SOFTWARE DEVELOPMENT PROJECT MANAGEMENT

Process and Support

DINA BERKELEY Ph.D.
London School of Economics and Political Science. London
ROBERT DE HOOG Ph.D.
The University of Amsterdam, The Netherlands
PATRICK HUMPHREYS Ph.D.
London School of Economics and Political Science, London

ELLIS HORWOOD
NEWYORK LONDON TORONTO SYDNEY TOKYO SINGAPORE

First published in 1990 by
ELLIS HORWOOD **LIMITED**
Market Cross House, Cooper Street,
Chichester, West Sussex, PO19 1EB, England

A division of
Simon & Schuster International Group
A Paramount Communications Company

Printed and bound in Great Britain
by Hartnolls Limited, Bodmin, Cornwall

British Library Cataloguing in Publication Data

Berkeley. D., de Hoog, R., Humphreys, P.
Software Development Project Management: Process and Support: — (Ellis Horwood book in information technology).

CIP catalogue record for this book is available from the British Library

ISBN 0–13–829847–5

Library of Congress Data available

Table of Contents

Preface

This book is based on, and extends, some of the results of ESPRIT I project 814: PIMS (Project Integrated Management System). This project was set up to develop a project management support system which could aid the management of medium size software development projects. A basic tenet of the project was that theory and practice should go hand in hand This book follows the same principle.

PIMS was a three and a half years long project with a total effort of about 56 man-years, partly funded by the Commission for the European Communities (CEC). The following organisations were involved in the PIMS project: Buro voor Systeemontwikkeling (BSO), Eindhoven, the Netherlands; Cap Sogeti Innovation (CSI; now Cap Gemini Innovation). Grenoble, France; London Business School (LBS). London, England; London School of Economics and Political Science (LSE), London, England; PA, London, England; Senter for Industriforskning (SI), Oslo, Norway; University of Amsterdam (UvA), Amsterdam, the Netherlands. The final software product of this project has been used by the partner organisations in managing their own development and is currently under further development to pave the way to its commercial availability.

This book presents mainly the theoretical results of the project although references to the practical results are made as and when required. It embeds and draws upon the extensive software development project management experience of the industrial partners of the consortium (BSO, CSI, PA). We can not be thankful enough to all the project managers from these companies who have shared their experience with us, enlightening our understanding of the project management process and our appreciation of its inherent difficulties.

We are also grateful to all our colleagues in the PIMS project for their contributions to our work and their constructive criticism which led to the sharpening of our thinking. In particular, we would like to thank: Willy Cats-Baril (LSE), Keith Dixon (BSO), Christer Fernstrom (CSI), John Hawgood (PA), Hans van de Klok (BSO), Frank Land (LBS), Annie Leclerc (CSI), Maurice Schlumberger (CSI), Gisle Stokke (SI), Ian Thomas (CEC expert reviewer), Philippe Vauquois (CSI) and Michel Vogler (UvA). For their indirect contribution to this work and, in particular, for teaching us the use of Petri nets, we would like to thank Hartmann Genrich, Gernot Richter and Klaus Voss from Gesellschaft fur Mathematik und Datenverarbeitung (GMD), Sankt Augustin, West Germany. Last but not least, we would like to thank Dimitris Tsoubelis (LSE) for his help in drawing the figures in this book.

Please note that, for reasons of convenience, only the masculine pronoun has been used to refer to the project manager, team members and other parties throughout this book.

Chapter 1

Introduction

A great number of books exist in the market whose subject matter addresses project management directly or indirectly. These tend to vary in scope, content and objective. Thus, they may be found useful in different ways or by different readers.

In general, one could classify such books into two distinct categories: those which are providing a general, discursive, scoping of the area (including suggestions about good practice, presenting success or failure stories about the management of projects, etc.) and those which are more concerned with particular aspects of management (e.g., detailing techniques, methods and tools which managers can use in their work). This latter category could be further subdivided into specialised areas (e.g., estimation and risk models, people management).

Within the domain of managing software development projects, most of the existing books tend to concentrate on the development process itself and treat its management at the same level as any other specific topic in the book (in detailing the tasks of the manager). Such an approach is justifiable given the readership these books wish to address: managers of software development projects who may wish to learn more about system development and how to improve the development process as well as the management process, and how to manage such projects better.

The present book also wishes to address project managers and attempt to aid them in their work. However, it starts from a different vantage point: project management itself. It concentrates on projects with software development as their subject matter, often described as the trickiest and most difficult type of project to manage. In this book, software development projects are treated as a specialised area of concern but only for the purposes of providing examples and enhancing understanding of what is being described. If this book can aid management of software development projects, we are convinced that it will be of use in aiding the management of any other type of project.

Moreover, we wish to address other types of readers as well as project managers. This is not simply by desire hut comes as a natural consequence of the context from within which this book was born, that is, an actual software development project whose end-result was a project management support system. This was the **ESPRIT** I project **PIMS** (Project Integrated Management System). Thus, the book has a practical as well as a theoretical orientation on the process of supporting system development, and, hence, it is also of particular relevance to developers of such systems. Through the development of the two **PIMS** prototypes, we discovered what works or does not work in practice during the process of developing the software system, the needs of project managers that have to be met through a computer-based project management support system, what kind of system architecture and design matches their preferred methods of work, and so on. Thus, this book can provide a good starting base for projects building project management support systems as well as provide details on modelling project management activities and the data they need in order to be carried out.

A further element which distinguishes this book from other books on the same topic area is that not only does it address what is involved in project management, discuss principles and provide information about the development of computer-based systems that support it, it also provides a case study running throughout the book on how to proceed from discovering what is involved in an organisational process, through modelling the process and the objects which it manipulates. to developing design principles on the basis of the models generated. Thus, it provides fruitful material for students just as much as practitioners with concerns beyond project management (such as those working on, or studying, information systems, systems analysis or organisational theory).

Finally, the majority of the material presented in this book is based on information we have gained through in-depth interviews with many experienced software development project managers, focusing on elicitation of knowledge relevant to the concerns of the **PIMS** project Relevant literature is used supportively rather than as the prime source of information in the discussion of the process of project management presented here. Thus, the book is founded on the practical experience of these managers gained through their work for considerable lengths of time as project managers, in various types of software development projects, at various levels of responsibility, within various industrial organisations, in three countries (England, France, and the Netherlands). It was instructive to discover that these variations in context played almost no role in accounting for the general difficulties involved in managing software development projects, although they did play a significant role in determining the way these difficulties might be handled.

1.1. Overview of the book

Chapters 2 through 5 of this book introduce the context, process and concerns of pmject management in a discursive fashion and can be read as an introduction to software development project management. In particular, chapter 2 provides an informed account of the environment within which projects materialise and its influence on the success of a project. Chapter 3 describes the risks that may threaten the success of a project and how these should inform its management. Chapter 4 discusses the actual process of project management. Chapter 5 brings together the concerns of project management discussed in the preceding chapters to address the goals of project management, indicating how they motivate project management activities and guide the perspectives that can be taken by a manager in collecting and organising information about his project.

Chapter 6 concentrates on a discussion of the modelling approach and concepts we employ, detailed only to the degree necessary to understand the remainder of the book. Chapter 7 presents pre-formal descriptions of a typical range of project management activities (PMACs) and chapter 8 proceeds to show how particular project management activities may be formalised in the development of a project management model. Chapter 9 examines in detail the processes of inference and diagnosis incorporated within management which are used in controlling progress on a project, in setting up an observation and reporting system and in identifying and predicting discrepancies between planned and actual progress. Chapter 10 presents the generic core of the project data model, which may be developed to provide a repository for the information about the project and its environment which is necessary for the execution of any set of project management activities.

In this book, developing models is not seen as an end in itself. Models are built for a purpose, thus, they are a means to an end. The desired end of this book is to use these means as a step towards understanding the kind of support that can be provided to a manager through a computer-based system. Thus, in chapter 11, we discuss the necessity of considering the division of labour between the manager and a support system for activities they can both share in carrying out the management of the project as a pre-requisite to even starting to think about how a project management support system may be developed. Chapter 12, then, uses this understanding and the results of the previous parts of the book to discuss, with examples, key issues in the development of project management support systems. Finally, chapter 13 discusses the implications of the material in this book for project management itself and for the future provision of integrated support for the whole process of project management.

