (** ?ins 2) (** ?tim 2) )))
    (modify ?meas (rms-total# ?rms-total))
    )
```

Code 118: Synergy rules from radiation budget community

9.3 EOSS-specific knowledge

9.3.1 EOS case study: value aggregation rules

1st LEVEL OF STAKEHOLDER NEEDS DECOMPOSITION -			
Panel	Id	Description	Weight
Clouds and radiation	WAE	Cloud formation, dissip	15%
Oceans	OCE	Exchange of energy, wa	15%
Greenhouse Gases	GHG	Chemistry of the tropo	15%
Land & Ecosystems	ECO	Land hydrology and eco	15%
Glaciers and Polar Ice Shee	ICE	Glaciers, Sea Ice, and Ic	15%
Ozone and Stratospheric C	OZO	Ozone and Chemistry o	8%
Solid Earth	SOL	Volcanoes and Climate	15%
			100%

2nd LEVEL OF STAKEHOLDER NEEDS DECOMPOSITION - Clouds and radiation panel			
Objec	Id	Description	Weight
1	WAE1	Atmospheric circulation	13%
2	WAE2	Cloud radiative feedback	19%
3	WAE3	Precipitation patterns	13%
4	WAE4	Water vapor	13%
5	WAE5	Aerosols	13%
6	WAE6	Radiation budget	19%
7	WAE7	Ice and snow	6%
8	WAE8	Land Surface Water	6%
			100%

3rd LEVEL OF STAKEHOLDER NEEDS DECOMPOSITION - SUBOBJECTIVES) Clouds and radiation panel			
Objective 1	**WAE1**	**Atmospheric circulation**	
Subobjective	Id	Description	Weight
1	WAE1-1	Atmospheric temperature profiles	25%
2	WAE1-2	Atmospheric humidity profiles	25%
3	WAE1-3	Ocean surface winds	25%
4	WAE1-4	Atmospheric winds	25%
			100%

Objective 2	**WAE2**	**Cloud radiative feedback**	
Subobjective	Id	Description	Weight
1	WAE2-1	Cloud amount, cover, type	17%

2	WAE2-2	Cloud top height	17%
3	WAE2-3	Cloud top temperature	17%
4	WAE2-4	Cloud liquid water	17%
5	WAE2-5	Cloud ice particles	17%
6	WAE2-6	Cloud albedo	17%
			100%

Objective 3 **WAE3** **Precipitation patterns**

Subobjective	Id	Description	Weight
1	WAE3-1	Cloud liquid water and precipitation	34%
2	WAE3-2	Rain rates, tropical storms and hurricanes	33%
3	WAE3-3	Lightning	33%
			100%

Objective 4 **WAE4** **Water vapor**

Subobjective	Id	Description	Weight
1	WAE4-1	Atmospheric water vapor	33%
2	WAE4-2	Water vapor transport - winds	33%
3	WAE4-3	Atmospheric humidity profiles	33%
			100%

Objective 5 **WAE5** **Aerosols**

Subobjective	Id	Description	Weight
1	WAE5-1	Aerosol height and optical depth	33%
2	WAE5-2	Aerosol scattering properties	33%
3	WAE5-3	Aerosol extinction and vertical concentration profiles	33%
			100%

Objective 6 **WAE6** **Radiation budget**

Subobjective	Id	Description	Weight
1	WAE6-1	Total solar irradiance	15%
2	WAE6-2	Short-wave radiation (solar reflected)	35%
3	WAE6-3	Long-wave radiation (thermal emission)	35%
4	WAE6-4	Albedo and reflectance	15%
			100%

Objective 7 **WAE7** **Ice and snow**

Subobjective	Id	Description	Weight
1	WAE7-1	Sea ice cover	33%
2	WAE7-2	Snow cover	33%
3	WAE7-3	Snow water equivalent	33%

	100%

Objective 7	WAE8	Land Surface Water	
Subobjective	**Id**	**Description**	**Weight**
1	WAE8-1	Leaf area index	25%
2	WAE8-2	Soil moisture	50%
3	WAE8-3	Vegetation state	25%
			100%

Oceans panel

Objective 1	OCE1	Carbon Sources and Sinks	
Subobjective	**Id**	**Description**	**Weight**
1	OCE1-1	Phytoplankton	17%
2	OCE1-2	Dissolved organic carbon	17%
3	OCE1-3	Atmospheric CO2	17%
4	OCE1-4	Sea surface temperature	17%
5	OCE1-5	Atmospheric correction: Ozone and Aerosols	17%
6	OCE1-6	Atmospheric temperature	17%
			100%

Objective 2	OCE2	Ocean Circulation, Heat Storage	
Subobjective	**Id**	**Description**	**Weight**
1	OCE2-1	Sea surface wind speed	20%
2	OCE2-2	Sea surface wind direction	20%
3	OCE2-3	Sea surface temperature	20%
4	OCE2-4	Atmospheric temperature	20%
5	OCE2-5	Sea level height	20%
			100%

Objective 3	OCE3	Sea ice dynamics	
Subobjective	**Id**	**Description**	**Weight**
1	OCE3-1	Sea ice thickness	33%
2	OCE3-2	Sea ice cover	33%
3	OCE3-3	Sea level height	33%
			100%

Objective 4	OCE4	Ocean productivity	
Subobjective	**Id**	**Description**	**Weight**
1	OCE4-1	Ocean color	100%
			100%

Greenhouse Gases panel

Objective 1	GHG1	Tropospheric Ozone	
Subobjective	**Id**	**Description**	**Weight**
1	GHG1-1	Ozone profile	50%
2	GHG1-2	Ozone total column	50%
			100%

Objective 2	GHG2	Water vapor	
Subobjective	**Id**	**Description**	**Weight**
1	GHG2-1	Atmospheric humidity profiles	50%
2	GHG2-2	H2O total column	50%
			100%

Objective 3	GHG3	CO2	
Subobjective	**Id**	**Description**	**Weight**
1	GHG3-1	Atmospheric CO2 total column	100%
			100%

Objective 4	GHG4	CH4	
Subobjective	**Id**	**Description**	**Weight**
1	GHG4-1	Total column methane	100%
			100%

Objective 5	GHG5	CFCs	
Subobjective	**Id**	**Description**	**Weight**
1	GHG5-1	Total column CFCs	100%
			100%

Objective 5	GHG6	Ozone precursors - CO, Nox, CH2OH	
Subobjective	**Id**	**Description**	**Weight**
1	GHG6-1	Total column CO	50%
2	GHG6-2	Total column Nox	25%
3	GHG6-3	Total column CH2OH and non CH4 VOC	25%
			100%

Objective 5	GHG7	Hydroxyl and other radicals	
Subobjective	**Id**	**Description**	**Weight**
1	GHG7-1	Total column OH, and other Hox	100%
			100%

Objective 5	GHG8	atmospheric state: T, humidity, winds	
Subobjective	**Id**	**Description**	**Weight**
1	GHG8-1	Atmospheric temperature	50%
2	GHG8-2	Atmospheric humidity	25%
3	GHG8-3	Atmospheric winds	25%
			100%

Objective 5	GHG9	clouds	
Subobjective	**Id**	**Description**	**Weight**
1	GHG9-1	Cloud optical depth	50%
2	GHG9-2	Cloud cover	25%
3	GHG9-3	Cloud particle size distribution	25%
			100%

Objective 5	GHG10	aerosols	
Subobjective	**Id**	**Description**	**Weight**
1	GHG10-1	Black carbon and other polluting aerosols	33%
2	GHG10-2	aerosol optical depth	33%
3	GHG10-3	aerosol scattering properties	33%
			100%

Objective 5	GHG11	deforestation and biomass burning	
Subobjective	**Id**	**Description**	**Weight**
1	GHG11-1	Vegetation state	33%
2	GHG11-2	Land cover	33%
3	GHG11-3	Fire monitoring	33%
			100%

Land & Ecosystems panel

Objective 1	ECO1	Soil moisture	
Subobjective	**Id**	**Description**	**Weight**
