

Semantic Federation of Product Information from Structured and Unstructured Sources

Matthias Wauer
TU Dresden
Computer Networks Group
01062 Dresden
Germany
matthias.wauer@tu-dresden.de

Johannes Meinecke
SAP Research Dresden
Chemnitzer Str. 48
01187 Dresden, Germany
johannes.meinecke@sap.com

Daniel Schuster
TU Dresden
Computer Networks Group
01062 Dresden
Germany
daniel.schuster@tu-dresden.de

Andreas Konzag
BMW Group
Munich
Germany
andreas.konzag@bmw.de

Markus Aleksy
ABB Corporate Research
Ladenburg
Germany
markus.aleksy@de.abb.com

Till Riedel
TecO/Karlsruhe Institute of
Technology
Karlsruhe
Germany
riedel@teco.edu

ABSTRACT

Product-related information can be found in various data sources and formats across the product lifecycle. Effectively exploiting this information requires the federation of these sources, the extraction of implicit information, and the efficient access to this comprehensive knowledge base. Existing solutions for product information management (PIM) are usually restricted to structured information, but most of the business-critical information resides in unstructured documents. We present a generic architecture for federating heterogeneous information from various sources, including the Internet of Things, and argue how this process benefits from using semantic representations. A reference implementation tailor-made to business users is explained and evaluated. We also discuss several issues we experienced that we believe to be valuable for researchers and implementers of semantic information systems, as well as the information retrieval community.

Categories and Subject Descriptors

H.3.4 [Information Storage and Retrieval]: Systems and Software; H.4.0 [Information Systems Applications]: General

General Terms

Design, Management, Experimentation

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 2011 ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

Keywords

Federated Information System, Product Information Management, Ontology, Semantic Web, Information Extraction

1. INTRODUCTION

Product-related information is generated, accessed and manipulated along the product lifecycle in heterogeneous formats. Only part of this information can be accessed using state-of-the-art product information systems as large parts of this information are only available in unstructured sources or distributed along different databases and legacy systems. The challenge to create an all-embracing view on products is huge. Such a comprehensive product information system has to integrate and harmonize data from all phases of the product lifecycle, all different source formats like unstructured documents, sensor information or product databases. Furthermore, it must even cross organization boundaries as different stakeholders may be responsible for the design, production, delivery, and service of a product.

The Aletheia project [1] is a unique attempt to bring together industry partners (ABB, BMW, Deutsche Post DHL, Otto, SAP) with five innovative application scenarios from different phases of the product lifecycle and five different landscapes of current state-of-the-art product information management. All these partners have a keen interest in improving the information flow internally as well as with their customers and partners and to open up new sources of product-related information like Web 2.0 pages.

In this paper we try to answer the research question if it is possible to federate structured as well as unstructured sources of product information along the product lifecycle. We use semantic technologies for this purpose and deploy and advance information extraction techniques. The scenario in Section 2 describe two of the use cases of the Aletheia project clarifying the opportunities of federated product information systems (FPIS). We further discuss requirements derived from these and other scenarios. A discussion of existing architectures for semantic information management and federation in Section 3 shows the need for

a new architecture matching the requirements mentioned in Section 2.3. The contributions of this paper consist of

1. A discussion of **design decisions for FPIS** in Section 4,
2. A high-level **component architecture for FPIS** in Section 5.1 including a concept for data sharing between organizations,
3. A detailed **concept of the Aletheia Service Hub** (Section 5.2), our central component for information federation within organizations,
4. A **reference implementation** of a semantic FPIS described in Section 6.

Section 7 contains a discussion of the results achieved so far.

2. SCENARIOS AND REQUIREMENTS

In order to motivate our research, we discuss two scenarios in the industrial sector. They are derived from two case studies conducted in the Aletheia project, focusing on

1. product lifecycle management (PLM) at ABB, a large company providing power and automation products, technology, and service¹, and
2. knowledge management in automotive engineering at the BMW company.

2.1 Use Case ABB

The customer has installed several products of the company at their local site. A service team of the customer notices that one of the devices is defective. Even though they have the knowledge which devices are applied at this installation, they lack the capability of identifying the actual cause and repairing the device. Hence, they contact the company's call center that records the service request. However, neither the customer's service team nor the call center associate have expert knowledge about the defective device. The service report therefore is frequently inaccurate, and preparing the service operation is laborious for an assigned service technician on the basis of this report.

On site, most of the suitable unstructured information is not consulted by the service technician because finding it based on the available information is cumbersome. If additional spare parts are required to repair the device, the service technician has to manually coordinate the order of a spare part and its delivery with the company's call center, the logistics provider, and the customer's service team. This causes several phone calls and requires much effort because the service technician's available information is not integrated with those of the other parties.

This use case can be optimized by three means. First, the customer should be able to solve well known problems with defined solutions without the need to consult a company's service technician. Aletheia can support this by providing such information related to the customer's actual installed

¹In addition to this company, the scenario includes a customer from the chemical sector that uses the company's products and services, and a logistics provider that stores and ships the company's spare parts. A similar use case is studied in more detail in [14].

base and corresponding historical information. On top of that, the different vocabulary of customers and service documents can be translated with semantic search. Second, this previously collected information is useful to more accurately define detailed problem descriptions for identifying the most appropriate service engineer, who can find related information from unstructured documents easier if they are extracted and semantically related to the respective device and problem symptoms. Third, the federation of RFID and sensor data correlated to the defective device can improve failure analysis by providing all relevant information, while the connected information helps assisting processes like ordering spare parts between the involved parties.

2.2 Use Case BMW

The development of vehicles is characterized by high complexity, a strong network of development issues, and many participants who work in a matrix organization. The product development process can be divided into the phases of concept and series development. In the concept phase, technical requirements are derived by customer requirements. It is characterized by dynamic processes and few formalisms. In addition, a large amount of unstructured data is needed for the main part, called the concept phase.

The use case identifies relevant requirements of the technology and methods development. For this purpose, the research group has examined key elements of the product development applied in a business process with its data, actors, and interaction patterns (Figure 1).

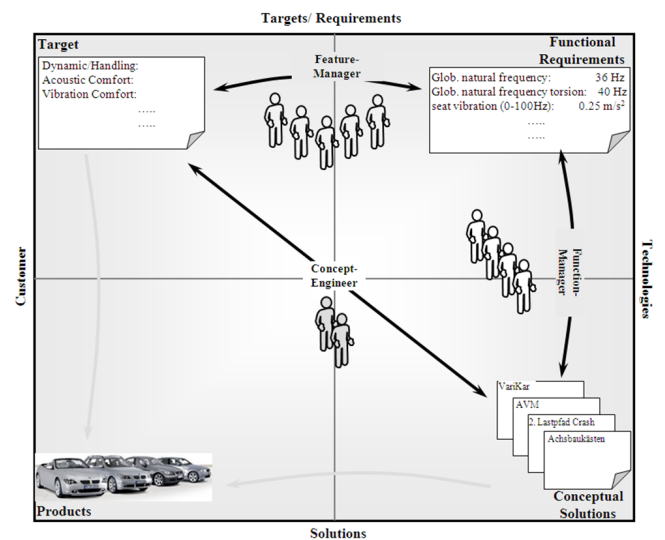


Figure 1: Development of vehicle concepts

This process describes the derivation of functional requirements of customer-related product goals by the feature managers and provision of design solutions by the function managers. The concept engineer is responsible for ensuring consistency and plausibility, which are ranked above all relevant development issues in the project. As illustrated in Figure 2 a major part of the relevant data for the concept phase is unstructured. Structured data are created during the concept phase of the development process.

Conventional methods and tools for data management, as they are productively employed in the series development

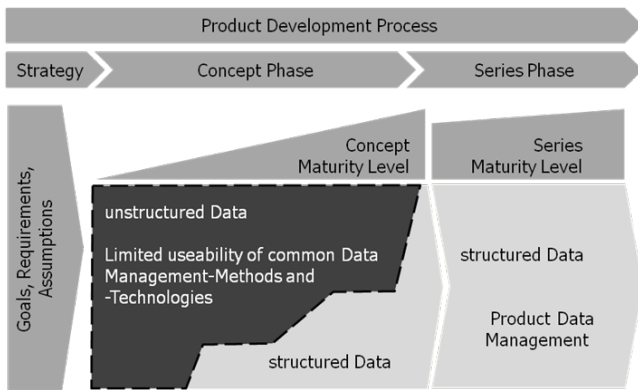


Figure 2: Relevance of managing unstructured data in the concept phase of vehicle development

of automobiles, are limited in their applicability due to the characteristics of the vehicle concept development.

The challenges which have to be mastered for the Aletheia use case can be roughly divided into two classes:

1. *Federation of different sources:* The data and information required for early product development are stored in different information silos. Not only structured data have to be considered. Especially during concept development, mostly unstructured data is produced, with office documents being common examples. Furthermore, these data are not accessible corresponding to its content or context, and the relations to structured data are not covered or described inadequately.
2. *User access to the federated information:* From a user perspective, the physical location of data and information is secondary. The information itself is more important. Between the various development issues there are several dependencies which are not well-known by all parties. It is not just about finding individual documents or facts. In addition, the dependencies between different information fragments from different sources should be useable and visible for users.

Conventional systems focus either on the search for structured data (like database knowledge discovery systems), or on the search of unstructured data (such as desktop search). Therefore, at the technical level, mechanisms must be provided to enable uniform access to data and documents from different physical sources. From a user perspective, methods and suitable surfaces are necessary to allow an integrated view of the federated informations and the dependencies between them.

2.3 Requirements

Out of these and other scenarios from the use case partners we identified a large number of requirements. The requirements were acquired on-site in 2-day workshops at each industrial partner. The resulting large set of requirements then has been further analysed and condensed to the following main categories:

Requirement 1: Federated Information Retrieval

The central functionality users want to use an FPIS for is federated information retrieval, i.e., formulating an information need and retrieving relevant results from the FPIS.