1.2. Suggested usage of the book

This book is not intended as a first introduction to project management. However, much of it can provide useful insight into project management concepts, issues and processes. For example, chapters 2 to 5 can function in this way as they provide a discursive, organised account of the project management process although this is not addressed across all topics in a great detail. A reader interested in learning about the scope of project management and its problems will find these chapters of use. Readers interested in learning more about specific techniques of project management, which are well covered in the literature, are referred, at the appropriate points in the text, to other material (books, articles) where this information may be found, as detailing such concerns falls outside the scope of this book.

However, we have provided detailed information in the case of concerns of project managers which genuinely cause problems for them where relevant information about how to handle them does not exist in the literature or, at least, is not presented in an integrated way. We had the opportunity within the PIMS project to tap the practical experience of many project managers and we are privileged to be able to share much of the information so provided with readers. Of particular interest in this respect may be our discussions on the issues to consider in assessing the quality of a project plan (in section 4.2.2.3) and the various types of "what-if' project simulations (in section 4.3).

This book was designed to be read in a linear fashion, building an argument throughout its structure and content. Thus, the "whys and wherefores" of a conclusion or a certain recommendation for practice (e.g., for management or development) will be usually found in a preceding analysis of the corresponding topic. Hence, we would recommend that the reader reads this book from start to finish (at least, for the first time) so that its implications will become apparent and its didactic intention achieved. However, once the argument developed has been grasped, the book can be used as a reference source of information on any of the various aspects it covers, for example, on the project management process, learning about modelling, developing a particular management support technique for inclusion in a computer-based system, and so on.

Finally, a word of caution to the reader. We do not profess that, in this book, we have addressed *all* aspects of software development project management in the same degree of detail. Some aspects, peripheral to the main developments covered in this book, have only been sketched in. Still, we hope that this book can provide the reader with an understanding which will enable him to improve on what we have written by using his own experience to enlarge on the material presented here. If the reader finds this possible, our goal for the book will have been achieved.

Chapter 2

The context of project management

Managing a project is not an easy enterprise. At its very core, the management process involves handling and facilitating complex interactions between and within various groups of people who are directly or indirectly involved in the project and are interested in its successful conclusion. The skills needed for accomplishing this task vary as the needs and the activities of the project evolve and change during its lifetime.

A project is a transitory organisation of individuals dedicated to the attainment of a specific objective within a schedule, budget, and technical performance target. It entails a purposeful, non-routine, activity that involves collaboration amongst people. Its purpose is "to work itself into ultimate dissolution after the objectives of the project have been accomplished (Cleland & King 1983). It is limited in its use of time and other resources, the scope of the work that is being done within it and the clientele to whom its products will be delivered. Thus, as an entity, a project is defined both in terms of the work that is being carried out within it (to achieve the accomplishment of its objective with technical quality) and in terms of what is being made available to it to achieve this objective (e.g., budget, timescale, quantity and quality of resources).

Projects differ in terms of the nature of work being carried out within them (e.g., construction projects, research and development projects, software development projects). Each area of work necessitates the use of particular methods and techniques within the project work to ensure that the quality of the product will be satisfactory to all those concerned (client and contractor organisation alike) and brings with it its own specifications for required skills of resources who will be employed in the project. Even within each particular area of work, we come across different types of project, depending on the particular application domain (e.g., a bridge construction project, a house construction project). The methods and techniques used in each application domain within a particular area usually differ only in terms of how they are implemented (e.g., the materials they will use, a further refinement of required skills of resources).

In terms of the resources made available to a project to achieve its objectives, the principal differences between projects are in terms of their size or budget and timescale. Thus, projects may be characterised as large or small, long-term or short-term, regardless of whatever their application area might be, although what is taken to be the (medium size) norm will vary across application areas. The difference these aspects of the project make has repercussions on the type of managerial skills required for the project to achieve its objectives. A large construction project, for example, will necessitate different managerial skills from a small one. Similarly, a large software development project will need different management skills from a small one. Large projects, typically, require disproportionately more interactions with other agencies, often necessitating complex negotiations, and requiring advanced project management expertise.

In fact, the skills required for successful project management depend more on the size and complexity of the project than on the application area of its project. The manager of an aerospace project has to carry out the same general types of operations as does the manager of a software development project (that is, planning, controlling, negotiating, etc.). The application area of the project impacts on specific rather than general aspects of this work (e.g., what is being planned, controlled, negotiated) and on the technical experience which it is desirable for the manager to possess (e.g., airplane development experience versus software development experience). The fundamental measure of success in project management also does not differ. It is based on the completion of the project with agreed manpower, resources, timescale and cost, with the facilities and service performance provided, and to the satisfaction of his own organisation, the client and the project team (Keen 1987).

In this book, we take software development project management as an example, showing how project objectives may be achieved through its successful execution. To that extent, our discussion of project management is biased by the specific examples we use. However, in that the nature of such projects places them among the riskiest ones to carry out and manage of those that can be found in any application domain, this only makes our discussion that much more interesting. We believe that most of what we discuss can also be applied to projects in other application areas which, by their nature, are easier to carry out and manage, with many fewer provisos.

2.1. Problems in software development project management

A major part of the difficulties encountered in software development projects has to do with the intangible nature of what they are set up to produce. As Licker (1985) points out, a software development product is "..like a hammer, its value is known only when it is used. Unlike a hammer, it is a unique tool, one that probably has never existed before and that may be valuable only to a small group of peculiar and untrained individuals.."

There are different types of software development projects, each with its own difficulties, risks, and requirements for special management skills. Keen (1987) describes these under four headings:

(a) Application orientated projects which develop an applications system implemented on an existing computer utilising established system software. These may relate to a particular self-contained function in the organisation (e.g., payroll), or to the implementation of a function spanning the whole organisation and involving many departments (e.g., non-integrated order sales system), or to those systems which rely heavily on data and files of other computer systems (e.g., accounting system), or to "across-the-board systems which assume the existence of all of the above (e.g., a management information system). The experience and required skills of the manager of these projects increases with the complexity of the system to be developed (in the order just described).

(b) Projects that install hardware, involving the implementation of a computer configuration (including tasks such as site preparation and acceptance tests). Such projects are usually relatively easy to manage unless early delivery of a new range of hardware or software is involved.

(c) Software implementation projects such as the introduction of new file structures for a future application whose difficulty relates to the end-product itself.

(d) Projects involving combinations of the above.

Assessing the easiness or difficulty of managing a particular project is based on determining whether or not the project has characteristics which are typically associated with problems in any project in the area being carried out successfully. For example, a project which involves enhancement of an already existing software system will be far less risky or in less need of expert project management than a project which is developing a similar system from scratch. This is because the latter kind of project faces a lower probability of getting the user requirements wrong (a typical problem in software development) as it starts from some already established requirements (the existing system) against which new requirements can be developed (Burns & Dennis 1985).

Developing software according to the *real* (ultimate) requirements for it is quite an infrequent occurrence in software development projects. In reality, change of requirements is one of the main causes of project slippage and cost overruns. Moreover, requirements are often incomplete, ambiguous, and/or inconsistent (e.g., Thayer & Lehman 1979). In a study of 23 organisations and 72 software development projects. Jenkins and Wetherbe (1984) discovered that, in two-thirds of the projects, the requirements analysis effort had to be repeated. Additional requirements were discovered either because something was missed out in the original requirements or because something changed in the application environment or the way the end-users expected to work with the system. However, they also found that only half of the managers of these projects had planned for this iteration of requirements analysis and its consequences. Various techniques are often used in an attempt to alleviate this

problem through making change control easier (e.g., structured analysis, DeMarco 1979; application prototyping, Boar 1984). Alternatively, the development of a close relationship between the user and the analyst/developer is often suggested as the best way of preempting eventual problems (e.g., Ferratt & Starke 1978, Lehman 1979, Farbman 1980).

While requirements that are potentially wrong or likely to change are a principal source of anxiety for the manager of a software development project, they are not the only one. Thayer, Pyster and Wood (1981), in a survey of the experience of 294 managers, identified some additional causes of problems which included:

- poor ability to estimate accurately and plan;
- lack of standards and techniques for measuring the quality of performance and the quantity of production expected from the project team,
- lack of decision aids in choosing the proper organisational structure or selecting the correct management techniques;
- lack of techniques which can aid the visibility of the project's progress or its effective control;
- poor accountability defining who is responsible for what;
- inappropriate success criteria for software development.