1	ECO1-1	Soil Moisture	33%
2	ECO1-2	Water vapor	33%
3	ECO1-3	Land surface temperature	33%
			100%

Objective 2	ECO2	Land use: deforestation, biomass burning	
Subobjective	**Id**	**Description**	**Weight**
1	ECO2-1	Land cover	50%
2	ECO2-2	Fire monitoring	25%
3	ECO2-3	Vegetation state	25%

				100%

Objective 3	**ECO3**	**Atmospheric carbon**	
Subobjective	**Id**	**Description**	**Weight**
1	ECO3-1	Atmospheric total column CO2	33%
2	ECO3-2	Atmospheric total column CH4	33%
3	ECO3-3	Atmospheric total column CO	33%
			100%

Objective 4	**ECO4**	**Snow cover and water equivalent**	
Subobjective	**Id**	**Description**	**Weight**
1	ECO4-1	Snow cover	50%
2	ECO4-2	Snow water equivalent	50%
			100%

Objective 5	**ECO5**	**Vegetation state**	
Subobjective	**Id**	**Description**	**Weight**
1	ECO5-1	Vegetation state	33%
2	ECO5-2	Leaf area index	33%
3	ECO5-3	Land cover	33%
			100%

Objective 6	**ECO6**	**Water cycle**	
Subobjective	**Id**	**Description**	**Weight**
1	ECO6-1	Water vapor	33%
2	ECO6-2	Cloud liquid water and precipitation	33%
3	ECO6-3	Ocean mass distribution	33%
			100%

Objective 7	**ECO7**	**Ocean color**	
Subobjective	**Id**	**Description**	**Weight**
1	ECO7-1	Ocean color	100%
			100%

Glaciers and Polar Ice Sheets panel

Objective 1	**ICE1**	**Ice sheets thickness**	
Subobjective	**Id**	**Description**	**Weight**
1	ICE1-1	Ice sheet thickness	100%
			100%

Objective 2 **ICE2** **Sea ice cover**

Subobjective	Id	Description	Weight
1	ICE2-1	Sea ice cover	20%
2	ICE2-2	Sea ice concentration	20%
3	ICE2-3	Sea ice temperature	20%
4	ICE2-4	Albedo and reflectance	20%
5	ICE2-5	Snow depth on sea ice	20%
			100%

Objective 3 **ICE3** **Sea level height**

Subobjective	Id	Description	Weight
1	ICE3-1	Sea level height	100%
			100%

Ozone and Stratospheric Chemistry panel

Objective 1 **OZO1** **Stratospheric ozone**

Subobjective	Id	Description	Weight
1	OZO1-1	Stratospheric ozone profiles	60%
2	OZO1-2	Stratospheric ozone total column	20%
3	OZO1-3	Total UV radiation	20%
			100%

Objective 2 **OZO2** **Tropospheric ozone**

Subobjective	Id	Description	Weight
1	OZO2-1	Tropospheric ozone profiles	60%
2	OZO2-2	Tropospheric ozone total column	20%
3	OZO2-3	Total UV radiation	20%
			100%

Objective 3 **OZO3** **Stratospheric chemistry**

Subobjective	Id	Description	Weight
1	OZO3-1	Nox	20%
2	OZO3-2	Water vapor	20%
3	OZO3-3	Halogen compounds ClO, BrO	20%
4	OZO3-4	CFCs	20%
5	OZO3-5	Hox	20%
			100%

Objective 4 **OZO4** **Vulcanic SO2 and aerosols**

Subobjective	Id	Description	Weight
1	OZO4-1	Vulcanic SO2 and aerosols	100%
			100%

Objective 5	OZO5	Polar stratospheric clouds	
Subobjective	**Id**	**Description**	**Weight**
1	OZO5-1	Polar stratospheric clouds	25%
2	OZO5-2	Cloud amount	25%
3	OZO5-3	Cloud top height	25%
4	OZO5-4	Cloud ice particles	25%
			100%

Objective 5	OZO6	Stratospheric state	
Subobjective	**Id**	**Description**	**Weight**
1	OZO6-1	Atmospheric temperature	33%
2	OZO6-2	Atmospheric humidity	33%
3	OZO6-3	Atmospheric winds	33%
			100%

Solid Earth panel

Objective 1	SOL1	Vulcanic SO2, gases and aerosols	
Subobjective	**Id**	**Description**	**Weight**
1	SOL1-1	Vulcanic SO2, OCS, and other aerosols	50%
2	SOL1-2	SO2 in troposphere and stratosphere	17%
3	SOL1-3	CO2 in troposphere and stratosphere	17%
4	SOL1-4	CH4 in troposphere and stratosphere	17%
			100%

Objective 2	SOL2	Ground deformation - Tectonics	
Subobjective	**Id**	**Description**	**Weight**
1	SOL2-1	High resolution topography	50%
2	SOL2-2	Surface deformation	50%
			100%

Objective 3	SOL3	Volcanic plumes and atmospheric temperature	
Subobjective	**Id**	**Description**	**Weight**
1	SOL3-1	Thermal plumes	33%
2	SOL3-2	Visible plumes	33%
3	SOL3-3	Atmospheric temperature	33%
			33%

Objective 3	SOL4	High resolution imaging of volcanoes	
Subobjective	**Id**	**Description**	**Weight**
1	SOL4-1	Land surface (lava) temperature	100%

100%

9.3.2 EOS case study: requirement satisfaction rules

EOS requirement satisfaction rules take too much space to be printed out on paper. The complete set of requirement satisfaction rules for the EOS case study is available at: http://web.mit.edu/~dselva/www/RBES/EOS/requirement_rules.xlsx

9.3.3 EOS case study: instrument capability rules

EOS instrument capability rules take too much space to be printed out on paper. A sample is provided below. The complete set of instrument capability rules for the EOS case study is available at: http://web.mit.edu/~dselva/www/RBES/EOS/capability_rules.xlsx

9.3.4 Decadal case study: value aggregation rules

1st LEVEL OF STAKEHOLDER NEEDS DECOMPOSITION (PANELS)			
Panel	Id	Description	Weight
Weather	WEA	Weather	21%
Climate	CLI	Climate	21%
Land	ECO	Land and Ecosystems	21%
Water	WAT	Water	16%
Health	HEA	Human health	11%
Solid Earth	SOL	Solid Earth	11%
			100%

2nd LEVEL OF STAKEHOLDER NEEDS DECOMPOSITION (OBJECTIVES)			
Weather panel			
Objective	Id	Description	Weight
1	WEA1	Atmospheric winds	19%
2	WEA2	High temporal resolution air pollution	15%
3	WEA3	All-weather temperature and humidity profiles	12%
4	WEA4	Comprehensive global tropospheric aerosol characterization	8%
5	WEA5	Radio Occultation	19%
6	WEA6	Comprehensive global tropospheric O3 measurements	15%
7	WEA7	Aerosol-cloud discovery	12%
			100%

Climate panel			
Objective	Id	Description	Weight
1	CLI1	Aerosol-Cloud Forcing	24%
2	CLI2	Ice Sheet, Sea Ice Volume and Ice Dynamics	24%
3	CLI3	Carbon Sources and Sinks	24%
4	CLI4	Radiance Calibration and Time-Reference Observatory	18%
5	CLI5	Ocean Circulation, Heat Storage, and Climate Forcing	12%
			100%

Land panel			
Objective	**Id**	**Description**	**Weight**
1	ECO1	Ecosystem Function	28%
2	ECO2	Ecosystem Structure and Biomass	24%
3	ECO3	Carbon Budget	20%
4	ECO4	Coastal Ecosystem Dynamics	16%
5	ECO5	Global Ocean Productivity	12%
			100%

Water panel			