There are many features similar to classical search engines like natural language queries, auto-complete, clustering of results, personalization, and faceted search that users expect from an FPIS. Beyond that, we also identified FPIS specific requirements: search results that mix up document links and ontology facts relevant to the search query. A sort of federated ranking method is needed to bring order to this result list. Furthermore, one should be able to restrict the sources to search for by a query, or some kind of intelligent source selection method should identify the best sources for each query.

Requirement 2: Information Exploration

Besides the retrieval part, users should also be able to navigate through the information space created by an FPIS and to explore connections between different information entities, documents, and related concepts. Information exploration and information retrieval can also be mixed up as exploration may be refined by a query and vice versa.

Requirement 3: Information Integration

Federation of product information means integrating existing sources of information that were formerly used separately to create an all-embracing view on all product-related information. Thus a large number of requirements targets on using existing databases and make them searchable within the FPIS. Other types of information sources include file shares with formatted documents such as Word, PowerPoint or PDF and information from the Internet of Things, i.e., RFID and sensor data. Information integration includes appropriate mapping schemas, the management of access policies for the different sources as well as the actual access technologies like Web service interfaces or the like.

Requirement 4: Information Extraction

Full-text indexing of unstructured information (i.e., documents on file shares as well as websites) is not enough to reach the goal of semantic federation of product information along the product lifecycle. Information extraction techniques are needed to obtain information from unstructured documents. This mainly means (but is not restricted to) Named Entity Recognition (NER) to recognize entities with different keywords but belonging to the same semantic concept. Meta information should also be extracted from the documents to improve the relevance assessment.

Requirement 5: Information and Ontology Management

Once an FPIS gets deployed we also need means to directly manipulate the information presented to the user. It might be incorrect or important information may be missing. This may optionally require update mechanisms to populate changes made in the FPIS back to the information sources. Another important point is the ability to easily manipulate the ontologies used to realize the information integration and information extraction.

Requirement 6: Information Sharing

Interestingly, the aspect of information sharing between organizations did not play an important role in the interviews. Only in the ABB case we actually found a use case where we need sharing of information as partner companies do part of the service for machines on behalf of ABB. But if FPIS will get used in organizations, the need for information sharing will soon arise and will be the next step in the evolution of FPIS. If we really want to create an all-embracing view covering all phases of a product's lifecycle, we need to share at least part of the information between a product's

designer, producer, retailer, logistics provider, and service provider.

3. RELATED WORK

Since the suggested federated product information system incorporates a multitude of functionality, related research spans a number of areas. Thus, this section will discuss federated information systems in general before moving on to systems that introduce semantic processing. After a survey of frameworks for information extraction the application of the extracted information for semantic search will be discussed. Finally, we present related work with a focus on product information.

3.1 Federated Information Systems

An early approach to federated search was presented as the Information Manifold [13]. The system uses source descriptions, describing contents and capabilities of different structured information sources, in order to determine appropriate execution plans for a query. In contrast to Aletheia, unstructured information is only integrated by applying manually defined “topics” to such information sources. Furthermore, the approach assumes a global schema, referred to as world view, to federate the information. Aletheia should instead provide the means to integrate information based on different models, as stated in requirement 3. Nevertheless, Information Manifold is an important guideline as it applies description logic based knowledge representations for its federation algorithms.

3.2 Semantic Federation Systems

The complex nature of the presented requirements led to the investigation of Semantic Web technologies in order to handle the complex task of relating the federated information. In this context, the NeOn project provides a generic architecture [29] for ontology-driven applications. It separates the required services for ontology engineering and ontology usage with a clear focus on the engineering part. The aspects that are most important regarding Aletheia, such as the interaction of the core services and the extraction of information from the data sources, is not covered in detail. Instead, the creation and maintenance of semantic information is the key aspect of NeOn. With its focus on the usage of federated information, Aletheia instead needs to define components and interfaces that not only handle semantic information, but also integrate them with uncertain information that has been extracted from unstructured sources. This also requires the definition of appropriate services that enable access to this comprehensive knowledge base.

Considering distributed semantic information, projects like SemaPlorer [23] have shown the benefits of federating such data sources. It proposes the use of NetworkedGraphs, providing distributed views over RDF datasets that can be queried using SPARQL. As shown in Figure 3, it presents scalable reasoning by preprocessing a transitive closure of the SKOS hierarchies of configured datasets. Again, it only partly supports the requirements of Aletheia, as it neither connects arbitrary sources, nor is there any distinction between public and confidential information.

Related to that, but based on a different motivation of the social semantic desktop, the NEPOMUK project [19] developed a semantic personal information search system and different ontologies suitable for ordinary desktop enti-

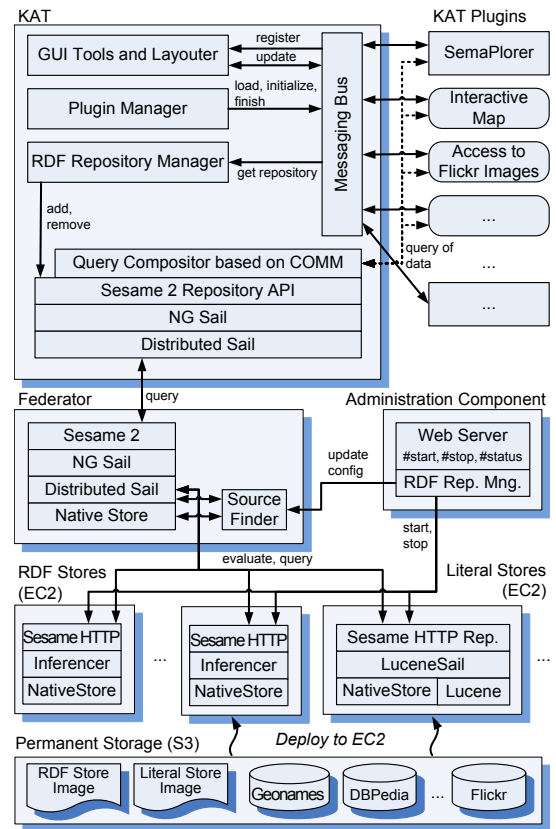


Figure 3: SemaPlorer architecture, from [23]

ties. As such, it also addresses the extraction of semantics from documents and how they can be indexed and stored, but apparently there are no means to handle databases. It proposes a P2P architecture for distributed storage of documents, but this overlay network does not ensure that all documents are accessible. Thus, it does not address the issue of actually connecting truly heterogeneous information in our context, but the document extraction components are highly valuable for Aletheia.

Several other architectures have been proposed that deal with semantics. The Knowledge Content Carrier Architecture KCCA [4] focuses on semantically describing documents, in this case paid content, and leveraging them using a proposed architecture. However, most of the work is about the definition of a schema, and the upper ontologies that provide its foundation, for describing facets such as content, presentation, a usage context called community, and business descriptions like negotiation protocol and pricing scheme. The only hints on the actual federation is given by a high-level overview of components like Registry and Manager, and a stateful protocol which is said to be based on serialized RDF graphs.

3.3 Information Extraction Related Research

Regarding unstructured information, SMILA (Semantic Information Logistics Architecture) [24] presents a simple data extraction model for different unstructured sources and an architecture based on OSGi, SCA, and BPEL. This allows for dynamically switching extraction components and flexible management of the execution depending on specific

use cases. Compared to the Aletheia requirements, it does not connect this data with structured information, and any semantic processing is designed to be executed on a higher level. Indeed, the ontology store proposed by the architecture is rather a wildcard for further extensions.

With regards to the extraction of information, the Unstructured Information Management Architecture (UIMA) [10] can act as a blueprint for the extraction components of Aletheia, separating the information access, analysis, and acquisition aspects. Some design decisions of UIMA, such as the annotation of metadata as simple sets of key-value pairs, may have to be revisited in order to gain major benefit of this data for the integration process, as explained in requirement 3.

Gaining precise knowledge from different sources is the objective of YAGO [12], which relies on few core sources that are assumed to provide correct information and semantically connects this information. Core extractors use rules to derive the knowledge base. The extracted facts are further restricted to those validated using the WordNet taxonomy. In a second two-step process, additional information is gathered from Web resources that is then judged with regards to the existing knowledge base. Although we generally follow a similar approach, the presented Aletheia use case would not benefit from publicly available but irrelevant core sources like Wikipedia, as intended by YAGO.

3.4 Semantic Information Retrieval

One major motivation of federating heterogeneous data sources can be found in the potential increase of relevance of the found documents, given that at the moment, no information extraction process can faithfully extract completely reasonable knowledge from unstructured data. Providing humans with more accurate search results that they can utilize to gain that knowledge therefore is more promising. The matching of a semantic representation of documents with a query model based on the same semantic representation is likely to achieve that.

An early study showing that potential was performed by Paralic and Kostial [17] who compared traditional full-text TF-IDF information retrieval and latent semantic indexing (LSI) with a similarity metric that compares the respective sets of concepts for query and documents:

$$sim_{onto}(\vec{Q}, \vec{D}_i) = \begin{cases} |Q_{con} \cup D_{i,con}| & \text{if } |Q_{con} \cup D_{i,con}| \neq 0 \\ k & \end{cases}$$

where Q_{con} and D_{con} are sets of concepts assigned to query \vec{Q} and document \vec{D}_i , respectively. According to the authors, results can be improved when this similarity metric is multiplied with the TF-IDF similarity. On a MEDLINE related corpus with manually assigned query concepts, this knowledge based approach showed a significantly improved performance, as shown in 4.

In reality, users prefer to use free text input methods to formulate a query for their information need, in contrast to ontology terms that would be required for the above method. Hence, user input needs to be translated into such semantic concepts. Tran et al. [28] studied how such an interpretation can be executed using a graph-based approach. Users can provide keywords that are matched with a set of concepts. Assuming that a subgraph of the concepts for the set of

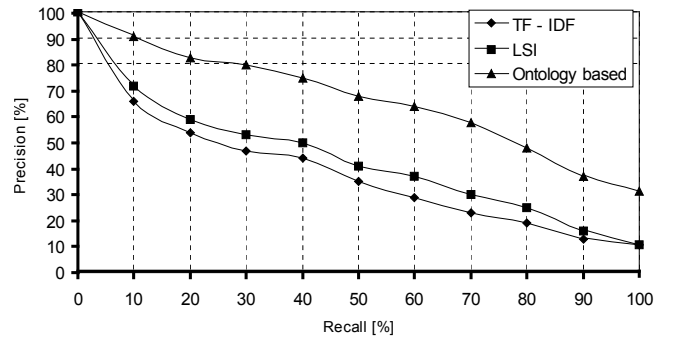


Figure 4: Performance comparison of TF-IDF full text indexing, latent semantic indexing, and ontology-based similarity metric, from [17]

keywords is more relevant if the concepts are closely related, the system ranks each subgraph accordingly, based on the number of edges in that graph.