While the causes listed above posed problems for the project manager in fulfilling his responsibilities, their effects may be propagated through his activities onto the quality of the work actually carried out in the project. Endres (1975) traced sources of errors in the actual project work to

- ***technological*** causes such as definability of the problem, feasibility of solving it, available procedures and tools;
- ***organisational*** causes such as division of workload, available information, communication, resources;
- ***historical*** causes such as the history of the project, of the program, special situations, external influences;
- ***group dynamic*** causes such as willingness to cooperate, distribution of roles inside the project team;
- ***individual*** causes such as experience, talent, constitution of individual team members;
- other and inexplicable causes.

The causes of problems which are encountered in software development projects are usually identified through retrospective analyses of the project after it has been completed or abandoned. However, Parkin (1983) has argued that

> "any attempt to review past failed projects through questioning the participants, or even by questioning oneself if one has participated in a failed project, is of dubious objectivity. The causes one is likely to dredge up are the subjective opinions or prejudices of questioner and questionee. What one is really getting is more like advice for the future... The success or failure of a project is subjective and can be a matter of dispute which is

not reconciled even when observers have discussed their viewpoint.." (p. 70).

Even when the cause of a problem has been undisputably identified, the solutions offered to the problem may be subject to dispute themselves. A solution originated by a manager, for instance, may be seen by the programmer as ***fixing only parts*** of ***the problem*** rather than providing a lasting solution (Fairley 1985).

Furthermore, many of the potential problems that may befall the project do not arrive together with the project brief. One can not always anticipate them clearly from the start of the project and try to plan for handling them. Things may happen after the project has started, necessitating a change in strategy or forcing some immediate action. For example, the urgency of the project work may increase, as the product is needed ***now,*** or incompatibility may be discovered between the specified software development tools and the hardware on which they are expected to run. In such situations, the manager is faced with an emergency which requires the immediate use of management suategies which are qualitatively different to those employed in non-emergency situations (Shaw 1984).

The problems we have talked about so far originate from outside the project, and the manager has to strive to handle them effectively through his activities. However, there are additional problems which come as part and parcel of the reality of managing a project. Thamhain and Wilemon (1986), in a study of 400 project managers on their experiences in regard to important challenges to their management skills, list the following issues, indexed by the degree of frequency of each issue having been mentioned by their informants:

- coping with end date driven schedules (85%);
- coping with resource limitations (83%);
- communicating effectively among task groups (80%);
- gaining commitment from team members (74%);
- establishing measurable milestones (70%);
- coping with changes (60%);
- working out project plan agreement with team (57%);
- gaining commitment from management (45%);
- dealing with conflict (42%);
- managing vendors and subcontractors (38%);
- other (35%).

Such problems, whatever their source or cause, define the ***everyday reality*** of the manager rather than highlight ability to handle special project problems and emergencies. In the next section, we set these everyday problems in context by considering the project, its environment and the role of the manager in their interface. This paves the way to discussing the project management process as the principal concern of this book.

2.2. Managing the project in its environment

The project can be seen as a microcosm which is both a socio-technical and human activity system: it has its own demands for resources, identity, need for appreciation, rewards, and so on. Any satisfaction it can gain, though, is dependent on what it itself can produce to satisfy others whose interest has been instrumental in setting it up. The project's main reason for existence is the need of somebody else for its end-result, the software system, which it can alone produce (desirably, in the agreed-upon way and in the agreed-upon form). Hence, the project has two types of stakeholders comprising parties who either affect or are affected by the project (Mitroff 1983). These are:

- ***external stakeholders*** such as the client organisation both as the project sponsor/financier and the future user of the system under development†, senior management and other managers within the project manager's own organisation as the contractor to the project, external agencies such as suppliers or regulatory agencies, other subcontractors; and,
- ***internal stakeholders,*** the people working in the project (i.e., the project team members), in that the success or failure of the project is affected by their own work and will eventually affect them directly whatever its cause.

It would be naive to assume that, since all these stakeholders have an interest in the project (that is, share a common desire for the successful completion of the project), they also share the same view on what constitutes a successful completion of the project. In fact, the aspects of the project which each stakeholder particularly stresses (and those aspects on which each is prepared to compromise) are determined by their particular interests and current goals. For example, the client, as the sponsor, may be prepared to compromise on the functionality of the software system in order to limit an increase in development costs. On the other hand, prospective users within the client organisation may be less concerned with the costs involved than with ensuring that the functionality they desire can be actually achieved. Moreover, different prospective users may have differing expectations for the software system under development (Keen & Gersch 1984). Within the contractor organisation, on the other hand, the desire to make a financial profit on the contract may result in a failure to provide the project with the amount, kind or quality of resources necessary for its successful execution.

The project manager is, in fact, the point of contact between all these external and internal stakeholders once his project has been established. His job is to transfer information from one stakeholder to the other, facilitate communication between them, plan the software system development and oversee its progress, and manage the people under him, their interrelationships and their activities, and his own relationships with other stakeholders. Most importantly, he must ensure that all parties have a clear and unambiguous understanding of the project and that this is the ***same*** understanding (Taylor & Watling 1979).

† The actual persons playing these different roles may not be necessarily the same.

His project itself is a central reason for the existence of this network of complex relationships between the various parties.

The project manager himself is a stakeholder in the project in the sense that he has a stake in the successful completion of it (leading to a promotion, recognition, etc.). However, his position at the *interface* between the project and its environment makes him sometimes appear as an external stakeholder, as in his relation to the project (in reviewing products and project activities), and sometimes as an internal stakeholder, as in his relation to the project environment (in presenting the results of the work of his team to external stakeholders). This dual role complicates his work even further.

Given this complex network of interrelationships, itself nested within the world of technology (e.g., available hardware and software), the world of business (e.g., competing contractor organisations) and society at large (people who might be affected by the project or what it produces), problems that often arise during the lifetime of a project are not always traceable to bad project management or lack of experience on the part of the manager (Morris & Hough 1987). It is not always easy to foresee *all* that can go wrong within a project.

For example, the project manager can not always foresee whether the client will turn out to be unwilling or unable to allocate some of his time to the project as a user to try out its intermediate results even if this is necessary; whether human resources will be withdrawn from the project or whether machines will break down unexpectedly; whether personal conflicts will arise within the team; whether the client will refuse to accept the developed product as being what he wanted; and so on. Of course, if the project manager can cope successfully with each problem of this kind as it arises, the result will still turn out to be satisfactory. Project managers, in fact, often do cope with unexpected problems quite well. Long experience helps a great deal in this respect. The difficulties sometimes arise later, when the unexpected problem has forced them down a decision path with *future* negative consequences (e.g., "cutting down" functionality which could increase the risk that the client will not accept the final product).

Hence, the manager's job is invested with complexity: his work is necessarily carried out in the midst of a great deal of conflicting interests and pressures. It is also loaded with uncertainty since the project he is managing is a *human activity system* (Checkland 1981, Mumford 1983a) vulnerable to disturbance from within itself and *from* the outside environment. In order to be able to cope with the complexity of his work and handle the uncertainties involved in carrying it out successfully, the project manager must be able to consider his project from the viewpoints of the various stakeholders involved in the project, anticipate their expectations of the project and speculate on the assumptions they make about it. Using the information gained this way, he must be able to manage his project in such a way that it can be carried out successfully according to the concerns of the technological, economic and social systems within which it is situated (Scacchi 1984).

Finally, the project manager's responsibilities are not confined to the particular project he is managing: he also has to think beyond it both in time and in organisational scope. An obvious and immediate concern is the furthering of his own career by leaving a good impression on other stakeholders and learning from the experience he has gained in managing the particular project. By the same token, he needs be concerned with the longer term interests of the members of his project team, furthering their own professional development and career prospects. Since, from the parent organisation's point of view, projects are evaluated not only in terms of their conmbution to the financial profit-related success of the business, but also in terms of the repeat sales, new contracts, company prestige and organisational learning which may result (Fife 1988), the project manager has also to be concerned with how his project can contribute to these wider interests of his own organisation. Whether or not these interests are satisfied can only be ascertained after the project has been completed. However, ensuring that they eventually *will* be means that he has to plan and manage his project with these interests in view *now.*

Projects are eventually evaluated retrospectively mainly on the basis of whether *all* the various stakeholders, internal and external, feel satisfied that all (or most of) their personal or organisational interests have been satisfied through it. It is the task of the project manager to ensure, to the best of his ability, that this will be the case and, certainly, this is not an easy task!