Objective	**Id**	**Description**	**Weight**
1	WAT1	Soil moisture and freeze-thaw state	29%
2	WAT2	Surface water and ocean topography	24%
3	WAT3	Snow and cold land processes	19%
4	WAT4	Water vapor transport	10%
5	WAT5	Sea Ice thickness, glacier surface elevation and glacier velocity	8%
6	WAT6	Groundwater storage, ice sheet mass balance and ocean mass	6%
7	WAT7	Inland and coastal water quality	5%
			100%

Health panel			
Objective	**Id**	**Description**	**Weight**
1	HEA1	Ozone Processes: Ultraviolet Radiation and Cancer	17%
2	HEA2	Heat Stress and Drought	17%
3	HEA3	Acute Toxic Pollution and Releases	17%
4	HEA4	Air Pollution and Respiratory and Cardiovascular Diseases	17%
5	HEA5	Algal Blooms and Waterborne Infectious Diseases	17%
6	HEA6	Vector-borne and Zoonotic Disease	17%
			100%

Solid Earth panel			
Objective	**Id**	**Description**	**Weight**
1	SOL1	Surface deformation	29%
2	SOL2	Surface composition	24%
3	SOL3	High resolution topography	19%
4	SOL4	Temporal variations in Earth gravity field	14%
5	SOL5	Oceanic bathymetry	14%
			100%

3rd LEVEL OF STAKEHOLDER NEEDS DECOMPOSITION (SUBOBJECTIVES)

Weather panel

Objective 1	WEA1	Atmospheric winds	
Subobjective	**Id**	**Description**	**Weight**
1	WEA1-1	Atmospheric wind speed	20%
2	WEA1-2	Atmospheric wind direction	20%
3	WEA1-3	Ocean surface wind speed	30%
4	WEA1-4	Ocean surface wind direction	20%
5	WEA1-5	Water vapor transport winds	10%
			100%

Objective 2	WEA2	High temporal resolution air pollution	
Subobjective	**Id**	**Description**	**Weight**
1	WEA2-1	Regional tropospheric ozone measurement	33%
2	WEA2-2	Regional tropospheric ozone precursors - CO	11%
3	WEA2-3	Regional tropospheric ozone precursors - NOx and other N compounds	11%
4	WEA2-4	Regional tropospheric ozone precursors - Formaldehyde and non-CH4 VOC	11%
5	WEA2-5	Regional tropospheric aerosols - SO2	17%
6	WEA2-6	Regional tropospheric aerosols - black carbon and other polluting aerosols	17%
			100%

Objective 3	WEA3	All-weather temperature and humidity profiles	
Subobjective	**Id**	**Description**	**Weight**
1	WEA3-1	Regional all-weather atmospheric temperature measurement	25%
2	WEA3-2	Regional all-weather atmospheric humidity measurement	25%
3	WEA3-3	Regional all-weather precipitation measurement	25%
4	WEA3-4	Regional all-weather sea surface temperature measurement	25%
			100%

Objective 4	WEA4	Comprehensive global tropospheric aerosol characterization	
Subobjective	**Id**	**Description**	**Weight**
1	WEA4-1	Global aerosol height/optical depth	20%
2	WEA4-2	Global aerosol shape and composition measurement	20%
3	WEA4-3	Global aerosol scattering properties	20%
4	WEA4-4	Global aerosol extinction profiles	20%
5	WEA4-5	Global aerosol size distribution	20%
			100%

Objective 5	WEA5	Radio Occultation	
Subobjective	**Id**	**Description**	**Weight**
1	WEA5-1	Radio occultation	100%
			100%

Objective 6	WEA6	Comprehensive global tropospheric O3 measurements	
Subobjective	**Id**	**Description**	**Weight**
1	WEA6-1	Tropospheric ozone	14%
2	WEA6-2	Global tropospheric O3 precursors: CH4	14%
3	WEA6-3	Global tropospheric O3 measurements: non-CH4 VOC	14%
4	WEA6-4	Global tropospheric O3 precursors: N2 compounds	14%
5	WEA6-5	Global tropospheric O3 precursors: CO	14%
6	WEA6-6	Global tropospheric aerosols: SO2	14%
7	WEA6-7	Global tropospheric aerosols: black carbon and other polluting aerosols	14%
			100%

Objective 7	WEA7	Aerosol-cloud discovery	
Subobjective	**Id**	**Description**	**Weight**
1	WEA7-1	Cloud type	8%
2	WEA7-2	Cloud amount and distribution	8%
3	WEA7-3	Cloud height	8%
4	WEA7-4	Cloud ice particles - size distribution	8%
5	WEA7-5	Cloud particles - phase	8%
6	WEA7-6	Cloud liquid water - precipitation	8%
7	WEA7-7	Cloud droplet size	8%
8	WEA7-8	Aerosol height	8%
9	WEA7-9	Aerosol shape and composition	8%
10	WEA7-10	Aerosol scattering properties	8%
11	WEA7-11	Aerosol extinction profiles	8%
12	WEA7-12	Aerosol size distribution	8%
			100%

9.3.5 Decadal case study: requirement satisfaction rules

Decadal requirement satisfaction rules take too much space to be printed out on paper. The complete set of requirement satisfaction rules for the Decadal case study is available at: http://web.mit.edu/~dselva/www/RBES/Decadal/requirement_rules.xlsx

9.3.6 Decadal case study: instrument capability rules

Decadal instrument capability rules take too much space to be printed out on paper. The complete set of instrument capability rules for the Decadal case study is available at: http://web.mit.edu/~dselva/www/RBES/Decadal/capability_rules.xlsx

10 References

Abshire, J. B., Smith, J. C., & Schutz, B. E. (1998). The Geoscience Laser Altimeter System (GLAS). *AIP Conference Proceedings*, *420*(1), 33-37. Aip. doi:10.1063/1.54816

Acevo, R., Aguasca, A., Valencia, E., Camps, A., Vall-llossera, M., & De, U. P. (2010). Altimetry Study Performed Using an Airborne GNSS-Reflectometer. *Proceedings of the 2010 IEEE International Geoscience and Remote Sensing Symposium* (pp. 3833-3836). Honolulu, Hawaii.

Agahi, H. H., Ball, G., & Fox, G. (2009). NICM Schedule & Cost Rules of Thumb. *AIAA Space Conference 2009* (pp. 6512-6512). Pasadena, CA.

Aliakbargolkar, A., & Crawley, E. F. (2012). Architecting Methodology for Spatially and Temporally Distributed Resource Extraction Systems. *Systems Engineering, In press*.

Aliakbargolkar, A., Wicht, A., Battat, J., Calandrelli, E., & Crawley, E. F. (2011). Heavy Lift Launch Vehicle Systems Architecting. *2011 International Astronautical Congress*. Cape Town, South Africa.

Apgar, H., Bearden, D., & Wong, R. (1999). Cost Modeling. In W. J. Larson & J. R. Wertz (Eds.), *Space Mission Analysis and Design* (3rd Edition., pp. 783-820). Microcosm.

Asrar, G., & Dozier, J. (1994). *EOS: Science Strategy for the Earth Observing System*. Woodbury, NY: American Institute of Physics Press.