This approach struggles with mapping terms to concepts when the terms aren't defined in the ontology. As a result, the average recall is relatively low (43-52%), whereas the F-measure in their evaluation averages between 64% for the generated query selected by hand, and 53% for the automatically selected highest-ranked query. If appropriate lexical knowledge is available, this could be utilized to improve the performance.

In order to circumvent the problem of vocabulary coverage, the K-search system [5] proposes an improved *hybrid search* method that applies semantic search for query parts where metadata is available, and traditional keyword-based search for parts that are not covered. Originally conceived in [21], which basically re-ranks the original search results returned by a syntactic search, K-search extends this combined approach with a method for directly addressing concepts and relations in the semantic model. Thus, a query is parsed into different parts for keyword, semantic, and keyword-in-context queries that are each processed by the respective search engines. Finally, the individual results are merged and ranked using the *provenance* information of each semantic assertion, i.e. the document each triple has been extracted from. Note that this provenance requires all semantic information (metadata) to arise from documents, which is not a valid assumption within Aletheia.

Evaluation of this hybrid search system showed that the accuracy is much better and combines high recall measures better than keyword search with the improved precision of semantic search, without the drawbacks of either method. In fact, the F-measure improved by 49% and 55% with regards to keyword and ontology based search, respectively. Additionally, a user-based evaluation was carried out with service engineers and designers. Here, the service engineers favored hybrid search by 61%, whereas designers did not have a clear preference. For Aletheia, this is of interest with regards to one of the use cases with a similar user group. As a detail, the user interface of K-search is form based, because the intended audience is familiar with it.

3.5 Product Information Related Research

Focusing on product information, Brunner et al. [8] examine the use of semantic technologies in the context of Master

Data Management (MDM). For their system named SOR, they argue that a subset of OWL DL is sufficient for most product information management scenarios. Furthermore, they present a generic meta-model for defining scenario-specific product information as well as a basic architecture for processing such product information. Although it shows how product information can be management semantically, it does not explain how to keep the complexity of the ontology from the user. The integration of existing data sets is not discussed either.

The Product Semantic Representation Language PSRL [18] instead focuses on the exchangeability of product information, and how they can be formalized in order to be interpreted by the involved parties. Similar to the paper mentioned above, they use W3C standards and a description logics representation. However the work mainly consists of a semantic mapping methodology between assumed ontologies for collaborating parties, for which the paper notes “no application has an explicit ontology”. Thus, they simply suppose that the involved applications already use the proposed model for their terminologies.

Other product information management architectures appear to ignore Semantic Web standards, e.g. OpenPDM [26]. They implement a product data sharing architecture on top of the ISO STEP standard data model and existing SOA middleware. Although this approach helps to reduce the cost of developing and maintaining connectors and data formats, we believe that the underlying data model enables an MDA-based code generation, but fails to support a more thorough processing and integration of product information, such as extensible reasoning.

All of these approaches do not seem to incorporate any mechanisms for extracting information from unstructured documents.

3.6 Conclusions

All of the presented research only solves part of the requirements. Hence, the architecture proposed subsequently aims to provide an integrated approach for a FPIS. Table 1 compares the discussed related work with regards to fulfilling the requirements towards a federated product information system.

4. DESIGN DECISIONS

Based on the requirements mentioned above, there are several design alternatives that are not straightforward to decide when designing an FPIS. We discuss the main design decisions we faced in the Aletheia project in the following.

4.1 Data Sources and Processing

The nature of federated information retrieval, as described in *requirement 1*, implies a range of data sources that may provide relevant information for a user. In most cases these data sources are beyond the control of the Aletheia system. We therefore decided to design a layer of data providers, components which are physically separated from the actual data processing and management. Furthermore, the flexibility of implementing the data-providing components using the technology of choice is valuable. That way, data sources can be made available that would otherwise not be accessible, or would require an extensive re-implementation of access logic.

These data-providing components can be further distinguished by their access characteristics. The content of some data sources often changes. Ideally these data sources should be queried on demand in order to ensure current data, such as the status of a system or the current location of a parcel. However, response times can be longer due to the increased communication overhead and the performance of the slowest data source. In some cases, it is not practical to *pull* all information because it may be required for, e.g., input suggestions (auto-complete).

On the other hand, much of the content in data sources can be considered static, and translating the source format into a processable form is costly. For such data sources, *push* access, i.e. loading the data into a logically centralized storage, is more appropriate. However, modifications of the original data sources need to be propagated, and updated pushed information may be inconsistent with previously stored information from this or a different source. We therefore explicitly decided to support both access paradigms and let FPIS system architects decide which method is most appropriate for each data source.

Push access frequently occurs for data sources like Web and file share documents. This access paradigm also makes for a more thorough processing of the source documents, such as semantic annotation, which would take too much time during query processing. It is possible to use a *pipes and filters* architectural style to connect the individual processing steps.

However, we found that separating document discovery, extraction, and storage from the semantic annotation tasks yield two major advantages. The implementation of the document preprocessing, such as crawling and full text extraction, can evolve independently from annotation processing like named entity recognition, given a defined data model. Additionally, documents that have already been crawled do not have to be crawled again if the annotation process is modified or restarted on a different knowledge base. The negligible drawback of this approach is a potential scalability limitation if the intermediate document storage is too slow or too small.

4.2 Storage of Structured and Unstructured Information

During the requirement analysis, we found the need for a strict separation of facts, such as data extracted from a curated database, and possibly erroneous information resulting from, e.g., automated document annotation. The latter kind of information always contains a level of uncertainty which, in some cases, can be quantified by the respective algorithm, and is linked to mostly unstructured data.

Furthermore, we noticed that semantic middleware is not equally appropriate for these two kinds of information that must be clearly distinguished. Some systems provide semantic views on a large number of instances and reasoning on assertions. Others are more suitable for document storage and attaching probabilities to annotations, which is necessary for reasonable ranking algorithms. In most cases, algorithms differ for fact and data search, e.g. in terms of navigation facets and ranking methods.

As a result, we decided to separate the management and processing of structured information from managing unstructured information, similar to [30]. This repository split increases the flexibility of modeling the ontologies, e.g., regard-

<i>Related Work</i>	<i>Requirements</i>					
	Federated	Exploration	Data Integration	Extraction	Management	Sharing
IM [13]	x	o	structured, unstructured	-	-	-
NeOn [29]	o	x	semantic	-	x	o
SemaPlorer [23]	x	x	semantic	-	-	o
NEPOMUK [19]	o (P2P)	x	semantic, unstructured	x	x	x
KCCA [4]	o	-	unstructured	-	o	o
SMILA [24]	-	-	unstructured	o	-	-
UIMA [10]	-	-	unstructured	x	-	-
YAGO [12]	-	o	semantic, structured, unstructured	x	x	-
PaKo [17], Tran [28], K-Search [5]	-	-	semantic, unstructured	-	-	-
SOR [8]	o	-	semantic	-	o	-
PSRL [18]	-	-	semantic	-	o	x
OpenPDM [26]	o	-	structured	-	o	x

x supported o partially supported - not supported

Table 1: Comparison of Related Work w.r.t. Aletheia requirements

ing the definition of certain concepts as instances or classes. Information can still be semantically connected between the two repository components by using the same URI. Reasoning about both repositories, however, is not possible, but it would not be sensible with regards to the nature of stored information. As a side effect, certain functionality that requires data from both repositories requires additional programming.

With this decision in mind, the implementation of core services accessing the repository is tightly coupled with the underlying semantic technology. An abstraction on that layer would require unnecessary effort and hinder performance optimizations, which regularly require direct access to the platform. As a drawback, switching the semantic middleware to another API would result in re-implementing most of these basic services.

Although the federation of structured data sources is a core aspect of the federated product information system, we found that our semantic middleware of choice already supports extension points for most of this functionality. The use of another semantic middleware may require additional design and implementation effort.

4.3 Front-end Interactions

In general, front-ends including additional back-end support are separated from the core services. Thus, any required technology can be applied, e.g. Google Web Toolkit (GWT) or Adobe Flex. Initial tests showed that communication overhead that may lead to a perceived increased response time is only marginal.

Additionally, the interfaces between the front-end and core services are independent of specific semantic technologies. Here, we favored reusability and additional control over extended functionalities that may result in, e.g., directly exposing a SPARQL interface.

Finally, benchmarks [15] and tests showed that certain expansive semantic queries may result in increased response times. Even though we selected a semantic middleware that performs very well, we decided to employ asynchronous communication between the front-end and Aletheia services, typically based on AJAX [11] Web interfaces, in order to improve the system’s user-perceived responsiveness. As a re-

sult, the implementation of the front-end tends to be more complex, depending on the applied technology.

5. REFERENCE ARCHITECTURE

Based on the design decisions described above, we developed a service-oriented reference architecture consisting of distributed components for information access and extraction, information federation and information presentation. All these components communicate via Web service interfaces thus separating these different concerns and ensuring a maximum flexibility and extensibility. Our reference architecture consists of five major entities that are described in the following. The central component connecting each of these entities is then explained in detail in Section 5.2.

5.1 Components and Information Flow

Figure 5 illustrates the architecture of an Aletheia system. With regards to requirement 6 (information sharing), it includes the connection of instances across different organizations and departments. It is comprised of the following major components:

Application Servers enable clients to access the Aletheia system and map domain-specific functionality to the generic interfaces. This includes a user-centered preparation of the available information. Both stand-alone and Web applications are supported as front ends.