2.3. Systems relevant to the project management process

So far, we have described some of the difficulties facing a project manager in achieving his aim of managing his project successfully. The measure of the success of his work, however, depends to a great extent on how well his subordinates do *their* work (Herbert *1976).* The degree to which the project manager's work can affect their work positively defines his ability to manage his project and affect the quality of what it produces.

Still, this is only part of the story. As we indicated above, the road a project will eventually take towards its successful or unsuccessful execution is also paved by other interests and concerns the sources of which have nothing to do with the manager's ability. The manager's activities may only serve to shape such concerns in their eventual transformations and in how they finally do affect the project. These interests we have described as stemming from external stakeholders to the project (i.e., the client, other managers in his own organisation, subcontractors, suppliers, and regulatory agencies).

In fact, successfnl project management entails considering the concerns, needs, and interests of *all* internal and external stakeholders before, during and after the execution of any particular project management activity. Otherwise, its success will be jeopardised from the outset. For example, planning project work without considering whether the plan can create problems for the team *and* whether the same plan will meet the requirements that external stakeholders have of the project will almost certainly result in a failed project (at least,

from some stakeholders' point of view).

The major difficulties in project management stem from knowing how to balance the often conflicting concerns, needs and interests expressed by different stakeholders to produce a project management result which can express them all to a satisfactory degree to all concerned. To achieve all this necessitates that the manager has, and maintains, a ***system*** view on his work in understanding the people with whom he has to interact. This enables him to identify not only what motivates each particular system (an organisation or an individual) but also how different systems may interact and relate to each other. It also permits a better understanding of how conflicts between the interests of different systems may be handled and resolved through his own activities.

To address the concerns of this book, a similar view needs to be adopted, and here we can identify the following three major systems which need to be within the purview of the manager throughout the process of project management:

- the ***project work system,*** which he manages so that it can produce what is required of it,
- the ***contractor organisational system,*** which controls the manager and his project and which provides resources to the project, and
- the ***client organisational system,*** which provides the funding for the project, the requirements for the software system to be developed and for the project and which expects its deliverables.

Systems like those of suppliers, subcontractors and regulatory agencies are of subsidiary importance here as the manager may view them within the concerns of any of the three major systems. For example, suppliers may be viewed as resources provided by the contractor organisational system if formal contracts are made at a level higher within his organisation than himself. Subcontractors, on the other hand, may be seen as comprising part of the project work system, as resources to be used, amenable to his control. Regulatory agencies may be viewed as part of the client organisational system or the contractor organisational system in that they provide requirements for the project and expect the project to produce its products in a specified way in the same way as the client and the project manager's superiors do. Thus, while, in chapter 3, we will discuss the need to take views into the ***worlds*** of potential subcontractors, suppliers and other external stakeholders, we do not propose that taking a view into each of these worlds entails thinking about a separate system

Nevertheless, for the concerns of this book which relate to project management itself, one further system needs to be in ***our*** purview: the one comprising project management, the ***project management system.*** For the manager, this comprises his ***own*** activities and needs not be in his purview more than just detailing to him what to .do when and how; it does not exist without his own agency. For us, it is the ***primary system of interest.***

The ***project*** itself consists of both the ***project work system*** and its ***project management system.*** In organisational reality, a project is located within the contractor organisational system in that the members of the project (team

members as well as the manager) are members of the contractor organisation. The client organisational system, however, specifies the project's brief, provides resources for its execution (money, time, and often people) and expects satisfactory deliverables from the project. The project owes its existence to the existence of a client. Thus, the project may be seen both within the contractor organisation if one focuses on one particular aspect of it (i.e., membership, policy, objectives) and outside it if one focuses on another aspect of it (i.e., its definition). Chapters 3 through 5 will elaborate on the implications of this dual definition of the project for its management.

Chapter 3

Establishing a software development project

"Look before you leap, for snakes among sweet flowers do creep."
(Old English proverb)

Most software development projects result from a successful proposal to a client organisation as a response to an invitation to tender for the development of a system which can solve a particular organisational problem (Berkeley, Fernstrom & Humphreys 1987). In this chapter, we shall consider what is involved in deciding whether to bid for and accept the undertaking of a software development project. Focusing on this decision making process is important because the way it is carried out and the results achieved at each stage of the process pave the way to a successful or unsuccessful management of such projects. The results of this process will determine the constraints under which the project manager will have to operate in carrying out his responsibilities. Even the most capable manager cannot succeed under unrealistic or impossible conditions set up in the contract (e.g., optimistic estimates, an underbudgeted project due to underbidding in order to get the contract against other competitors).

The decision making process itself consists of three steps leading towards the formal establishment of the project through agreeing on and signing the contract and appointing a manager to the project. The first step involves deciding whether the particular invitation is even worth considering. The second step involves determining the risks inherent in the envisaged project and provides information for setting up the price offered to the client organisation for the development of the product. The final step relates to deciding on how to minimise these envisaged risks by providing for them within the contract and the agreement with the client. What is involved in these steps is described in the following sections.

3.1. Deciding on project attractiveness

The essential first step in the project acceptance decision making process involves deciding, at policy level within the contractor organisation, on whether or not to give consideration to the project at all. Such a decision is typically made by taking into account issues that matter from the viewpoint of a partner or senior consultant of the contractor organisation. It entails taking an informal general view into the client organisation to answer the basic question: ***might we want to do business with this potential client?*** and into the wider contractor organisation to answer the basic question: ***might this business be beneficial to our organisation overall?***

It is difficult to answer these two questions with accuracy. The potential client may not be known, the benefits to the contractor organisation may be difficult to define. But they are important questions to ask even if the answer is ***tentatively*** "Yes" to both of them in the absence of any more specific information. The often informal nature of this early but critical decision making process is important because it sets the atmosphere in which the later stages of the project are played out. An enthusiastic, headlong rush into an unconsidered course of action at this early stage can lead to the risk of haphazard negotiations and pitfalls later on.

Usually, there is little, if any, hard information on which to base this decision; thus, simple heuristics tend to be employed to determine the answers to these basic questions. Parkin (1983, pages 12-13) identifies some typical ones relating to direct subjective forecasts of effects on funds, ability of management to achieve the ends they have perceived through the project, global risk of project failure, timespan for development, return on investment, congruence with software development policies, and congruence with in-house system architecture.

3.2. Determining project risks: deciding whether the project warrants contractual negotiations

Having decided to give the project consideration, decision making about the possibility for actually accepting the project moves on to the next step, that of determining whether the project justifies contractual negotiations. Senior management within the contractor organisation now have to set about determining more precisely the nature of the client's requirements and the risks inherent in the project both from a managerial point of view (e.g., what demands on management the project will make) and a technical point of view (e.g., technical skills needed, technical feasibility of the project).

For this purpose, an initial proposal must be developed and costed for a project which will deliver a software product to meet the requirements of the client organisation as perceived through the invitation to tender and, often, through informal contacts with the client. In this process, again, senior management members are involved, all conmbuting their corporate experience to the

exercise of making initial cost, duration and quality estimates, for internal use, and developing a price for the project to present to the client.

The offered price is usually considered to be the sum of three components: (a) the estimated cost of the project (assuming all goes well); (b) a risk (or contingency) budget, available for spending on contingencies which may arise due to the materialisation of risks inherent in the project; and, (c) a profit component (Cooper & Chapman 1987).