Aumann, H. H., Chahine, M. T., Gautier, C., Goldberg, M. D., Kalnay, E., McMillin, L. M., Revercomb, H., et al. (2003). AIRS/AMSU/HSB on the aqua mission: design, science objectives, data products, and processing systems. *IEEE Transactions on Geoscience and Remote Sensing*, *41*(2), 253-264. doi:10.1109/TGRS.2002.808356

Aumann, H. H., & Miller, C. R. (1995). Atmospheric infrared sounder (AIRS) on the Earth Observing System. *Proceedings of SPIE* (Vol. 2583, p. 332). Retrieved from http://link.aip.org/link/?PSISDG/2583/332/1

Barnes, A. A. (1998). The Ball Commercial Platform and their applications, QuickBird, QuikSCAT and ICESAT. *IAF, International Astronautical Congress, 49th, Melbourne, Australia*.

Barnes, R. A., & Holmes, A. W. (1993). Overview of the SeaWiFS ocean sensor. *Proceedings of SPIE* (Vol. 1939, pp. 224-232). Retrieved from http://link.aip.org/link/?PSISDG/1939/224/1

Beer, R., Glavich, T. A., & Rider, D. M. (2001). Tropospheric emission spectrometer for the Earth Observing System's Aura satellite. *Applied optics*, *40*(15), 2356–2367. Optical Society of America. Retrieved from http://www.opticsinfobase.org/abstract.cfm?id=64298

Bertsekas, D. P. (1996). *Dynamic programming and optimal control*. *Control* (3rd editio., Vol. II). Athena Scientific. Retrieved from http://ocw.alfaisal.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-231Dynamic-Programming-and-Stochastic-ControlFall2002/D13859AD-A9E2-49F7-A751-15550083D667/0/DP_2ndEDITION_Corrections.pdf

Bianchi, L., Dorigo, M., Gambardella, L. M., & Gutjahr, W. J. (2008). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239-287. doi:10.1007/s11047-008-9098-4

Bianchini, G., Cortesi, U., Palchetti, L., & Pascale, E. (2004). SAFIRE-A (spectroscopy of the atmosphere by far-infrared emission-airborne): optimized instrument configuration and new assessment of improved performance. *Applied optics*, *43*(14), 2962-77. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/15143825

Blackwell, W. J., Bickmeier, L. J., Leslie, R. V., Pieper, M. L., Samra, J. E., Surussavadee, C., & Upham, C. A. (2011). Hyperspectral microwave atmospheric sounding. *IEEE Transactions on Geoscience and Remote Sensing*, *49*(1), 128–142. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5545388

Boggs, P. T., & Tolle, J. W. (1995). Sequential Quadratic Programming. *Acta Numerica*, *4*(1), 1-51.

Borradaile, G., Heeringa, B., & Wilfong, G. (2009). The Knapsack Problem with Neighbour Constraints. *Arxiv preprint arXiv09100777*, 18. Retrieved from http://arxiv.org/abs/0910.0777

Boukezzoula, R., Foulloy, L., & Galichet, S. (2011). Model Inversion using Extended Gradual Intervals Arithmetic. *IEEE Transactions on Fuzzy Systems*, *PP*(99). Retrieved from http://www.atlantis-press.com/php/download_paper.php?id=2243

Brown, O. C., Eremenko, P., & Collopy, P. D. (2009). Value-Centric Design Methodologies for Fractionated Spacecraft: Progress Summary from Phase 1 of the DARPA System F6 Program. *AIAA SPACE 2009 Conference & Exposition*. Pasadena, CA.

Brown, O., & Eremenko, P. (2008). Application of Value-Centric Design to Space Architectures : The Case of Fractionated Spacecraft. *AIAA SPACE 2008 Conference and Exposition*. San Diego,CA.

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule-based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. *Language*. Addison-Wesley.

Buckley, J. J. (1985). Ranking Alternatives Using Fuzzy Numbers. *Fuzzy Sets and Systems*, *15*, 21-31.

Butler, D. M., Hartle, R. E., Abbott, M., Ackley, S., Arvidson, R., Chase, R., Delwiche, C. C., et al. (1984). Earth Observing System Science and Mission Requirements Working Group Report, *1*.

Camps, A., Rodriguez-Alvarez, N., Bosch-Lluis, X., Marchan, J. F., Ramos-Perez, I., Segarra, M., Sagues, L., et al. (2008). PAU in SeoSAT: A proposed hybrid L-band microwave radiometer/GPS reflectometer to improve Sea Surface Salinity estimates from space. *Proceedings of the 2008 Conference on Microwave Radiometry and Remote Sensing of the Environment*. Ieee. doi:10.1109/MICRAD.2008.4579467

Capderou, M. (2005). *Satellites - Orbits and missions*. Springer Verlag.

Castro, L. D., Knowles, M., & Goodman, A. J. (2010). *Iridium Announces Comprehensive Plan For Next-Generation Constellation*.

Castro, L. D., & Rhodes, J. (2011). *First Iridium NEXT Hosted Payload Agreement Signed*. Retrieved from http://investor.iridium.com/releasedetail.cfm?releaseid=547289

Chen, W., Wiecek, M. M., & Zhang, J. (1999). Quality Utility—A Compromise Programming Approach to Robust Design. *Journal of Mechanical Design, 121*(2), 179. doi:10.1115/1.2829440

Cheney, R., Miller, L., Agreen, R., Doyle, N., & Lillibridge, J. (1994). TOPEX/POSEIDON: The 2-cm solution. *Journal of Geophysical Research, 99*(C12), 24,555-24,563.

Chu, P. C., & Beasley, J. E. (1998). Constraint Handling in Genetic Algorithms: The Set Partitioning Problem. *Journal of Heuristics, 11*, 323-357.

Clancey, William J. (1987). Knowledge-Based Tutoring: The GUIDON program. *MIT Press Series in Artificial Intelligence*. Cambridge, MA: The MIT Press.

Clancey, W.J., Letsinger, R., & Dept, S. U. C. S. (1984). NEOMYCIN: Reconfiguring a Rule-based Expert System for Application to Teaching. In W.J. Clancey & E. H. Shortliffe (Eds.), *Readings in Medical Artificial Intelligence: The First Decade*. Reading, MA: Addison-Wesley. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.7789&rep=rep1&type=pdf

Cline, D. W., Davis, R. E., & Yueh, S. H. (2005). *Cold-land Processes Pathfinder Mission Concept*.

Colson, J. M. (2008). *System Architecting of a Campaign of Earth Observing Satellites*. Massachusetts Institute of Technology.

Committee on Earth Studies, Space Studies Board, N. R. C. (2000). *The Role of Small Satellites in NASA and NOAA Earth Observation Programs*. *Earth* (p. 104).

Conrow, E. H. (2011). Technology Readiness Levels and Space Program Schedule Change. *Journal of Spacecraft and Rockets, 48*(6). doi:10.2514/1.A32020

Cost Analysis Division - NASA Headquarters. (2008). *2008 NASA Cost Estimating Handbook*. Washington DC. Retrieved from www.ceh.nasa.gov

Crawley, E., De Weck, O., Eppinger, S., Magee, C., Moses, J., Seering, W., Schindall, J., et al. (2004). *The influence of architecture in engineering systems - Engineering Systems Monograph*. *Architecture*.

Crawley, E. F. (n.d.). *ESD.34 Theory of Systems Architecture. MIT Lecture Notes.*

Dalkey, N., & Helmer, O. (1963). An Experimental Application of the Delphi Method to the Use of Experts. *Management Science, 9*(3), 458-467.