Aletheia Service Hubs (ASH) are the key component for managing the semantic model, indexed documents, and stored facts and metadata that are stored inside the repository component. They offer Web Service interfaces for application servers and access information sources using Web Service interfaces again. They can be connected to each other across organizational boundaries, with distributed query processing as in [30].

Repository A repository is attached to each ASH to store or cache the primary data retrieved from the information sources, additional meta data about this information and a semantic model to harmonize all the information from the different sources.

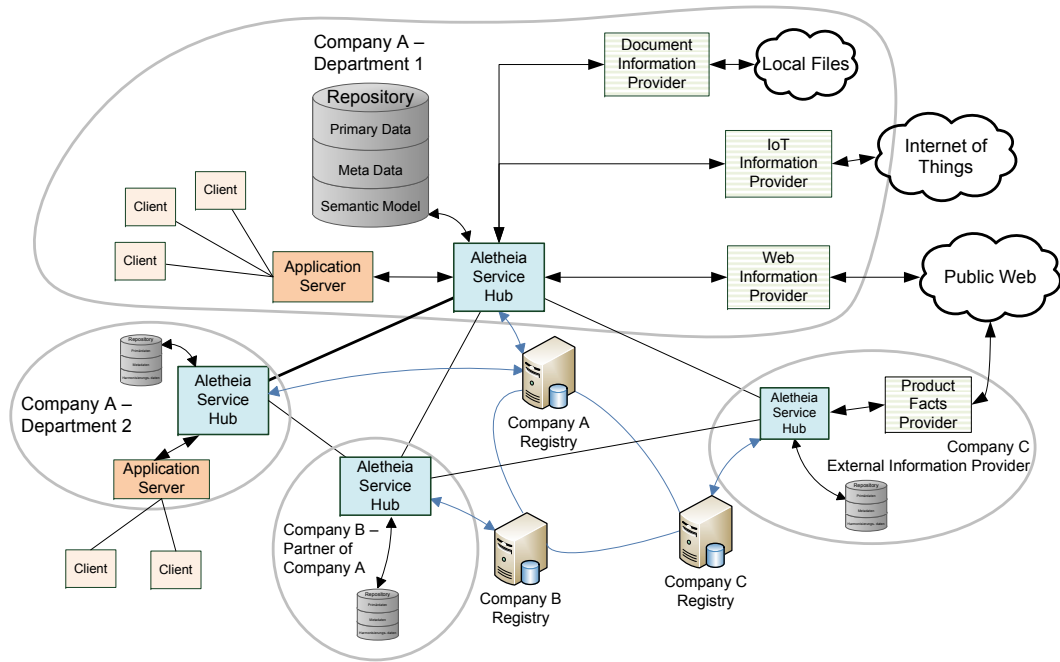


Figure 5: Aletheia reference architecture

Information Providers act as wrappers to existing data and information sources that should be leveraged by the Aletheia system, and facilitate both push and pull access depending in the type of source. They are the components that actually access the data sources.

Registries store information about each ASH (external registry interface) and the connected information sources and services (internal registry interface), enabling discovery for more or less close cooperations between different departments and organisations.

With regard to distribution, each of the Aletheia Service Hubs is the central node of a generally closed system that can be connected to external parties due to defined terms and conditions. This decision was an implication to the data sovereignty required by all the industry partners. Nevertheless, the platform may be configured to provide publicly available information via various channels, particularly with regards to linked data [6].

As the ASH and its connected repository are the central entities of our architecture, their specific composition is explained in detail subsequently.

5.2 Aletheia Service Hub

A more detailed view on the Aletheia Service Hub's components, corresponding to the architecture, can be seen in Figure 6, a fundamental modeling concepts (FMC) block diagram. Here, the client components are shown at the top, whereas the data sources appear at the bottom of the architecture. This detailed architecture can be separated into different layers.

A few vertical services such as generic configuration and monitoring features can be used by all components. Please note that many connections have been left out in order to improve readability.

Front End Services.

These Web services are supposed to abstract the user queries from the technical implementation of the repository, hence reducing the complexity of the system from the client's point of view. The major task of finding federated semantic information is supported by the Semantic Search and Navigation Service providing faceted search for extracted and stored knowledge as well as documents corresponding to this knowledge. The semantic integration of these different types of information into one condensed result list is one of the major achievements of Aletheia. Besides the search functionality, a user can also browse the virtual catalog of product information using this service.

The result lists can be downloaded for offline processing using the Export Service if needed. The Update Service provides the ability to modify the information presented in the result lists thus enabling a feedback channel for the users of the system.

User customization is realized by the Login Service, setting up user sessions and assigning roles to these users as well as managing the user profile (context, preferences, history) to improve the quality of search results by personalization. The Configuration Service complements the frontend services by providing a tool for administrators to manage and configure the components of the Aletheia Service Hub.

Repository.

Due to the different types of stored information, we conceived a combination of different repository components. The general and domain ontologies, stored facts, mappings and rules as well as the reasoner component are considered part of the semantic repository. An additional text index enables search of ontology concepts and facts. This part of the repository is capable of managing different modules, which can be exploited for storing the individual ontologies

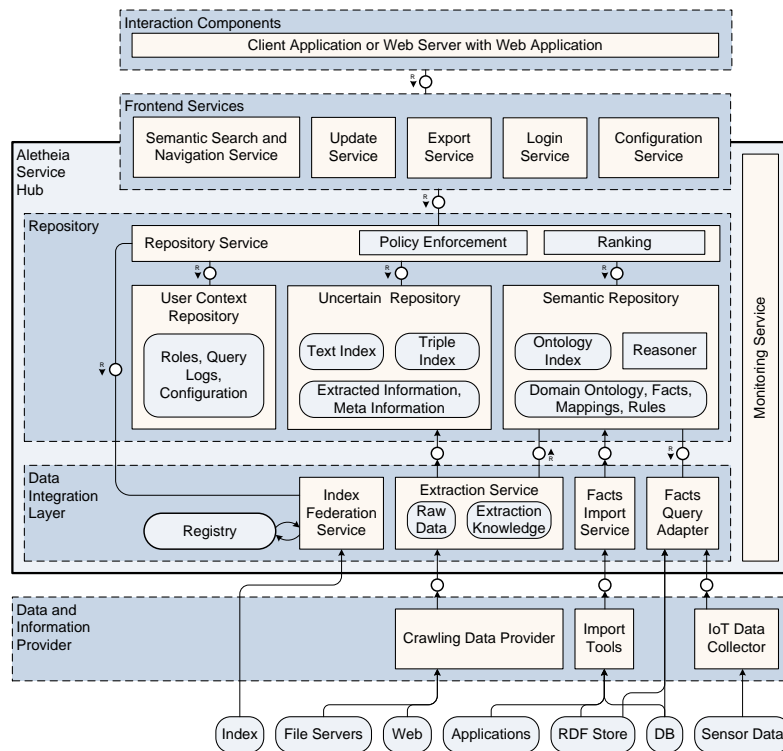


Figure 6: Architecture of the Aletheia Service Hub

of the participants for a certain use case.

In addition to that, the uncertain information repository manages knowledge that has been automatically extracted using approaches such as natural language processing (NLP), i.e., it has a certain probability and cannot be safely trusted like facts. The uncertain repository stores the extracted information and assigned meta information (source, time of extraction, trust value, ...) as RDF triples. A triple index enables search on this information. Additionally, the documents are also indexed in a classic text index so information can be searched in two ways either syntactically or semantically.

The third sub-repository stores the user context, i.e., the roles, query logs, preferences and other configuration information. This information is exploited by the policy enforcement point as well as the ranking component to improve ranking by personalization.

The three different repositories are tied together to a virtual repository by the Repository Service providing a single query interface. Queries are then transmitted to the different repositories to fetch information already inside the repository as well as to federation components like the Index Federation Service and the Facts Query Adapter relaying the queries to external information providers.

Data Integration Layer.

The actual federation of information is done in the Data Integration Layer supporting both push and pull semantics. Facts from structured information sources can be integrated in the semantic repository using the Facts Import Service. Unstructured documents can be delivered to the Extraction Service which executes information extraction processes and

pushes the extracted information as well as the raw text of the document to the Uncertain Repository.

Pull semantics for structured information is supported by the Facts Query Adapter, a plugin in the Semantic Repository which forwards structured queries to external sources. It provides a mapping of Aletheia queries to SPARQL and SQL thus enabling any SPARQL endpoint or SQL database to be integrated in the Aletheia system. Nevertheless, a schema mapping to the respective domain ontology has to be created for each structured data source regardless if it is integrated on push or pull basis.

Pull integration of unstructured information is done by the Index Federation Service. It uses a registry of external indexes to be queried at runtime. Such indexes could be existing search engines or special Aletheia information providers capable to do Named Entity Recognition. We are currently investigating resource selection methods for this data integration path to be able to selectively query external indexes depending on query topics.

Data and Information Provider.

The last layer of our architecture comprises the Data and Information Providers which are residing at the information source to pre-process information for the Alethea system. The Crawling Data Provider is a Web and file share crawler delivering raw document data to the Extraction Service in a unified format. Several Import Tools are necessary to get product information out of existing enterprise applications or RDF stores and databases. These tools export the information to XML which is then further processed and semantically lifted by the Facts Import Service. Thus, RDF stores and databases can be accessed in both modes - push

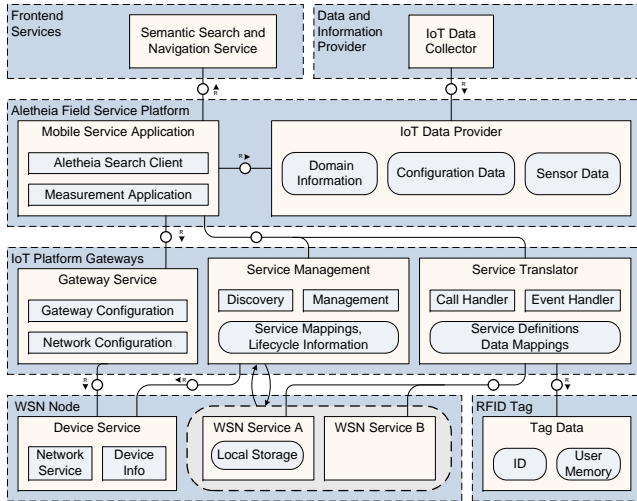


Figure 7: Integration of the Internet of Things via gateways, and interaction components towards Frontend and Data and Information Provider

and pull.