Current cost estimation techniques, whether based on COCOMO factors (Boehm 1981) or function-point analysis (Albrecht 1979, Albrecht & Gaffney 1983) or component measurement (DeMarco 1982), essentially focus on certain variables, the most important ones being software size and complexity (Brooks 1982, Boehm 1981), expected productivity of project personnel, and required quality of the product (Sneed 1989). Consideration may also be given to the time available to the project and by how far it can be stretched or compressed (Boehm 1981, Putnam 1978), staff levels required (Conte, Dunsmore & Shen, 1986), and the environment within which the project has to take place (hardware, software, organisational issues). These cost estimation techniques are, generally, insensitive to risks which could result in system redesign, re-implementation, enforced human and technical resource substitution, and so on, despite the fact that these risks are one of the major determinants of why so many software development projects run into great trouble. Instead, using these techniques to estimate the cost of the project usually involves estimating a separate risk budget to pay for the costs that may be involved in handling such contingencies.

It is essential that the estimate of the size of the risk budget is a reasonable one as an overestimate is likely to lead to the rejection of the proposal by the client (frequently in favour of one made by a competitor). An underestimate, on the other hand, although it may win the contract for the organisation, may also reduce the profitability of the project to a dangerous degree or provide the project manager and his team(s) with impossible targets to meet (Keider 1979).

In estimating the size of the risk budget, it is a common practice to use the rule of thumb of attaching a given percentage of additional cost (e.g., 10% or 15%) to the estimated cost of the project depending, for example, on the kind of project, the type of client organisation (e.g., governmental agency, private company) and what the client will be prepared to accept as a reasonable risk budget (if any), and the experience the company has had with similar projects in the past. This practice, however, although it reduces the time spent in preparing the project proposal, provides no feedback concerning the actual nature of the risks involved in the project or on how to counteract them.

An alternative, more promising, solution is to employ risk analysis tools or risk analysts with the aim of assessing, in quantitative terms, the risks inherent in the specific project. The results of this analysis are then used to set up the risk budget for the project. The risk analyses carried out at this time are usually based on the assessment of ***risk drivers*** in terms of their degree of impact on the factors which are taken as indicators of the riskiness of the project. A ***risk***

driver is an observable phenomenon which is likely to **drive** up the **possibility** of some risked consequence whose **future occurrence** depends, in part at least, on the occurrence of this phenomenon (Berkeley, **Humphreys** & **Thomas** 1990).

This approach to risk modelling has been called the *blackbox* approach. as it concentrates on risk assessments which do not rely upon a **structured** view of the inside workings of the project itself but "on the values of situational or product characteristics **(i.e.,** risk drivers) which, individually, or in combination. are believed to **contribute** to the probability of undesirable outcomes" **(Moynihan, McCluskey** & **Verbruggen** 1989, p. **3).** The alternative, ***whitebox*****,** approach, often promoted in texts on risk analysis **(e.g.,** Cooper & Chapman **1987),** involves **building** a **workbreakdown** for the project, showing the various tasks to be carried out, and the dependencies between them. For each task, outcomes which would indicate that the task had gone wrong **are** explicitly identified, together with their estimated probabilities and costs. **A** bottom-up analysis is then made to provide the total expected cost distribution associated with "things going wrong". This distribution forms the basis for semng the risk budget

There are advantages and disadvantages associated with both these methods. The **blackbox** approach, as a method which is not project-specific, enables the creation of a ***generic*** risk assessment **structure** by senior management (to be used in the way that we describe in section **3.2.3** below) which incorporates company policy and corporate experience. However, for the very same reason, it provides information about project risks only in terms of the domains **(e.g.,** expected problems with the client) in which they **are** to be expected rather than detailing these risks.

The **whitebox** approach has the advantage of enabling simulation through the project model it has built, thus providing for a better understanding of project-specific risk areas. However, the usefulness of this approach is dependent on the validity of the project model it incorporates which, at this stage of knowledge about the project, is usually based on ***guess-work*** by the persons involved in actually building the model. Thus, although the potential benefits of a **whitebox** analysis may be great if this project model provides a correct view on the project as it will eventually evolve, its potential disbenefits if this model is wrong are even greater as the consequence **will** be an incorrect estimation of the difficulties involved in the project. Furthermore, the **whitebox** approach is ruled out at the early stage of establishing a project, as the requirements, resources, and constraints **are** still far too fluid for the development of a detailed workbreakdown (which is essential for this approach) to approximate the real **workbreakdown** for the project when actually implemented. Thus, in this chapter, we concentrate on what is involved in taking the **blackbox** approach.

3.2.1. Project-specific factors affecting the riskiness of the project

Three complex variables are frequently examined *as critical success factors* when assessing the risk of a particular software development project (Cash, McFarlan & McKeeney 1983). These are: project size, familiarity and experience with the required technology, and knowledge of the content area of the project.

Project size† may be assessed in terms of the total costs of development, the staffing requirements, the number of people in the organisation affected by the system, and the time it takes to develop the system. Jenkins and Wetherbe (1984) discovered a general agreement between 23 different system development organisations concerning the definition of project size (large or small) which was independent of the size of the organisation. However, in examining the *risks* inherent in the particular project, project size is an important variable to assess *in comparison with* previous projects completed within the particular organisation, as a large project taken on by an organisation that routinely develops large projects holds far less risk than if the same project were to be developed by an organisation which had never been involved with a similar sized project before.

Familiarity and experience with the technology required for the development of the software product is also an important factor to consider as lack of expertise in the technology (hardware and software) considerably increases the uncertainty of project schedules and project outputs (Boehm 1981). This is also a relative attribute of the project since it depends upon the experience of the people assigned to the project: what is "high" technology to one p u p of developers using a certain set of tools for the first time is "low" technology to others who have mastered the same set of tools. However, most contracting organisations with a track record in software development become "naturally" geared towards the type of project which figures most attractively in their own record as these present little risk. If other types of projects are ventured, they will require strong dedication from the entire project team and strong support from the contractor organisation if they are to succeed.

The existence and degree of *knowledge of the content area* of the project is also important to consider as, if it exists within the project, it leads to structured projects where specifications are well-defined and fixed *a priori*, the nature of the final deliverables is known or easily determined and not subject to much change. Unstructured projects, in contrast, are defined through negotiation and consensus, and the pressures for changing specifications within such projects are great, thus increasing project risks.

---

† See Keen (1987) for a useful guideline of how large or small a project is in terms of total costs, man-years' effort, duration, nodes of interaction, number of people involved, number of program statements. resources, number of activities, number of programs, data capture demands.

These project-specific factors can be assessed on their own in determining whether or not even to proceed with the proposal. For example, it will be inherently risky and, thus, probably unwise to push for the acceptance of a project which is unfamiliarly large for the organisation and the required technology and content area of which is new or unknown. However, in reality, it is never the case that such issues are that clear-cut. The project may be larger than usual but one may know the required technology quite well; the required technology may be only slightly different to the one the organisation is used to and one can gain knowledge about the content area easily; and so on. Therefore, although considering these factors and how the project scores against them may be a useful device for determining which projects the organisation should ***not*** venture into, a great deal of more information is needed to determine the risks of the project when the project does not score negatively on all of them. This is the place where the importance of risk drivers materialises.

3.22. Identifying risk drivers

As stated earlier, a risk driver is an observable phenomenon which is likely to increase the probability of occurrence of some risked consequence (e.g., the project failure in terms of cost, meeting its requirements). Discovering what constitutes a risk driver and which risk drivers are particularly important to watch for within the project involves considering many, diversely located, sources of risk that could potentially affect a particular project if it were to be established.

In the previous chapter, we described the different stakeholders who may affect the fate of the project as originating in: (a) the client organisation, (b) the contractor organisation, (c) other external agencies (e.g., regulatory agencies, existing partners whose interest may be affected, clients at large), (d) potential suppliers and subcontractors, and, (e) the potential project team(s). Information about sources of risk drivers can be gained through exploring the worlds of these stakeholders. As these explorations address different worlds, they provide different kinds of information to the decision making process essential to its successful conclusion. They also, often, provide indications about how to proceed once the decision to establish the project has been taken ***so*** that any risks encountered during these explorations may be minimised (e.g., concerning particular management techniques that need to be employed, managerial skills that may be required). Below, we consider what each of these explorations addresses and the kind of information each can provide.

(a) The exploration into ***the client's world*** starts with information provided by the client on project requirements and provides information on client-related issues which may affect the project's progress and outcome and which are not readily available from the project specifications themselves. Information that can be gained through this exploration concerns, for example, the quality of already existing or past relations with the client, the track record of the client in the project area, other competitors for the

contract, the background to the invitation to tender, the extent to which the client knows what he wants and his experience in the application area.