Dantzig, G., & Fulkerson, R. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society*, *1934*. Retrieved from http://www.jstor.org/stable/10.2307/166695

Das, I., & Dennis, J. E. (1998). Normal-Boundary Intersection: A New Method for Generating the Pareto Surface in Nonlinear Multicriteria Optimization Problems. *SIAM Journal on Optimization*, *8*(3). doi:10.1137/S1052623496307510

Das, N. N., Entekhabi, D., & Njoku, E. G. (2011). An Algorithm for Merging SMAP Radiometer and Radar Data for High-Resolution Soil-Moisture Retrieval. *IEEE Transactions on Geoscience and Remote Sensing*, *49*(99), 1–9. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5661825

Deb, K. (2004). A population-based algorithm-generator for real-parameter optimization. *Soft Computing*, *9*(4), 236-253. doi:10.1007/s00500-004-0377-4

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182-197. doi:10.1109/4235.996017

Dechezelles, J.-J., & Huttin, G. (2001). Proteus: A mutlimission platform for low earth orbits. *Air & Space Europe*, *2*(1), 77-81.

Dincbas, M. (1980). A knowledge-based expert system for automatic analysis and synthesis in CAD. *IFIP Congress* (pp. 705–710). Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+knowledge-based+expert+system+for+automatic+analysis+and+synthesis+in+CAD#0

Diner, D. J., Bruegge, C. J., Martonchik, J. V., Ackerman, T. P., Davies, R., Gerstl, S. a. W., Gordon, H. R., et al. (1989). MISR: A multiangle imaging spectroradiometer for geophysical and climatological research from EOS. *Geoscience and Remote Sensing, IEEE Transactions on*, *27*(2), 200–214. IEEE. doi:10.1109/36.20299

Dominguez-Garcia, A., Hanuscahk, G., Hall, S., & Crawley, E. (2007). A Comparison of GN&C Architectural Approaches for Robotic and Human-Rated Spacecraft. *AIAA Guidance, Navigation, and Control Conference and Exhibit* (pp. 20-23). Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+Comparison+of+GN&C+Architectural+Approaches+for+Robotic+and+Human-Rated+Spacecraft#0

Dorrer, M., Laborde, B., & Deschamps, P. (1991). DORIS (Doppler orbitography and radiopositioning integrated from space): System assessment results with DORIS on SPOT 2. *Acta Astronautica*, *25*(8-9), 497–504. Elsevier. Retrieved from http://www.sciencedirect.com/science/article/pii/009457659190032Z

Drexl, A. (1988). A simulated annealing approach to the multiconstraint zero-one knapsack problem. *Computing*, *40*(1), 1–8. Springer. doi:10.1063/1.3665391

Dreyfus, S. E. (1969). An Appraisal of Some Shortest-Path Algorithms. *Operations Research*, *17*(3), 395-412.

Dubayah, R., Bergen, K., Hall, F., Hurtt, G., Houghton, R., Kellndorfer, J., Lefsky, M., et al. (2008). Global Vegetation Structure from NASA's DESDynI Mission: An Overview. *AGU Fall Meeting Abstracts* (Vol. 1, p. 01). Retrieved from http://adsabs.harvard.edu/abs/2008AGUFM.B31H..01D

Dubos, G. F., Saleh, J. H., & Braun, R. (2008). Technology readiness level, schedule risk, and slippage in spacecraft design. *Journal of Spacecraft and Rockets*, *45*(4), 837-842.

Duda, R. 0., Gaschnig, J. G., & Hart, P. E. (1979). Model Design in the PROSPECTOR Consultant System for Mineral Exploration. In D. Michie (Ed.), *Expert Systems in the Microelectronic Age* (pp. 153-167). Edinburgh, Scotland: Edinburgh University Press.

Durkin, J. (1994). *Expert Systems: Design and Development*. New York: Macmillan Publishing Company. Retrieved from http://www.amazon.com/Expert-Systems-Development-John-Durkin/dp/0023309709

Edmonds, J. (1972). Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Computing*, *19*(2), 248-264.

Ehrgott, M., & Gandibleux, X. (2000). A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization. *OR Spectrum*, *22*(4), 425-460. doi:10.1007/s002910000046

Feigenbaum, E. A., Buchanan, B. G., & Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program. In B. Meltzer & D. Michie (Eds.), *Machine Intelligence 6* (pp. 165-190). Edinburgh, Scotland.

Fisher, M. L. (2004). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, *50*(12).

Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, *19*(1), 17-37. doi:10.1016/0004-3702(82)90020-0

Fortin, J., Dubois, D., & Fargier, H. (2008). Gradual Numbers and Their Application to Fuzzy Interval Analysis. *IEEE Transactions on Fuzzy Systems*, *16*(2), 388-402. doi:10.1109/TFUZZ.2006.890680

Franz, B., Werdell, P., & Meister, G. (2005). The continuity of ocean color measurements from SeaWiFS to MODIS. *Proc. SPIE, 5882*, 58820W. Retrieved from http://0-oceancolor.gsfc.nasa.gov.iii-server.ualr.edu/DOCS/Publications/5882-34.pdf

Frazier, E., & Engineer, P. S. (1998). The EOS Common Spacecraft. *Proceedings of SPIE 3439* (Vol. 3439, pp. 2-13).

Freeman, A., Donnellan, A., & Rosen, P. (2008). Deformation, ecosystem structure, and dynamics of ice (DESDynI). *(EUSAR), 2008 7th*, (3). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5757227

Friedman-Hill, E. (2003). *JESS in Action*. Manning. Retrieved from http://www.ulb.tu-darmstadt.de/tocs/184099633.pdf

Friedman-Hill, E. J. (2000). *Jess, The Java Expert System Shell*. Livermore, CA.

Fu, L. L., Christensen, E., Yamarone Jr, C., Lefebvre, M., Menard, Y., Dorrer, M., & Escudier, P. (1994). *TOPEX/POSEIDON mission overview*. Retrieved from http://trs-new.jpl.nasa.gov/dspace/handle/2014/34628

Gaier, T., Lambrigtsen, B., Lim, B., Kangaslahti, P., Tanner, A., Dwyer, I. O., Ruf, C., et al. (2010). GeoSTAR-II Technology Risk Reduction for the NASA Decadal Survey PATH Mission. *Proceedings of the 2010 NASA Earth Science Technology Conference*.

Garfinkel, R. S., & Nemhauser, G. L. (1969). The Set-Partitioning Problem: Set Covering with Equality Constraints. *Operations Research*, *17*(5), 848-856. doi:10.1287/opre.17.5.848

Giarratano, J. C., & Riley, G. D. (n.d.). *Expert Systems: Principles and Programming, Fourth Edition*. Course Technology. Retrieved from http://www.amazon.com/Expert-Systems-Principles-Programming-Fourth/dp/0534384471

Gille, J. C., Barnett, J. J., Whitney, J. G., Dials, M. A., Woodard, D., Rudolf, W. P., Lambert, A., et al. (2003). The High-Resolution Dynamics Limb Sounder (HIRDLS) experiment on AURA. *Proceedings of SPIE* (Vol. 5152, p. 162). Retrieved from http://link.aip.org/link/?PSISDG/5152/162/1

Gleason, S., Adjrad, M., & Unwin, M. (2005). Sensing Ocean, Ice and Land Reflected Signals from Space: Results from the UK-DMC GPS Reflectometry Experiment. *Proceedings of the 2005 ION GNSS Technical Meeting* (pp. 13–16). Nashville,TN. Retrieved from http://spacejournal.ohio.edu/issue9/pdf/SensingOcean.pdf

Glover, F. (1990a). Tabu Search: Part I. *ORSA Journal on Computing*, *2*(1), 4-32.

Glover, F. (1990b). Tabu search: Part II. *ORSA Journal on computing*, *2*(1), 4–32. Retrieved from http://www.iro.umontreal.ca/~dift6751/paper_glover_ts_2.pdf

Glover, F., Kelly, J. P., & Laguna, M. (1995). Genetic Algorithms and Tabu Search: Hybrids for Optimization. *Computers & Operations Research*, *22*(1), 111-134.