5.3 Internet of Things Integration

A special challenge for a future FPIS is the integration of the so called Internet of Things. Wireless Sensor Network (WSN) technology and Radio Frequency Identification extends information systems towards real world objects enabling fine granular updates of the product state. RFID technology in particular can play an important role for product life cycle management [25]. WSNs can provide important maintenance critical information, especially for products and parts that otherwise could not be accurately captured by information technology [27].

Technology like WSNs and RFID can both link information to the physical reality and provide fine-granular and up-to date information about the state of physical entities and make this information accessible towards an FPIS. Especially, as figure 7 illustrates, the Aletheia architecture can integrate Internet of Things technologies in two different ways: at the frontend and at the backend.

We use RFID technology as frontend extension for Information discovery. E.g., in the ABB use case we focus on identifying machinery and parts by linking physical items to named entities via a machine readable nameplate. This enables physical interaction with both real world systems and the FPIS connecting physical products and their digital counterpart. Therefore, the physical objects play in important role in practical product information management and retrieval. In the architecture depicted in figure 7 this is reflected by the fact that the mobile application queries both the real world via the IoT Platform Gateways as well as its virtual representation via the Semantic Search and Navigation Services.

The second aspect of Internet of Things integration comprises the federation aspect of our FPIS architecture. As already highlighted in figure 5, the Internet of Things is a distinguished federation source for the Aletheia system. However, at the lowest layer of the architecture we have to deal with (structured, non-human readable) binary data

that is stored or communicated via an arbitrary radio interface. The IoT platform gateways technically abstract a very heterogeneous IoT hardware landscape that can be found practically in the field.

Building upon the standard Devices Profile for Web Services (DPWS), we can technically and semantically uniquely identify measurement devices as named entities via an URI. This URI can be resolved locally via an infrastructureless discovery mechanism to a Web Service communication endpoint. We use automatic translation mechanisms to generate platform specific (Web) Service Translators [20] towards the devices that dynamically translate any communication to schema-based XML messages. At the level of the IoT Platform Gateways we can thus provide structured, well-typed and self-descriptive Web Services interfaces towards the Measurement Application.

The WSDL interfaces already contain important syntactical information like data types, minimum and maximum, or numerical precision of a value but are also used to semantically enrich the information by adding, e.g., information about quality of the sensor. Such annotation information can be provided as unstructured text, or as Semantic Annotations for WSDL and XML Schema, or RDFa. The important aspect of this gateway architecture is that beyond we can provide technology independent, homogeneous access to heterogeneous IoT sources for identification and measurement.

The major challenge when federating real world sensor information is including necessary context information necessary for an interpretation of such data. Typically, this interpretation is strongly dependent and interlinked with the real world processes. In the ABB use case, this is the service and diagnosis tasks performed by a field service engineer on site. One example is the mapping of sensor readings to locations and products, which may change over the temporal domain. The measurement application informs the IoT Data Provider of such changes. E.g., when an engineer places a sensor or tracking device on a part for diagnosis or monitoring the part on site or during transportation, the semantic association between the named product entity and the IoT device has to be made explicit. Further, when a node is re-configured or calibrated, the configuration data are also associated with the consecutive sensor readings, together with additional annotations and data such as images provided by the service worker on site. All data relations are captured in the underlying work flow of the ABB measurement application and are serialized as XML, based on a domain specific schema.

IoT Data Provider interfaces the Data Integration Layer of the architecture as a structured document-based (XML) data source. For the ABB use case we have decided not to embed raw sensor data into the semantic repositories. The actual raw sensor data is not embedded, but linked via additional serialized documents. We chose this interface separation at the current point, because the sensors make up a significant amount of data that can rarely be stored efficiently or semantically queried in the proposed repository. Instead, they are transferred to file servers.

The platform still obtains all the necessary metadata available that contains part associations, placement, calibration and configuration information, but can also feature statistical information about the readings such minima, maxima, or median values aggregated over time or discrete critical

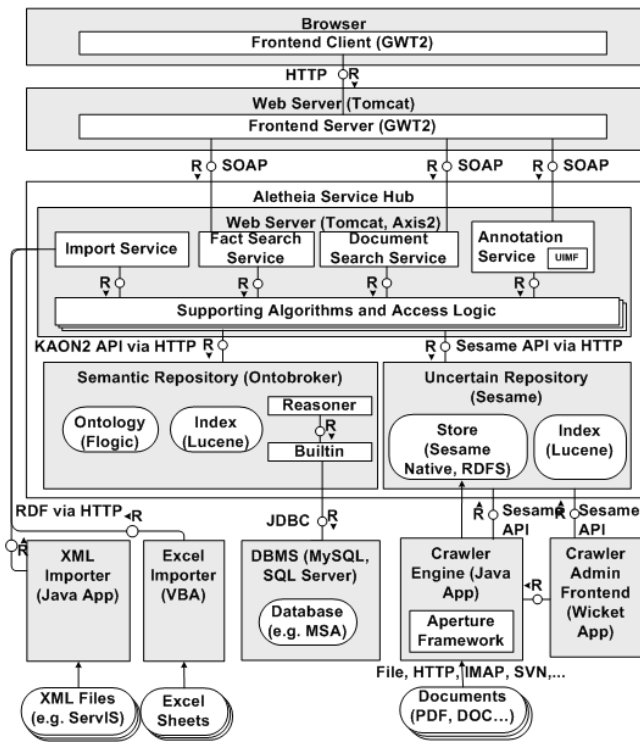


Figure 8: Overview of implemented components

events detected by the IoT devices. By connecting the individual measurements with machine product information they can be discovered and accessed later on, e.g., in specific analyzing applications via the FPIS.

6. PROTOTYPICAL REALIZATION

The presented architecture has been implemented with the most important components as a fully-working prototype within the Aletheia project and applied to real application partner data. In this section, we give an overview of the technical realization and used components, present the different semantic-aware user interface modes for accessing information, explain the activities that stand behind the information integration in the Service Hub, briefly describe the realization of data source lifting mechanisms and give an end-to-end example that shows the interaction of the technical components.

6.1 Overview of Technical Realization

Figure 8 gives an overview of the realized components and the technologies used for them. This implementation architecture can be seen as one possible instance of the reference architecture presented in section 5.

The front end of the Aletheia prototype has been realized as an AJAX Web application based on Google Web Toolkit (GWT2). This asynchronous user interface technology was chosen to account for the need for sending multiple complex semantic queries to the Service Hub. Hence, partial results can be presented to the user before all queries complete, greatly improving the user experience.

The Aletheia Service Hub currently encompasses 4 major services for importing facts into the repository, searching for structured information, searching semantically indexed

documents and annotating text. As syntactic and uncertain repositories, we have integrated an Ontobroker [2] and a Sesame store [7]. With these choices, we take advantage of the Ontobroker support for querying large numbers of instances over live data sources (e.g. relational databases over JDBC) and of the interoperability of Sesame with the Aperture extraction framework [3] applied inside the Web crawling data provider (crawler engine). The semantic document annotation service is implemented based on the UIMA framework. Other facts providers have been integrated via the import service’s very lightweight interface that can easily be accessed from many different platforms, including from Microsoft Excel macros.

In order to support the different domain models of individual use case participants, we utilize the Ontobroker capability of managing several ontologies in terms of multiple modules and respective namespaces. For example, the prototype can be switched from an ABB centric ontology to one that has been developed at BMW, simply by an appropriate selection in the front end, as shown in Figure 9.

6.2 User Interaction Paradigms

The Aletheia Frontend supports user to find the product-related information they need with appropriate user interfaces. Depending on the situation of the user, different interaction paradigms are suitable. We implemented several paradigms that leverage the knowledge of how the objects of interest are related to each other (via the links in the Semantic and Uncertain Repository). The focus was on realizing generic mechanisms that can generate the user interface completely from the domain models, without any domain-specific programming. Furthermore, the goal was to support complex structured queries (to take advantage of the semantic relationships), while not forcing the user to think in terms of complex queries (to account for the analyzed non-IT user needs in the PLM domain).

6.2.1 Search

Search in this context means that users enter text in a search box which is interpreted as a structured query, using the ontology of the Semantic Repository as background knowledge. The user is supported to disambiguate the query so that he can find precisely what he meant. In the prototype, this takes the form of auto-complete suggestions. The system can support homonyms (disambiguating different things with the same name) and synonyms (finding the same thing via different names). The search text can be interpreted as a structured query by taking into account semantic relationships (e.g. “Mueller service job reports”: finding all reports on jobs performed by a service engineer Mueller). The system can explain to the user why a result was found with natural text or graphically.

6.2.2 Navigation

The most straight-forward way of leveraging the semantic relationships in the Semantic Repository is to offer users links to related information in the user interface. For example, when information about a machine aggregated from multiple databases is displayed, interesting links point to information on spare parts for the machine, to a list of documents that mention the machine and to other machines that are also mentioned in the documents.