(b) The exploration into ***the world of the contractor organisation*** typically starts with the organisation's guidelines for project development. These tend to be idiosyncratic to the contractor organisation and constitute company policy covering project development. Information which may be gained through this exploration relates to resources and know-how currently available within the organisation, which managers should be consulted given the size and type of the project, the kind of links that must be set up with existing software systems, added benefits which this enterprise could provide to the organisation, financial constraints, and so on. Other issues that need to be considered relate to company policy and objectives, other currently considered or running projects that may need to compete with the current project for resources and priority within the organisation, and such like.

(c) The exploration into ***the worlds of other external stakeholders*** is very important in cases when these have a regulatory role in relation to the conduct of the project and the characteristics of its products (for example, when medical or military agencies are involved). The regulatory requirements of such agencies may be clearly written down but, at this stage, it is the responsibility of the contractor organisation to identify ***which*** regulatory agencies are involved. Information which can be gained through this exploration is intended to identify the full set of regulatory requirements which may apply to the project and its deliverables.

(d) The exploration into ***the worlds of potential suppliers and subcontractors*** enables one to investigate the extent to which resources which are expected to be provided to the project by suppliers and subcontractors are likely to be actually available at the time required and to conform to their specifications. This information is needed when developing the project schedule and examining its sensitivity to problems which may originate from suppliers and subcontractors. A great number of problems besiege projects when the required hardware has not been delivered at the expected time or when subcontractors do not follow similar standards or disciplines to those of the contractor organisation (Druding 1984). In cases where the client has preferences between suppliers and/or subcontractors and he expects the contractor organisation to recognise them (either informally or formally), information gained through exploration (a) has to be considered within this exploration as well.

(e) Finally, the exploration into ***the world of the potential project team*** is important in anticipating the effects on the future project personnel of the changes involved in being transferred to work within the project as envisaged. Projects impose their own requirements on personnel involved in them and these need to be taken explicitly into account (e.g., White 1988). Information gained through this exploration can answer questions concerning how team building may be achieved, whether the project work is unfamiliar and may generate new stresses and training requirements for

team members, whether individual team members will find new benefits or difficulties in working in the particular groups envisaged within the project team(s), and so forth. Information on these issues is usually obtained by looking into the *past,* finding projects whose work options exhibited similar features as that now under consideration. and examining the records (which may often exist as anecdotes and stories) of personnel reactions and development within them Where the identity of personnel proposed for the new project is known, then their own performance on past projects, and, where they have worked together, the strength and weakness of their interpersonal relations during the previous work may also be relevant (Keider 1979). Over-reliance on specific historical analogies, however, can be dangerous as it risks discounting *subsequent* changes in team members' personal skills, preferences and relationships.

These explorations provide information for constructing scenarios looking into the future. and identifying risk drivers which may adversely affect the project. In theory. if a set of risk drivers can be identified for a particular project which drivers, collectively, cover the range of sources of risks to project objectives, then the manager who will be eventually assigned responsibility for the project will be in a position to anticipate potential risks by observing the *actual* levels of the risk drivers as the project progresses and to use his risk budget to handle the contingencies as they arise.

In practice, this may not be so straightforward, as risk drivers are initially identified only through hindsight: there is no guarantee *a priori* that one can always identify a concise set of risk drivers which are exactly appropriate for anticipating the risks which may actually materialise in a particular project. The remedy of trying to observe *everything.* in case it may be a risk driver, is doomed to failure: the time taken would be enormous, the amount of intrusion on the worlds to be explored would be unacceptable and the resulting plethora of information would be impossible to interpret in terms of its overall impact on project risk (Glahn & Borg 1988). Many of the real risk drivers would be missed at this stage anyway, as they will be dependent on the detailed plans and system design produced later in the project's life cycle (Telfer 1989).

Hence, the major initial goals for a *risk analysis* is to define a limited set of key risk drivers which apply for a particular project (according to its type. development environment, stage of development, etc.), and to assess their relative impact on risked consequences (that is, failing a, satisfy project objectives). particularly those that will jeopardise the achievement of the technical, financial and timescale targets for the project. This allows for the development of the *project risk profile* (Cash, McFarlan & McKeeney 1983). This is a multi-factor index of project risk which can be compared across projects and which reflects the degree of risk that the project will result in disbenefits for the contractor organisation across the range of domains involved in conducting its business.

3.2.3. Generic risk assessment models

Senior managers who are responsible for the decision of whether or not to accept the project, when left to their own devices, may prioritise different aspects of what is considered within any of the explorations described in the previous section depending on their different individual responsibilities, concerns and expertise. This could result in basing their judgements about the riskiness of the project on different sets of factors indicating the potential risks to be run and benefits to be gained through taking the project, or, at least, in assigning different importance to particular factors in comparison to other factors. The result could be that different decisions are being made and different recommendations being put forward by different decision makers. Group decision making *can* resolve some of these differences by considering what is being assessed within each exploration and analysing the assumptions different persons make within it (Mason & Mitroff 1981. Humphreys & Berkeley 1987). This can be constructive as the collective experience of the group is utilised to the full.

However, organisations habitually tend to avoid such confrontations as they can be very time-consuming, often necessitating the involvement of external consultants and, thus, increasing the cost of the preparation of a proposal for the project. Instead, at the early stage of project consideration, organisations tend to rely on the use of already developed in-house project risk assessment models which are available within the organisation as *generic* structures to *be* interpreted in terms of the new potential project. The generic structure that an organisation favours, and habitually uses, will usually have been originally developed at policy level within the organisation, capitalising on the collective experience of senior managers. This structure is then fixed and conveyed wholesale to those personnel in the organisation with executive responsibility for negotiating. accepting and managing specific projects, together with smct instructions on its use in assessing individual projects. The structure is *fixed* in that all risk factors that may affect any project are set; they have only to review them and provide their values concerning the particular project. The use of a project risk assessment model incorporating a *fixed* structure facilitates comparison across projects and provides for a consistent interpretation of the organisation's policies on the importance and acceptability of various types of risk.

Many organisations who regularly use such procedures consider that the construction of a generic risk assessment model, carefully tuned to reflect their policies and operating conditions, gives them a major competitive advantage in deciding how to price and negotiate project contracts. For this reason, the proprietary information in the project risk assessment generic model is usually kept strictly confidential within the organisation.

3.3. Managing the risks within the contract negotiation process

Allowing for risks solely by employing risk analysis assigns, by default, an entirely passive role to the decision maker who is considering the potential project in the face of the identified risks. This is because, in a conventional risk analysis, the dependency linkage between risk drivers (the ***independent variables*** in the analysis) and the consequential risks being run in the project (the ***dependent variables*** in the analysis) is examined without considering the potential intermediary effects of managerial actions. These actions might actually create some of these dependency links, thus reducing the actual risks run in the project significantly below the level indicated in the risk analysis.

However, effective risk management on projects under contractual negotiations requires that one is able both to ***anticipate*** the risks to the project and to ***design*** suitable organisational structures within the project to minimise their negative practical impacts if and when they materialise. To this purpose, information gained through the explorations described in section 3.2.2 above and which was used in developing the project risk profile now needs to be taken into account with a view to determining how any of the identified risks may be minimised by making provisions for their occurrence within the contract and/or deciding on strategies about how they may be counteracted if and when they occur. This entails focusing on the project and its needs ***given*** the risks that have been discovered (rather than focusing on characteristics of these worlds as defined through their explorations for the purpose of determining potential sources of risk for the project).

A principal managerial objective at this stage of the project acceptance process is to be able to use the characterisation of the project on risk indicating factors (i.e., the project risk profile produced by risk analysis) to uncover what needs to be taken into account in determining particular ***management*** techniques to minimise the risk inherent in the project. For example, in the case where the organisation has not been involved with the particular client in the past or when difficulties are expected from the client according to previous encounters with the client, the appointment of a very skillful project manager will be an essential part of senior management strategy.