Goetz, a. F. H., & Herring, M. (1989). The high resolution imaging spectrometer (HIRIS) for Eos. *IEEE Transactions on Geoscience and Remote Sensing*, *27*(2), 136-144. doi:10.1109/36.20291

Graham, R. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing*, *7*(1), 43-57. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4392963

Grimaldi, R. P. (2003). *Discrete and Combinatorial Mathematics: An Applied Introduction* (Fifth Edit., p. 800). Addison Wesley Publishing Company.

Hansen, P., & Jaumard, B. (1990). Algorithms for the maximum satisfiability problem. *Computing*, *303*, 279-303.

Haskins, C. (2006). INCOSE Systems engineering handbook - A guide for system life cycle processes and activities. *Systems Engineering*.

Healey, K. J. (1986). Artificial Intelligence Research and Applications at the NASA Johnson Space Center. *AI Magazine*, *7*(3), 146-152.

Hochba, D. S. (1997). Approximation Algorithms for NP-Hard Problems. *ACM SIGACT News*, *28*(2).

Hughes Space Communications. (1990). High Resolution Microwave Spectrometer and Sounder (HIMSS) - Final Report, *1*(1). doi:10.1016/0956-7135(91)90217-K

Ignizio, J. (1983). Generalized goal programming - An overview. *Computers & Operations Research*, *I*(4). Retrieved from http://www.sciencedirect.com/science/article/pii/0305054883900035

Imhof, W., Spear, K., Hamilton, J., Higgins, B., Murphy, M., Pronko, J., Vondrak, R., et al. (1995). The polar ionospheric X-ray imaging experiment (PIXIE). *Space Science Reviews*, *71*(1), 385–408. Springer. Retrieved from http://www.springerlink.com/index/l77h7155g1631452.pdf

Jamieson, S. (2004). Likert scales: how to (ab)use them. *Medical education*, *38*(12), 1217-8. doi:10.1111/j.1365-2929.2004.02012.x

Jet Propulsion Laboratory. (2007). *Soil Moisture Active/Passive (SMAP) Mission NASA Workshop Report*.

Jin, S., & Komjathy, A. (2010). GNSS reflectometry and remote sensing: New objectives and results. *Advances in Space Research*, *46*(2), 111-117. COSPAR. doi:10.1016/j.asr.2010.01.014

Johnson, W. T. K., Rosen, P. A., Hensley, S., & Freeman, A. (2009). Radar Designs for the DESDynI Mission. *Proceedings of the 2009 IEEE Radar Conference* (pp. 2009-2011).

Keeney, R. L., & Raiffa, H. (1993). *Decisions with Multiple Objectives: Preferences and Value Trade-Offs* (p. 592). Cambridge University Press. Retrieved from http://www.amazon.com/Decisions-Multiple-Objectives-Preferences-Trade-Offs/dp/0521438837

Kelley, J. E. (1960). The Cutting-plane Method for Solving Convex Programs. *Journal of the Society for Industrial and Applied Mathematics*.

Kerr, Y. H. (1991). The Multi-frequency Imaging Microwave Radiometer: Applications To Land Surface Parameter Retrieval. *Proceedings of the 1991 International Geoscience and Remote Sensing Symposium*, *4*, 2365-2368. Ieee. doi:10.1109/IGARSS.1991.575519

King, M. D. (1990). *EOS Science Plan*.

King, M. D., & Greenstone, R. (1999). *1999 EOS Reference Handbook: A Guide to NASA's Earth Science Enterprise and the Earth Observing System*. Greenbelt, MD.

Koo, B. H. Y., Simmons, W. L., & Crawley, E. F. (2009). Algebra of Systems: A Metalanguage for Model Synthesis and Evaluation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *39*(3), 501-513. doi:10.1109/TSMCA.2009.2014546

Koo, H.-yung B. (2005). *A Meta-language for Systems Architecting*. Massachusetts Institute of Technology.

Lai, Y. J., Liu, T. Y., & Hwang, C. L. (1994). Topsis for MODM. *European Journal of Operational Research*, *76*(3), 486–500. Elsevier. Retrieved from http://www.sciencedirect.com/science/article/pii/0377221794902828

Lambrigtsen, B., Brown, S., Gaier, T., Kangaslahti, P., & Tanner, A. (2008). A Baseline for the Decadal-Survey PATH Mission. *Proceedings of the 2008 IEEE International Geoscience and Remote Sensing Symposium* (p. III - 338-III - 341). Ieee. doi:10.1109/IGARSS.2008.4779352

Lambrigtsen, B. H., & Calheiros, R. V. (2003). The humidity sounder for Brazil-An international partnership. *IEEE Transactions on Geoscience and Remote Sensing*, *41*(2), 352–361. IEEE. doi:10.1109/TGRS.2002.808304

Larson, W. J., & Wertz, J. R. (1999a). Spacecraft Subsystems. *Space Mission Analysis and Design* (3rd Editio., pp. 353-518). Microcosm.

Larson, W. J., & Wertz, J. R. (Eds.). (1999b). *Space Mission Analysis and Design*. Microcosm.

Lawler, E. L., & Wood, D. E. (1966). Branch-and-bound Methods: A Survey. *Operations Research*.

Leshner, R. B. (2007). *The evolution of the NASA Earth Observing System: A case study in policy and project formulation*. George Washington University.

Levelt, P. F., van den Oord, G. H. J., Dobber, M. R., Malkki, A., Visser, H., de Vries, J., Stammes, P., et al. (2006). The ozone monitoring instrument. *IEEE Transactions on Geoscience and Remote Sensing*, *44*(5), 1093–1101. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1624590

Levine, D. (1994). A Parallel Genetic Algorithm for the Set Partitioning Problem. *Ph.D. Thesis, Department of Computer Science, Illinois Institute of Technology*.

Lighthill, J. (1973). Artificial Intelligence: A General Survey. In B. S. R. Council (Ed.), *Artificial Intelligence: a Paper Symposium*.

Lin, M. (2010). *Multi-objective Constrained Optimization for Decision Making and Optimization for System Architectures*. *I Can*. Massachusetts Institute of Technology.

Lindberg, R. E., Lyon, K. G., Meurer, R. H., & Coxon, E. A. (1992). PegaStar spacecraft concept for remote sensing missions. *Proc. SPIE 1691, 157*. Orlando, FL.

Lindsay, R., Buchanan, B. G., & Feigenbaum, E. A. (1993). DENDRAL: A Case Study of the First Expert System for Scientific Hypothesis Formation. *Artificial Intelligence*, *61*(2), 209-261. doi:10.1016/0004-3702(93)90068-M

Long, D., Freilich, M., & Leotta, D. (1990). A scanning scatterometer for the Eos polar platform. *Proceedings of the 1990 International Geoscience and Remote Sensing Symposium*, 2447-2450. Retrieved from http://www.mers.byu.edu/long/papers/conf/IGARSS1990MayFreilich.pdf

Luo, M., Rinsland, C. P., Rodgers, C. D., Logan, J. a., Worden, H., Kulawik, S., Eldering, A., et al. (2007). Comparison of carbon monoxide measurements by TES and MOPITT: Influence of a priori data and instrument characteristics on nadir atmospheric species retrievals. *Journal of Geophysical Research*, *112*(D9), 1-13. doi:10.1029/2006JD007663

Maier, M. W., & Rechtin, E. (2000). *The Art of Systems Architecting*. New York: CRC press.

Manne, A. S. (1960). On the Job-Shop Scheduling Problem. *Operations Research*, *8*(2), 219-223.