6.2.3 Graph Exploration

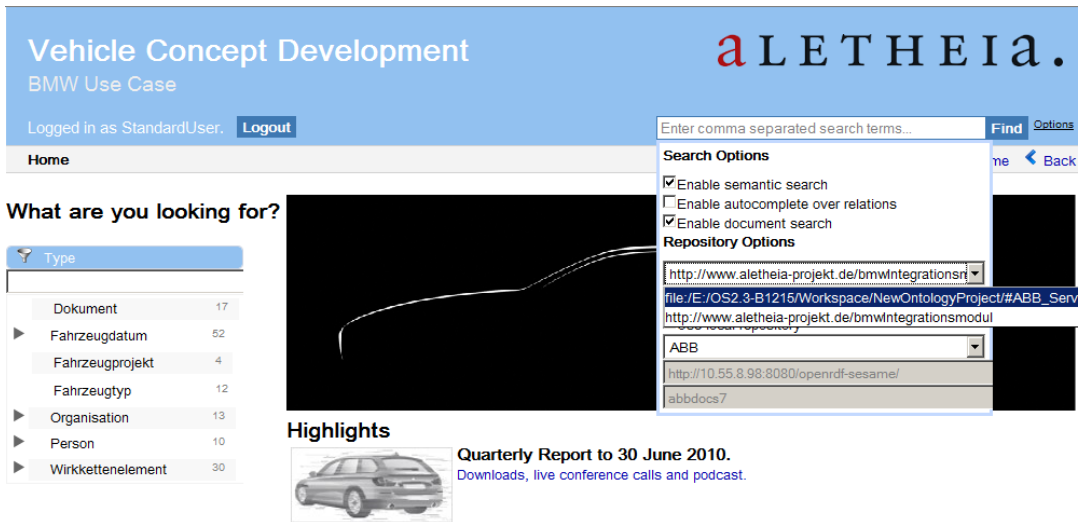


Figure 9: Selection of different ontology configurations by users in the prototype

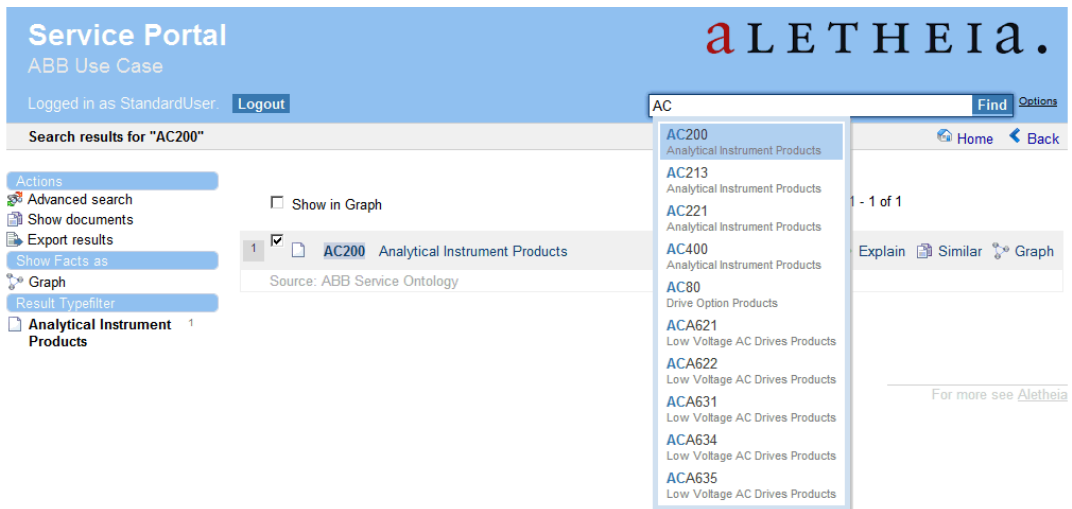


Figure 10: Search example

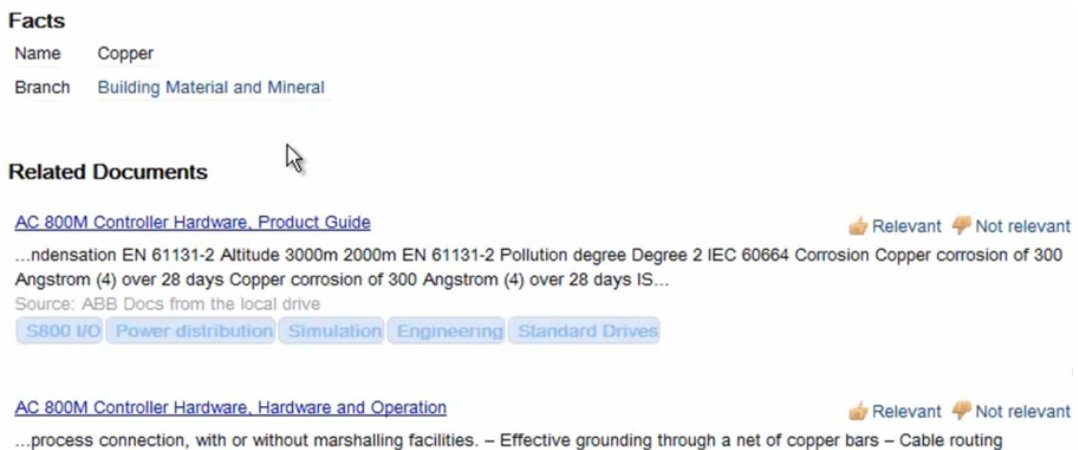


Figure 11: Navigation example

Figure 13: Filtering example

In some cases, it makes sense to partially display the information graph from the Semantic Repository itself to the user and let him navigate and explore the nodes by expanding neighboring nodes and links. This allows the user to understand complex dependencies that are otherwise hard to see (e.g. “Who worked together in which service jobs?”).

6.2.4 Filtering

Another paradigm we realized for finding information is to narrow down a list to the entries of interest by applying multiple filters (e.g. from a list of engineers to only those engineers who have worked at a specific site; or: from a list of engineers to the list of sites at which these engineers have worked).

6.3 Information Integration Activities

A core function of the Aletheia system realized in the Service Hub is the semantic integration that makes the data from the different sources correspond to one unified human-understandable model. We describe a number of activities that are necessary to establish integration with the implemented Service Hub in the following sub sections.

6.3.1 Domain Modeling

The first and foremost pre-requisite for setting up the Service Hub is that someone has modeled the domain of interest by creating an ontology. The ontology defines, among other things, classes (e.g. “company”, “product”), relationships (e.g. “sells”) and attributes (e.g. “company name”). The modeling is performed by domain experts that have been trained or are supported by ontology engineering experts. It is created based on a formal language (in our case: F-Logic) with the help of an ontology tool (in our case Ontostudio).

The modeling task must necessarily be manual and cannot be automated, because the model is supposed to reflect how humans understand the domain and how they want to view and search for the data. In practice, the ontology often already exists partially in other forms (e.g. an Excel sheet with product categories) that can be imported into

the ontology.

6.3.2 Source Mapping

Structured data residing in different data sources have different schemas (e.g. different database schemas and XML schemas). When the data sources are lifted to graph representations, they usually adhere to different technical ontologies that have been automatically generated from the data source schemas. In order to resolve the semantic gap between the different ontologies and make the data sources queryable according to the central domain ontology, we defined mappings. We created mapping rules with the help of tools, which were then executed by the semantic middleware automatically (e.g. when a query is posted for instances of the domain ontology class “Company”).

The mapping cannot be fully automated, because only humans can say with certainty what the meaning of the class and attribute names are in the different schemas. The mapping process can however be supported semi-automatically with algorithms that detect similarities in the schemas to be mapped.

Furthermore, we were faced with XML schemas that contain implicit references. These kind of internal links between nodes are important in order to model relationships, but current tools do not recognize them. Thus, we provided a semi-automatic mapping tool that extends JXML2OWL [22] with additional attribute to node mapping features. That way, an attribute or text node *id* of a XML element can be marked as a reference to another node which contains an attribute with the same content as *id*. This solution provides an increased flexibility for schemas that don’t adhere to the XLink [9] specification.

6.3.3 Link Generation

One important aspect of the semantic integration is the computation of the semantic links that capture how the objects of interest are related to each other. In some cases, we derived these with certainty from the data sources, e.g. from foreign-key-relationships in a database schema or the respective technical ontology. Here, we manually defined rules for detecting these relationships and making them explicit.

In other cases, if there was no certain rule, the links had to be computed by comparing individual instances and their attribute values. Here, we developed a tool that proposes links based on similarity measures and having a human confirm or alter these links.

6.3.4 Text Analysis

In order to fill the Uncertain Repository with information on unstructured data, text has to be analyzed so that explicit links can be drawn between terms occurring in the text and the objects in structured sources (e.g. linking the mentioning of a company name in a bill to the customer entry in a CRM system). In the project, named entity recognition was used to identify entities (i.e. concepts and instance data) in text by transforming available structured background knowledge into dictionaries valuable for text based search. Typically, each entity is associated to different human readable natural language representations (labels), which occur in text corpora. Each such representation is treated as a cue for the according instance. Each cue has a certain associated information content which is based on the frequency of that term in structured data. An