Another managerial objective is to seek out, review and interpret key information in the project environment that is necessary to ensure the smooth running and successful outcome of the project. In support of this objective, information which is useful to review and interpret at this stage concerns, among other things:

- the requirements to be met by the project;
- the functions to be achieved by the project and their interrelations;
- the coherence of, or contradictions in, the objectives imposed on the project by different parts of its environment;
- the project's inputs, resources consumed, recipients for its outputs, and resources produced;

- pertinent characteristics of key resources likely to be needed, possibilities for resource substitution, and training needs and possibilities for human resources;
- regulations guaranteeing security and privacy; and so on.

This information may be required in order to set realistic price and timescale targets for the project, making adequate provision for the costs incurred in managing and overcoming the risks which may materialise during the running of the project. It is also important that the resource targets for the project set through the use of this information are made explicit at this stage rather than evolve later. Otherwise, senior management may unwittingly give way on these targets in the final stage of contract negotiations, thus increasing the risk of severe planning and management difficulties after the project has started.

Finally, senior management will need to ascertain when, how and with whom to conduct negotiations which are likely to facilitate the project's progress and acceptability of its outcome. Early negotiations can help considerably in obviating or minimising the risks inherent in the potential project. The risk profile can be used to guide the negotiation process by indicating those specific factors on which the project exhibits a high level of risk which wuld be reduced through procedures agreed with the client (e.g., concerning product specifications, review procedures, acceptance testing, financial arrangements, preferred suppliers). A review of the identified risk drivers, at this stage, can provide guidelines to facilitate these negotiations which may cover, for example:

- identifying potential conflicts in the objectives and structures required for the project and finding ways to resolve them;
- negotiating mutually acceptable deadlines and milestones;
- negotiating a shared understanding of contractual terms;
- clearing ambiguities with the client or other involved parties; and so on.

This information can be used in negotiating a contract which is acceptable to both client and contractor. A well-negotiated contract can considerably reduce the risk inherent in a project and refine the scope of managing the residual risk during the execution of the project. In negotiations on project risk reductions, the contractor and client organisations are not adversaries: they share a common interest as the prospect of negotiating a project where risk is shown to be well-contained and manageable will be attractive to both parties.

3.4. Defining project management responsibilities

Signing a contract imposes a specific set of responsibilities, both formal (as detailed in the contract) and informal, on both client and contractor organisation. The importance of a clear and unambiguous contract can never be overemphasised (Jones 1983); still most contracts lack such characteristics and this can cause considerable uncertainty and difficulty in the interpretation of the requirements for the project

The role of the project manager is established and becomes operational at this point Some of the responsibilities of this role are set out in the contract and other documents regulating the project. Others are implied within his role in order to transform project requirements into project deliverables which meet their defined specifications. It is wise that the manager's responsibilities be negotiated at the same time as the contract is negotiated (rather than later, when contractual terms are difficult to alter) with the client, the potential project manager, and with other managers withii the wider contractor organisation who are involved in transactions with the project. The boundary of the project manager's responsibilities should be defined on those activities which constitute the transactions which interface the project with its environment

The two principal project interfaces for such transactions are:

(1) The ***interface with the client organisation,*** regulated in part through explicit conditions formalised within the contract, in part through interpretation of less formally defined client requirements and through the production of deliverables. Informal contacts with the client throughout the life of the project are likely to be of great value in clarifying client expectations and re-negotiating the overall agreements.

(2) The ***interface with the contractor organisation*** which tends to be less well-regulated than that with the client. For example, project resource requirements will still have to be negotiated within the contractor organisation but the results of these negotiations may simply be vague promises of future availability rather than being backed up by any interdepartmental contract They are, therefore, subject to change when senior management makes trade-offs between the needs of different projects competing for resources.

Depending on the project, it may also be necessary to establish additional project interfaces through transactions with other external stakeholders (regulatory agencies, etc.) as perceived during exploration (c), and with suppliers and subcontractors as perceived during exploration (d) (described in section 32.2. above). Transactions with suppliers and subcontractors should be regulated in the same way as those with the client, the client in this case being the contractor organisation to whom the suppliers will undertake to provide the necessary supplies to the project and to whom the subcontractors will undertake to provide the required services.

All these transactions fall within the purview of the project manager. The success with which they are carried out at each point in time they are required is vital to the success of the project.

Chapter 4

The project management process

> *"The charge to a project manager is not to perform particular duties, but to bring about a desired result. Accordingly, anything that may bear on the result falls within the project manager's purview."*
> *(Gustafson 1984, p. 298)*

Our discussion in chapter 3 of the various steps involved from deciding on the response to an invitation to tender for a software development project to the signing of the contract was aimed at outlining what issues should be looked at from within the contractor organisation at each of these steps in order to reach the degree of knowledge necessary to make rational informed decisions or enter into negotiations with the client with the necessary strategies worked out. Otherwise, the chances are that the contract that will be ultimately signed will specify impossible conditions for the project and its management.

In a competitive market, however, underbidding in order to gain the contract with the client is a common occurrence under the hope that, eventually, once the contract has been signed, further negotiations with the client may improve the conditions for the project. Alternatively. if this appears to be difficult or impossible, ways may be found to cut comers during development (e.g., reduce functionality of the system, perform less testing thus reducing its quality) to enable the project to make a profit or, at least, not result in a financial loss for the contractor. Sometimes, however, the fact that the project offers work for otherwise under-employed resources may make up for a loss-making contract. On the other hand, an organisation may intentionally underbid for a contract for the purpose of gaining the current project at a loss but with the view to future engagements for work with the client organisation that the current project will bring about. In the latter case, the project may not suffer much from underbudgeting as the organisation may make available to it additional, internal, funds to ensure that the quality of the software system developed is high enough for the client to guarantee future contracts.

Whatever the quality of organisational decision making or of the results of client-contractor negotiations during the project establishment stage, the

contract that has been signed at the conclusion of this stage sets up three sets of constraints for the project as a whole:

(1) constraints for the ***software system to be developed*** in the form of specifications it must meet;

(2) constraints for the ***project work*** in the form of required characteristics for deliverables, development standards to be followed, deadlines, budgets, etc.; and,

(3) constraints for the ***project manager*** which directly ***affect*** his actions in managing the project (requirements for reporting, monitoring the project work system, overcoming threatened cost and time overruns, acting to ensure deliverables meet specifications, and so on).

The manager of the project inherits all these three sets of constraints and he is instructed to act at his discretion in order to meet them as well as meeting the objectives of his organisation (e.g., make a profit or avoid a loss, gain prestige, retain or increase reputation). This chapter presents what is involved in the management of a software development project, that is, how the manager exercises his discretion to that purpose. We start by describing the various roles the manager is expected to, and does, play in the project and the functions he has to fulfil in managing it. Then, we proceed to discuss the project management phases in the project lifetime from the time the project manager takes over the responsibility for the project to the time when he has to close the project and we describe what activities each phase entails. We conclude with a detailed discussion of how a manager can increase the chances of his project being successful through anticipating possible threats to this success and planning and managing his project in a way that they may be preempted before they occur, or so that their effects, if and when they occur, may be counteracted.

4.1. The role of the project manager

Typically, the role of a project manager tends to be defined in terms of the functions he has to fulfil in the project, which are, often, unlike the functions other members of the project team have to carry out. Thus, the focus in discussions of his role tends to lie on the difference between his role and the role of others project members.

In general, there is an agreement about what functions he has to fulfil in the project. However, these functions are not always referred to by the same name, nor is the same name always used to describe the same function in the same way. For example, Brech (1967) defines the four fundamental elements of management as planning, coordination, motivation and control; Herbert (1976) describes the managerial functions as planning, organising, directing and controlling; Turner (1984) defines the classic duties of a project manager as planning, organising, motivating, controlling and carrying out a postmortem on the project; Thayer (1988) defines planning, organising, staffing, diiting and controlling as the elements of the classic management model. Each account

proceeds to detail what achieving each of these functions entails and how each is achieved. Correspondingly, the manager is described in his role as a planner, director, controller, and so on.

Such approaches are based on an attempt to understand and discuss project management in relation to functions the manager needs to fulfil for the project so that it can be successfully carried out. However, as we discussed in chapter 2, this is only part of the story. The project manager is located at the interface of the project work system with the project environment. His success is dependent on his ability to transact with the project environment well as well as to manage the project work system well (that is, carrying out all the managerial functions successfully). Any attempt to define the role of the project manager purely in terms of project management functions such as the ones listed above will not be able to address this aspect appropriately (if at all).