Marsten, R. (1981). Exact solution of crew scheduling problems using the set partitioning model: Recent successful applications. *Networks*, *11*, 165-177. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/net.3230110208/abstract

Marsten, R. E. (1974). An Algorithm for Large Set Partitioning Problems. *Management Science*, *20*(5), 774-787. Retrieved from http://www.jstor.org/stable/2630089

Martin-Neira, M., Caparrini, M., Font-Rossello, J., Lannelongue, S., & Vallmitjana, C. S. (2001). The PARIS Concept: An Experimental Demonstration of Sea Surface Altimetry Using GPS Reflected Signals. *Geoscience and Remote Sensing, IEEE Transactions on*, *39*(1), 142–150. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=898676

Matossian, M. G. (1995). Earth Observing System mission design - Constrained optimization of the EOS constellation configuration design. *Proceedings of the 46th International Astronautical Congress*. Oslo, Norway.

McCormick, M. P., Chu, W. P., Zawodny, J. M., Mauldin III, L. E., & McMaster, L. R. (1991). Stratospheric aerosol and gas experiment III: aerosol and trace gas measurements for the Earth Observing System. *Proceedings of SPIE* (Vol. 1491, p. 125). Retrieved from http://link.aip.org/link/?PSISDG/1491/125/1

McDermott, J. (1982). R1: A Rule-Based Configurer of Computer Systems. *Artificial lntell., 19: 39*, *19*(1), 39-88. doi:10.1016/0004-3702(82)90021-2

Messac, a., Ismail-Yahaya, a., & Mattson, C. a. (2003). The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization*, *25*(2), 86-98. doi:10.1007/s00158-002-0276-1

Messac, A., & Ismail-Yahaya, A. (2002). Multiobjective robust design using physical programming. *Structural and Multidisciplinary Optimization*, *23*(5), 357-371. doi:10.1007/s00158-002-0196-0

Miller, B. L., & Goldberg, D. E. (1995). Tournament Selection , and the E ects of Noise Genetic Algorithms, (95006).

Minsky, M. (1974). A framework for representing knowledge. Cambridge, MA.

Monaldo, F. M., Thompson, D. R., Pichel, W. G., & Clemente-Colón, P. (2004). A systematic comparison of QuikSCAT and SAR ocean surface wind speeds. *IEEE Transactions on Geoscience and Remote Sensing*, *42*(2), 283–291. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1266716

Moore III, B., Dozier, J., Abbott, M., Butler, D., Schimel, D., & Schoeberl, M. (1991). The restructured Earth observing system: Instrument recommendations. *EOS Transactions*, *72*(46), 505–505. Retrieved from http://adsabs.harvard.edu/abs/1991EOSTr..72..505M

NASA Earth Science and Technology Office. (2009). Aerosol-Cloud-Ecosystems (ACE) Decadal Survey Mission. *2009 NASA Earth Science Technology Conference*.

NASA, & US Air Force. (2002). NAFCOM, NASA/Air Force Costing Model. *Propulsion for space transportation of the 21st century*.

NRC Committee on Earth Science and Applications from Space. (2007). *Earth Science and Applications from Space: National Imperatives for the Next Decade and Beyond*. *Earth Science*. Washington DC. Retrieved from http://www.nap.edu/catalog.php?record_id=11820

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Njoku, E., Jackson, T., & Lakshmi, V. (2003). Soil moisture retrieval from AMSR-E. *Proceedings of the 2003 International Geoscience and Remote Sensing Symposium* (Vol. 41, pp. 215-229). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1196040

Nonobe, K. (1998). A tabu search approach to the constraint satisfaction problem as a general problem solver. *European Journal of Operational Research*, *106*(2-3), 599-623. Retrieved from http://www.sciencedirect.com/science/article/pii/S0377221797002944

Nordhaus, W. (1998). What is the value of scientific knowledge? An application to global warming using the PRICE model. *The Energy Journal*, *39*(1), 65. doi:10.1016/S0140-6701(97)86051-1

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley and sons.

Rasmussen, A. L., & Sun-synchronous, L. E. O. (1998). Cost Models for Large versus Small Spacecraft. *Office*, *3439*(July).

Rayside, D., Estler, H., & Jackson, D. (2009). MIT-CSAIL-TR-2009-033: The Guided Improvement Algorithm for Combinatorial Optimization.

Reber, C., Trevathan, C., & McNeal, R. (1993). The upper atmosphere research satellite (UARS) mission. *of geophysical research*, *98*(D6), 10643-10647. doi:10.1029/92JD02828

Rechtin, E. (1991). *System Architecting, Creating & Building Complex Systems*. Englewood Cliffs, NJ: Prentice Hall.

Reeves, E. I. (1999). Spacecraft Design and Sizing. In W. J. Larson & J. R. Wertz (Eds.), *Space Mission Analysis and Design* (3rd Editio., pp. 301-352). Microcosm.

Rider, D. M., & McCleese, D. J. (1991). Stratospheric wind infrared limb sounder. *Proceedings of SPIE* (Vol. 1540, p. 142). doi:10.1049/me:19910078

Riley, G., Culbert, C., Savely, R. T., & Lopez, F. (1987). CLIPS: An expert system tool for delivery and training. *3rd Conference Artificial Intelligence for Space Applications* (pp. 53-57).

Rodriguez-Alvarez, N., Camps, A., Vall-llossera, M., Bosch-Lluis, X., Monerris, A., Ramos-Pérez, I., Valencia, E., et al. (2011). Land Geophysical Parameters Retrieval Using the Interference Pattern GNSS-R Technique. *Geoscience and Remote Sensing, IEEE Transactions on*, *49*(1), 71-84. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5475216

Rodriguez-Alvarez, N., Marchan, J., Camps, A., Valencia, E., Bosch-Lluis, X., Ramos-Pérez, I., & Nieto, J. (2008). Soil moisture retrieval using GNSS-R techniques: Measurement campaign in a wheat field. *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International* (Vol. 2, p. II–245). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4778973

Ross, G. T.; Soland, R. M. (1975). A Branch and Bound Algorithm for the Generalized Assignment Problem. *Mathematical Programming*, (8), 91-103.

Rote, G. (1992). The convergence rate of the sandwich algorithm for approximating convex functions. *Computing*, *48*(3-4), 337-361. Springer Wien. doi:10.1007/BF02238642

Rottman, G. (2000). Solar UV irradiance measurements: The UARS and EOS SOLSTICE. *Physics and Chemistry of the Earth, Part C: Solar, Terrestrial & Planetary Science*, *25*(5-6), 401-404. doi:10.1016/S1464-1917(00)00042-8

Ruf, C. S., Keihm, S. J., & Janssen, M. a. (1995). TOPEX/Poseidon Microwave Radiometer (TMR). I. Instrument description and antenna temperature calibration. *IEEE Transactions on Geoscience and Remote Sensing*, *33*(1), 125-137. doi:10.1109/36.368215

Russell, Stuart; Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.

Saaty, T. L. (2008). Decision Making With the Analytic Hierarchy Process. *International Journal of Services Sciences*, *1*(1), 83-98. doi:10.1504/IJSSCI.2008.017590

Sarsfield, L. (1998). *Cosmos on a shoestring-Small spacecraft for Earth and Space Sciences*.

Schoeberl, M. R., Douglass, A. R., Hilsenrath, E., Bhartia, P. K., Beer, R., Waters, J. W., Gunson, M. R., et al. (2006). Overview of the EOS Aura mission. *IEEE Transactions on Geoscience and Remote Sensing*, *44*(5), 1066–1074. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1624588

Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van De Velde, W., & Wielinga, B. (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. Cambridge, MA: MIT Press.

Schrijver, A. (2003). *Algorithms and Combinatorics - Polyhedra and efficiency* (p. 1800). Berlin, Germany: Springer.