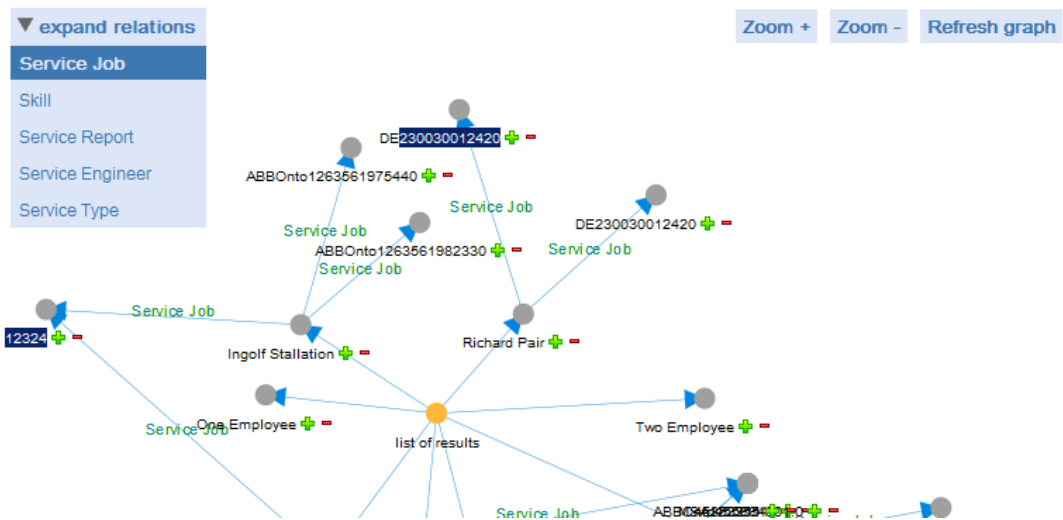


Figure 12: Graph exploration example

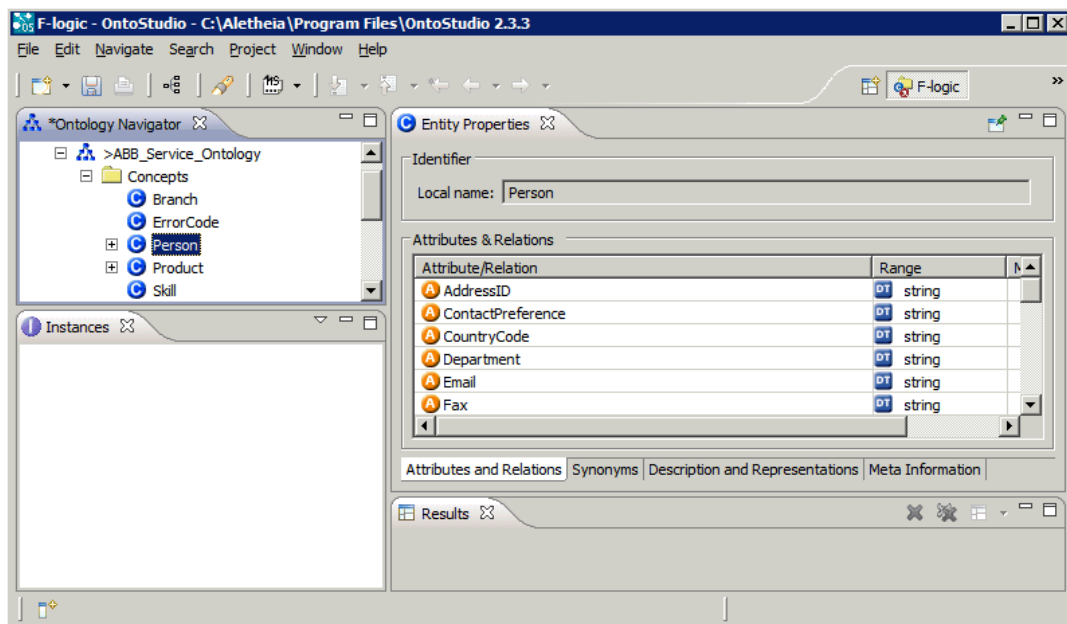


Figure 14: A domain ontology modeled in a tool

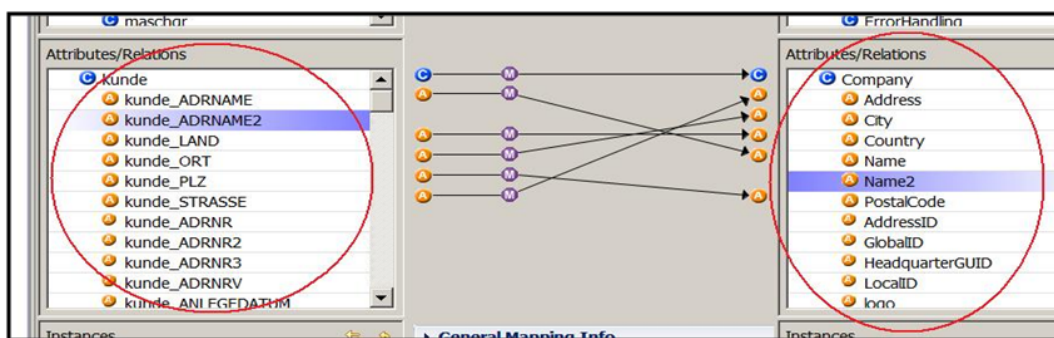


Figure 15: Example of mapping between technical data source ontology (left) and central domain ontology (right)

example for a created dictionary looks like that:

```
<token canonical="http://.../a1/bmw#Thorax">  
  <variant base="beifahrerairbag" i="4.515037"/>  
  <variant base="Thorax" i="5.767800"/>  
  <variant base="thorax" i="4.615120"/>  
  <variant base="gurtsystem" i="4.301462"/>  
  <variant base="fahrerairbag" i="4.515037"/>  
</token>
```

Documents are tokenized and annotated using the cue-based dictionary. Tokens and entity annotations are persisted in separate query-optimized indices containing additional meta-data about the annotation process. In addition global document corpora statistics (mean document length, document count) needed for ranking are calculated. Having this information, different ranking schemes based are applied on the information content, TF-IDF and the distance of terms and entities to each other within one document.

The creation of dictionaries from the structured data and the text analysis are fully automated processes. Human manual effort might have to be necessary when a new document corpus has to be integrated that uses language that is not well covered by the ontology / the structured data. In this case, the ontology might have to be extended with new labels.

6.4 Data Source Lifting

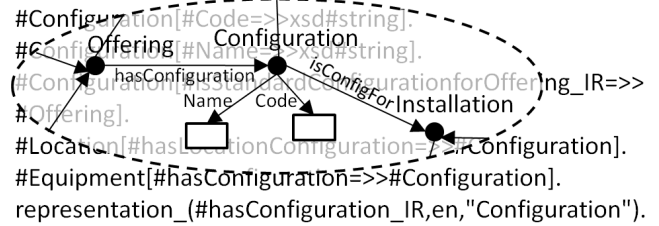
In order to integrate different data sources, the heterogeneity of the different media types and data representation formats has to be overcome. This is achieved by data providers that make the data from the sources available as graphs in an RDF format that corresponds to source-specific F-Logic ontologies. This step can be done fully automatically, as it is just a syntactic transformation.

- Database tables are transformed into classes, rows into instances and columns into attributes.
- XML nodes are transformed into instances or attributes, XML Schema types into classes, XML child nodes are linked to their parent nodes with relationships.
- Spreadsheets are transformed in a very similar way as databases. The transformation can be done.
- For unstructured documents, the metadata attributes (e.g. title, author, date of creation etc.) are transformed into RDF statements. The text itself is left as analyzed later by the annotation service.

Example

The screenshot in Figure 10 shows an example where a user searches for configurations of devices used in the Chemical industry branch with AC it their name. The search terms are entered in a similarly simple way as users know from public Web search engines. However, to leverage the semantic model underneath, auto-complete suggestions with terms from the domain are offered to the user. This is supported by the fact search service that uses the ontology index of the semantic repository. When entered, the keyword query is than interpreted by the fact search service as a structured query, depending on disambiguated instance names, class names, role names and free-text terms. This is then executed by the semantic repository over the ontology designed

```
#Configuration[  
#Offering[#hasConfiguration_IR=>#Configuration].  
#Configuration[#Code=>xsd:string].  
#Configuration[#Name=>xsd:string].  
#Configuration[#hasConfiguration_IR=>#Configuration].  
#Configuration[#hasLocationConfiguration=>#Configuration].  
#Location[#hasLocationConfiguration=>#Configuration].  
#Equipment[#hasConfiguration=>#Configuration].  
representation_(#hasConfiguration_IR,en,"Configuration").  
...
```



The diagram illustrates the F-Logic code sample. It shows two classes: #Configuration and #Offering. #Configuration has attributes #Code and #Name, both of type xsd:string. #Offering has a relationship #hasConfiguration_IR to #Configuration. There is also a relationship #isConfigurationforOffering from #Offering to #Configuration. A third class, #Location, has a relationship #hasLocationConfiguration to #Configuration. A fourth class, #Equipment, has a relationship #hasConfiguration to #Configuration. The code sample also includes a representation function: representation_(#hasConfiguration_IR,en,"Configuration").

Figure 16: F-Logic code sample of the ontology

collaboratively by domain and ontology experts. An excerpt of the ontology, modeled in FLogic, is shown in Figure 16.

The reasoner executes the query and, based on mapping rules (cf. section 6.3.2), decides, for which sub queries to perform database SQL queries via the JDBC builtin (similar as in [2]). Other facts necessary for the query may be permanently stored in the semantic repository, e.g. imported from an XML dump of a legacy application via the XML importer before query time. After query execution, graphs are generated that explain to the user, why each particular found instance is considered to be a match for the entered keywords. The same search box can also return documents from the uncertain repository. In this case, the query is interpreted by the document search service, not as a structured query, but as a vector of free text terms (via the document full-text index in Lucene) and disambiguated URIs (via the semantic annotations in the RDFS store).

7. EVALUATION AND DISCUSSION

Although the work on the reference implementation of Aletheia is still in progress, we already evaluated the initial results with regards to aspects such as usability, search functionality, data sources, interfaces, and added value due to the semantic technologies.

7.1 Internal Evaluation

The evaluation involved all Aletheia project members and has been conducted using a survey on the prototype variants built for all of the five scenarios. With ten responses and a total of 59 rated criteria, this initial feedback was mixed:

- In summary, an earlier version of the implementation in Section 6 received 26 positive, 15 negative and 15 “partially fulfilled” ratings, with regards to the collected use cases and requirements. Here, partially fulfilled means that the evaluator did see some of the capabilities implemented w.r.t. a certain requirement, but not in its entirety that is expected for a final prototype.
- Several of the individual solutions that have been implemented for scenario-specific use cases, such as the recognition of product terms in full-text requests and graph visualizations, received very positive feedback, both for highlighting the benefits of semantics and usability of the search function. Hence, we currently integrate them as components to the reference implementation.

- As expected, some evaluators criticised that not all potential information sources are integrated already. They evaluated a comparatively early version of the prototype. Since then, the available data providers have been extended considerably.
- The benefit of semantics is not obvious for a few use cases related to the Internet of things. The implementation did not utilize this information in conjunction with the stored knowledge, and the higher-level integration is clearly necessary to show the advantage.

We will present the results of a more comprehensive evaluation to be executed later this year. Hence, we rather discuss the lessons learned to date with the development of this FPIS.

7.2 Discussion

Within the Aletheia research project, we prototypically set up an integrated browsing and searching application over heterogeneous product data sources in realistic industry environments, using a technology that represented the integrated information as a graph. In principle, this proved to be a feasible approach that offers a number of advantages acknowledged by the application partners.

The most important advantage is that users can intuitively enter search queries whose answers require knowledge of how objects are related to each other (independently from where the objects are stored). The relationships can also be leveraged for graph visualizations to provide end users with advanced insight into complex dependencies in the underlying data. Moreover, the graph model makes it easier to let users access all relevant pieces of information together, including documents related to objects described in a database. Such features can be supported in a generic way for all relationships, without the need to hardcode special cases. It is generally not necessary to resort to heavy-weight semantic technology with advanced reasoning capabilities. Instead, it is sufficient to be able to view and query data according to a graph model.

As shown above, the current reference implementation exhibits most of the benefits of the intended Aletheia system. Although the architecture proposes the federation of at least structured information at the time a client actually poses a query, we noticed that this is difficult to accomplish. For the auto-complete functionality presented in Figure 10, an index of the name and other properties a semantic entity can be referred to *must* be available in the system in order to meet response time constraints (latency). This functionality is present in the employed semantic repository implementation, but is limited to the facts stored in the repository. Even though the repository federates information from other structured sources like relational databases, it does not make them available on this index unless this data is materialized, i.e. replicated to the repository.

As we argued in section 4.2, uncertain semantic information are separated from syntactic data, e.g. a full-text index, in the proposed architecture. While this is a sensible decision due to the different nature of those repositories, we employed the Sesame LuceneSail [16] component which combines both aspects. Hence, the management of extracted facts like `<Document1> <isAbout> <DriveComponentA>` is accomplished by the same component that stores the full-text index of that document.

We further noticed that the traceability of federated information is difficult. This is due to the overhead of storing provenance information for every fact in the repository, and gets even more complicated if the information should be annotated with confidence or trust ratings. While initially considering RDF reification, we also studied whether named graphs can be used appropriately for such annotations. The requirement of federated ranking would benefit from such reliable confidence assertions. We found, however, that this poses a high processing load on the system that can't be handled using current technology for the required amount of information.

The authentication and, to a greater degree, the authorization of users to access certain information remains an issue. Considering the federation during a client's request, existing single-sign-on (SSO) solutions can be applied. As the auto-complete issues have shown, the ad-hoc federation is an approach that causes several difficulties. However, federating the information prior to actual requests and, hence, not requesting the original information sources requires the repository to keep track of access rights to individual information. Additionally, this causes the data providers to determine and relay this authorization information in the first place. This is a critical aspect for all studied scenarios, and is subject to current work on integrating an authentication and authorization component.

Collectively, the current implementation provides a complete "vertical cut" through the architecture, integrating different heterogeneous data sources. It also proved to be applicable to different domains by means of switching the semantic model, i.e. the ontology, and connected data sources.

8. CONCLUSIONS

In this paper, we have presented an architecture that supports the federation of heterogeneous information, originating from various data sources and arising throughout the product lifecycle. We propose this solution with regards to the limitations of current product information management products. These are less flexible than this semantic approach and usually only cover few aspects of the product lifecycle.

Prior to that, we generalized a number of requirements derived from multiple real-life scenarios. The proposed architecture and associated reference implementation enables the exploration of information, both using semantic search and exploring related information. Most of the required data sources have already been integrated, based on generic solutions like XML to RDF transforms, in order to provide the desired all-embracing view. Internet of Things devices, such as wireless sensor networks and RFID tags, can be integrated using the proposed gateway architecture. Existing frameworks for information extraction are attached that integrate unstructured information, using the respective domain ontology and existing knowledge.

The presented information can also be modified by the users of the system, thereby keeping the product information up to date, even though these modifications are not performed at the information's origin. Finally, we presented our vision of enabling collaboration of such federated product information systems between different organisations, which is a requirement for exploiting the full potential of such a solution.

Compared to an earlier publication [31], this paper presents our approach and related work in much more detail. It also

discusses the Internet of Things integration and more thorough results regarding the reference implementation.

9. ACKNOWLEDGMENTS

This work was partly funded by the German Ministry of Education and Research under the research grant number 01IA08001.

We would like to thank many Aletheia project members contributing to this architecture and reference implementation, including but not limited to Bernd Stieger, Nic Fantana, Thomas Janke, Tobias Münch, Robert Rieger, Sandro Reichert, and Maximilian Walther, as well as project partners for fruitful discussions on use cases and requirements.

10. REFERENCES

- [1] Aletheia project consortium. Aletheia - semantic federation of comprehensive product information. <http://www.aletheia-projekt.de/>, 2010.
- [2] J. Angele and M. Gesmann. Data Integration using Semantic Technology: A use case. In *Rules and Rule Markup Languages for the Semantic Web, Second International Conference on*, pages 58–66, 2006.
- [3] Aperture Framework. A Java framework for getting data and metadata. <http://aperture.sourceforge.net/>.
- [4] W. Behrendt, A. Gangemi, W. Maass, and R. Westenthaler. Towards an ontology-based distributed architecture for paid content. In *Proceedings of the Second European Semantic Web Conference*. Berlin, Springer, 2005.
- [5] R. Bhagdev, S. Chapman, F. Ciravegna, and V. Lanfranchi. Hybrid search: Effectively combining keywords and ontology-based searches. In *5th European Semantic Web Conference (ESWC2008)*, pages 554–568, June 2008.
- [6] C. Bizer, T. Heath, and T. Berners-Lee. Linked data—the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
- [7] J. Broekstra, A. Kampman, and F. Van Harmelen. A Generic Architecture for Storing and Querying RDF and RDF Schema. *The Semantic Web-ISWC 2002*, pages 54–68, 2002.
- [8] J.-S. Brunner, L. Ma, C. Wang, L. Zhang, D. C. Wolfson, Y. Pan, and K. Srinivas. Explorations in the use of semantic web technologies for product information management. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 747–756, New York, NY, USA, 2007. ACM.
- [9] S. J. DeRose, E. Maler, D. Orchard, and N. Walsh. Xml linking language (xlink) version 1.1. Technical report, W3C, 3 2010.
- [10] D. Ferrucci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, 2004.
- [11] J. J. Garrett. Ajax: A new approach to web applications. Essay at <http://www.adaptivepath.com/ideas/essays/archives/000385.php>, February 18, 2005.
- [12] G. Kasneci, M. Ramanath, F. Suchanek, and G. Weikum. The YAGO-NAGA approach to knowledge discovery. *SIGMOD Rec.*, 37(4):41–47, 2008.
- [13] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, pages 85–91, 1995.
- [14] S. Kunz, F. Brecht, B. Fabian, M. Aleksy, and M. Wauer. Aletheia—improving industrial service lifecycle management by semantic data federations. *International Conference on Advanced Information Networking and Applications*, pages 1308–1314, 2010.
- [15] S. Liang, P. Fodor, H. Wan, and M. Kifer. Openrulebench: an analysis of the performance of rule engines. In *Proceedings of the 18th international conference on World wide web, WWW '09*, pages 601–610, New York, NY, USA, 2009. ACM.
- [16] E. Minack, L. Sauermann, G. Grimnes, C. Fluit, and J. Broekstra. The sesame lucenesail: Rdf queries with full-text search. *Technical Report 2008-1, NEPOMUK*, 2008.
- [17] J. Paralic and I. Kostial. Ontology-based information retrieval. *Information and Intelligent Systems, Croatia*, pages 23–28, 2003.
- [18] L. Patil, D. Dutta, and R. Sriram. Ontology-based exchange of product data semantics. *Automation Science and Engineering, IEEE Transactions on*, 2(3):213 – 225, July 2005.
- [19] G. Reif, T. Groza, S. Scerri, and S. Handschuh. Final NEPOMUK Architecture – Deliverable D6.2.B. Public deliverable of the NEPOMUK project, Dec 2008.
- [20] T. Riedel, N. Fantana, A. Genaid, D. Yordanov, H. Schmidtke, and M. Beigl. Using web service gateways and code generation for sustainable IoT system development. In *Internet of Things (IOT), 2010*, pages 1–8, 2010.
- [21] C. Rocha, D. Schwabe, and M. P. Aragao. A hybrid approach for searching in the semantic web. In *Proceedings of the 13th international conference on World Wide Web, WWW '04*, pages 374–383, New York, NY, USA, 2004. ACM.
- [22] T. Rodrigues, P. Costa, J. Cardoso, and J. Fernandes. Jxml2owl. <http://jxml2owl.projects.semwebcentral.org>, 2006.
- [23] S. Schenk, C. Saathoff, A. Baumesberger, F. Jochum, A. Kleinen, S. Staab, and A. Scherp. Semaplorer-interactive semantic exploration of data and media based on a federated cloud infrastructure. *Web Semant.*, 7(4):298–304, 2009.
- [24] T. Schütz. D11.1.1.b concept and design of the integration framework. Public deliverable of the Theseus-Ordo project, Sep 2008.
- [25] S. Soga, Y. Hiroshige, A. Dobashi, M. Okumura, and T. Kusuzaki. Products lifecycle management system using radio frequency identification. In *Proc. of IEEE Int. Conf. on Emergent Technologies and Factory Automation (ETFA99)*, page 1459–1467, 1999.
- [26] V. Srinivasan, L. Lämmer, and S. Vettermann. On architecting and implementing a product information sharing service. *Journal of Computing and Information Science in Engineering*, 8(1):011006,

2008.

- [27] A. Tiwari, F. L. Lewis, and S. S. Ge. Wireless sensor network for machine condition based maintenance. In *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, volume 1, page 461–467, 2005.
- [28] D. T. Tran, P. Cimiano, S. Rudolph, and R. Studer. Ontology-based interpretation of keywords for semantic search. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, pages 523–536, Busan, Korea, NOV 2007.
- [29] T. Tran, P. Haase, H. Lewen, O. M. Garcia, A. Gomez-Perez, and R. Studer. Lifecycle-support in architectures for ontology-based information systems. In *Proceedings of the 6th International Semantic Web Conference (ISWC'07)*, November 2007.
- [30] T. Tran, H. Wang, and P. Haase. SearchWebDB: Data web search on a pay-as-you-go integration infrastructure. Technical report, University of Karlsruhe, 2008.
- [31] M. Wauer, D. Schuster, and J. Meinecke. Aletheia - an architecture for semantic federation of product information from structured and unstructured sources. Paris, France, 2010. ACM.