Schutz, B. E., & Suite, W. B. L. (2001). Laser Altimetry and Lidar from ICESat / GLAS. *Proceedings of the 2001 International Geoscience and Remote Sensing Symposium* (pp. 1016-1019).

Scolese, C., & Bordi, F. (1990). *NASA Earth Observing System*. Washington DC.

Seher, T. (2009). *Campaign-level Science Traceability for Earth Observation System Architecting*. Massachusetts Institute of Technology. Retrieved from http://dspace.mit.edu/handle/1721.1/51639?show=full

Selva, D., & Crawley, E. F. (2010). Integrated Assessment of Packaging Architectures in Earth Observing Programs. *Proceedings of the 2011 IEEE Aerospace Conference*. Big Sky, Montana.

Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., & Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision support systems*, *33*(2), 111-126. Elsevier. doi:10.1016/S0167-9236(01)00139-7

Shishko, R. (2004). Developing analogy cost estimates for space missions. *Proceedings of the AIAA Space 2004 Conference & Exhibition*. Pasadena, CA. Retrieved from http://trs-new.jpl.nasa.gov/dspace/handle/2014/41042

Silver, M., & de Weck, O. L. (2007). Time-expanded decision networks: A framework for designing evolvable complex systems. *Systems Engineering*, *10*(2), 167-188.

Simmons, W. L. (2008). *A Framework for Decision Support in Systems Architecting*. Massachusetts Institute of Technology.

Steele, G. L. (1990). *COMMON LISP: the language* (p. 1029). Digital Press. Retrieved from http://books.google.com/books?hl=en&lr=&id=FYoOIWuoXUIC&pgis=1

Steuer, R. (1986). *Multiple Criteria Optimization: Theory, Computation, and Applications*. New York: John Wiley and sons.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343-351. doi:10.1068/b070343

Stiny, G., & Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. *Information Processing*, *71*(1460-1465).

Studer, R. (1998). Knowledge Engineering: Principles and Methods. *Data & Knowledge Engineering*, *25*(1-2), 161-197. doi:10.1016/S0169-023X(97)00056-6

Suarez, B. (2011). *Integrating Spacecraft and Aircraft in Earth Observation System Architectures*. Massachusetts Institute of Technology.

Sutherland, T. A. (2009). *Stakeholder Value Network Analysis for Space-Based Earth Observations*. Massachusetts Institute of Technology.

Suwa, M., Scott, A. C., & Shortliffe, E. H. (1984). Completeness and Consistency in Rule-based Systems. *Rule-based Expert Systems*. Reading, MA: Addison-Wesley.

Travis, L. D. (1747). Remote sensing of aerosols with the earth-observing scanning polarimeter. *Proceedings of SPIE* (Vol. 154, pp. 154-164). Retrieved from http://link.aip.org/link/?PSISDG/1747/154/1

Tsai, W.-T., Spencer, M., Wu, C., Winn, C., & Kellogg, K. (2000). SeaWinds on QuikSCAT: sensor description and mission overview. *Proceedings of the 2000 International Geoscience and Remote Sensing Symposium*, *3*, 1021-1023. Ieee. doi:10.1109/IGARSS.2000.858008

Turrini, S. (1996). *Optimization in permutation spaces*. Palo Alto,CA.

Tuyahov, A. J. (1986). The Earth Observing System for the 1990 and 2000 decades. *Science of the Total Environment*, *56*(1986), 3–15. Elsevier. Retrieved from http://www.sciencedirect.com/science/article/pii/0048969786903098

Vandenberghe, L., & Boyd, S. (1996). Semidefinite Programming. *SIAM Review*, *38*(1), 49-95.

Waters, J. W., Froidevaux, L., Harwood, R. S., Jarnot, R. F., Pickett, H. M., Read, W. G., Siegel, P. H., et al. (2006). The Earth Observing System Microwave Limb Sounder EOS (MLS) on the Aura Satellite. *IEEE Transactions on Geoscience and Remote Sensing*, *44*(5), 1075–1092. IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1624589

Way, J., & Smith, E. a. (1991). The evolution of synthetic aperture radar systems and their progression to the EOS SAR. *IEEE Transactions on Geoscience and Remote Sensing*, *29*(6), 962-985. doi:10.1109/36.101374

Weck, O. D. (2004). Adaptive weighted sum method for bi-objective optimization. *Proceedings of the 45th AIAA/ASME/ASCE/* (pp. 1-13). Retrieved from http://deweck.mit.edu/PDF_archive/3 Refereed Conference/3_28_AIAA-2004_1680.pdf

Weigel, A., & Hastings, D. (2004a). Evaluating the Cost and Risk Impacts of Launch Choices. *Journal of Spacecraft and Rockets*, *41*(1), 103-110. doi:10.2514/1.9270

Weigel, A., & Hastings, D. (2004b). Measuring the Value of Designing for Uncertain Future Downward Budget Instabilities. *Journal of Spacecraft and Rockets*, *41*(1), 111-119. doi:10.2514/1.9271

Wielicki, B. A., Barkstrom, B. R., & Harrison, E. F. (1996). Clouds and the Earths radiant energy system (CERES): An Earth observing system experiment. *Bulletin of the American Meteorological Society*, *77*(5). Retrieved from http://www.osti.gov/energycitations/product.biblio.jsp?osti_id=258894

Williams, B., & Ragno, R. (2007). Conflict-directed A and its Role in Model-based Embedded Systems. *Discrete Applied Mathematics*, *155*(12), 1562-1595. doi:10.1016/j.dam.2005.10.022

Willson, R. (2001). The ACRIMSAT/ACRIM III Experiment-Extending the Precision, Long-term Total Solar Irradiance Climate Database. *The Earth Observer*, *13*(3), 14–17. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+ACRIMSAT/ACRIM+III+Experiment-Extending+the+Precision,+Long-term+Total+Solar+Irradiance+Climate+Database#0

Wolpert, D. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE*, 1-32. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=585893

Woods, T. N. (2000). Overview of the EOS SORCE mission. *Proceedings of SPIE*, *4135*, 192-203. Spie. doi:10.1117/12.494229

Woods, W. A. (1973). Progress in natural language understanding — An application to lunar geology. *Proceedings of the national computer conference and exposition*.

Xu, R., & Wunsch II, D. (2005). Survey of Clustering Algorithms. *IEEE transactions on Neural Networks*, *16*(3), 645-78. doi:10.1109/TNN.2005.845141

Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decision making. *Systems , Man and Cybernetics, IEEE Transactions on*, (1), 183-190. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=87068

Yamaguchi, Y., Kahle, A. B., Tsu, H., Kawakami, T., & Pniel, M. (1998). Overview of Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER). *IEEE Transactions on Geoscience and Remote Sensing*, *36*(4), 1062-1071. doi:10.1109/36.700991

Ye, L. R., & Johnson, P. E. (1995). The impact of explanation facilities on user acceptance of expert systems advice. *Management Information Systems Quarterly*, *19*(2), 157-172.

Zadeh, L. (1963). Optimality and Non-Scalar-Valued Performance Criteria. *IEEE Transactions on Automatic Control*, *59*.

Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, *8*(3), 338-353. Prentice Hall PTR New Jersey. doi:10.1016/0165-0114(78)90029-5

Zimmermann, H.-J. (1987). *Fuzzy sets, decision making, and expert systems*. Boston, MA: Kluwer Academic Publishers.

Zwicky, F. (1969). *Discovery, Invention, Research through the morphological approach*. Macmillan Publishing Company.

de Weck, O. L., & Jones, M. B. (2006). Isoperformance: Analysis and design of complex systems with desired outcomes. *Systems Engineering*, *9*(1), 45-61. doi:10.1002/sys.20043