Typically, books on project management or management in general address this aspect of managerial work under a discussion of the skills that a manager should have. Thus, a discussion of managerial functions provides a view on what the manager *has to do,* and a discussion of skills that he generally needs provides a view of managerial characteristics which, if present, can help in carrying out these functions. For example, Keen (1987), in describing general management qualities, identifies skills that the manager could have as being able to shield his staff from pressures, to delegate, to gain respect from staff, to work effectively, to communicate with others, to show maturity, to have a capacity for toughness, to be realistic and committed. Additional characteristics which Keen (1987) proposes as desirable for the project manager to possess as an individual include determination, flexibility and adaptability; being a negotiator, planner and administrator; and an ability to use psychology to understand his staff's needs. However, the notion of *skill* assigns agency to the manager as a person (i.e., he has this skill or not) rather than a requirement to any particular role he has to play by virtue of his position between the project and its environment.

Mintzberg (1975) described three different types of roles the manager has to play, which, collectively, form an integrated whole. These are:

(a) *interpersonal* roles such as being a figurehead, a leader and a liaison;

(b) *information* roles such as being a monitor, a disseminator and a spokesman; and,

(c) *decision* roles such as being an entrepreneur, a disturbance handler, a resource allocator and a negotiator.

All these roles relate to the transactional aspects of project management both in relation to the project and in relation to its environment. Each role is guided by a different goal and it is defined differently according to the person towards whom it is played (e.g., disseminating information to the project environment is different to disseminating information to the project work system). Playing a particular role will generate its own information needs; the process by which various roles are played and the activities they entail are different. The outcomes of these activities will also affect the project in different ways.

Focusing on roles rather than skills enables also the definition and understanding of role conflicts which often materialise in carrying out managerial functions. For example, in planning, the manager may experience a conflict between his role as a leader in making the plan and his role as a negotiator in making sure that the created plan takes into account information from his staff and their needs. The final result should reflect the resolution of this conflict if the plan is to be an agreed plan and, hence, provide a good basis for the project work.

Within this understanding, skills are seen as a subsidiary to a role: the possession by the manager of a particular skill may be specified as requirement for a particular role to be played out successfully. The ability to play the role, rather than possessing the skill itself, now becomes the focal point in evaluating a project manager. Not all projects necessitate that the manager has to play ***all*** the roles identified above successfully; some may not even be required by the project while others may be vital. For example, as we discussed in section 3.3, the anticipation that a project will face problems with the prospective client organisation necessitates the engagement of a project manager who has, above all, the necessary skills called for by the negotiator role.

4.2. Project management phases

In books on software development, it is customary to discuss what is involved in the development process in terms of the successive phases it goes through (e.g., specifications, design, coding, testing) while the management of the project tends to be discussed in terms of the general functions the manager needs to fulfil (e.g., planning) or in terms of general activities he needs to carry out to fulfil these functions (e.g., estimating). Good project management, after all, entails vigilance at all times and timely response to any emerging needs of the evolving project. Thus, it is assumed that, if the need arises, the manager will know which is the appropriate function to carry out.

However, such approaches fall short in specifying differences between similar functions being carried out at different points in time during the development of the project. Moreover, by focusing on managerial functions, one tends to miss out activities which are carried out either outside the concerns of particular functions or within the concerns of more than one function. Thus, this section discusses the project management process as it evolves through time, that is, it discusses ***project management phases*** and what each entails, and describes project management functions as they are played out within each one of them. Different types of causes for eventual project failure can be found at each of these phases, the results of which perpetuate themselves throughout the project (Keider 1979).

4.2.1. Taking over the project

In accepting the responsibility of managing a project, the manager also accepts the responsibility for the commitments already made for the project and the responsibility to act to ensure that these commitments can be maintained (or, in the worst case, to provide early warning that they cannot). However, he does not have any guarantee that he is going to be able to meet them. It is in his best interest to check for himself that these commitments are in fact achievable, or at least realistic, within the bounds of the resources available (Keen 1987). Of course, if the project manager was also involved previously in the stage of project establishment, this would give him the advantage of having contributed to the generation of these commitments and, possibly, in the brief of the project which is now handed over to him for management.

Although some initial estimation of the project's resource needs will have been made before the project manager takes up his responsibilities (e.g., in order to estimate demands on the contractor organisation and project cost to specification within the contract), in a reexamination of the project, the project manager may uncover new aspects of resource demands and opportunities, as he will be looking more closely this time at its technical and activity aspects than was possible earlier on. Moreover, the project itself may have changed since its inception without a corresponding change being made in these initial estimates (Keider 1979). Through performing such a reexamination, the project manager can gain new insights into his project and, hopefully, increased confidence in its feasibility under his management. He may also discover informal communication channels which will be important in later stages of the project for information gathering and dissemination.

A re-examination of the project is likely to be the first activity of the newly established project manager in order to establish that the ground on which he is standing is firm, or, at least, that he knows he has problems, and act accordingly. This involves orientating the real world of the project work within the constraints, requirements and objectives for the project Bridges have to be built to the world of the project environment where these constraints, requirements and objectives may subsequently change as new conditions and knowledge arise within the client and contractor organisations. Thus, the manager may need to review and gather information about, for example,

- formal conditions defined in the contract;
- the client's requirements and expectations concerning the deliverables;
- informal agreements made with the client as understood by the client and the contractor organisation;
- the technical aspects of the project as described in the contract and by senior technical staff;
- the results of any project-related activities that had been carried out earlier (e.g., during the feasibility study);

- required secondary activities (e.g., reporting, review, etc.) defined by the contract or company procedures;
- resource requirements and potential resource availability;
- estimates of effort and costs involved in the execution of this type of project by using his own experience and corporate memory; and so on.

In the light of this information, and any further information which has been gained during the project establishment stage, the project manager can now

- re-analyse the risks and requirements for the project,
- negotiate any necessary changes,
- set up liaison with other organisational departments and external stakeholders (e.g., client, suppliers, etc.),
- make a coarse plan for his project in order to find out the amount of resources that may be needed and how the project objectives may be achieved,
- negotiate resources with his own organisation and with the client organisation,
- negotiate working conditions for his team(s), and,
- decide on how he is going to manage the envisaged project work system.

The contractor organisation's recommended or required project management practices may guide the manager's decision on how to manage his project These practices address the management methodology, standards, templates and estimation models to be employed in a particular project. These may be company-wide management requirements, or may have been developed specifically for a particular type of project which is similar to the one under consideration. Sometimes, clients or regulatory agencies impose their own requirements for methodologies, standards, etc. to be followed in connection with the management of a particular project. Their desires usually tend to take precedence over those of the contractor organisation.

Management methodologies are often presented in textbooks on management as a fully specified set of prescriptions concerning the sequence in which defined activities are carried out in managing a project. Possible branches in the activity sequences are also pre-specified and the conditions for branching or repeating an activity before continuing are clearly and exhaustively given.

However, methodologies which are successfully operated in actual practice in guiding software development management are rarely so cut-and-dried. The project situation and its environment in general is too fluid and complex for such an approach to make sense for all possible management actions, and their sequencing, to be pre-specified with any hope of effective and successful application. Instead, in reality, much is left to the manager's own discretion in formulating and sequencing his management activities in planning and controlling the project, so that he can take advantage of opportunities and respond flexibly to contingencies as they arise.

The practical aim of such methodologies is thus not to prescribe what the manager ***must*** do at any time, but to guide and constrain him in observing good management practice, by not launching activities at inappropriate times or in inappropriate ways, given the management requirements for the project and the results of the activities that have been carried out so far. In general, a methodology tends to be viewed as a ***partial set of constraints,*** which shapes the manager's activities at various levels from the most global ones (e.g., "you must create a coherent workbreakdown before assigning resources to tasks") to the quite specific (e.g., "actual travel and subsistence expenses must be tracked monthly against the values allowed for them in the budget, and a specific report must be made concerning any overspending").

Standards, templates and estimation models are sometimes considered as integral parts of management methodologies (Sneed 1989), but here we view them separately as they provide a partial set of constraints which shapes the ***how,*** rather than the ***what and when*** of actual project management activities. For instance, the ***standards*** specified for the project may regulate and constrain the way a task decomposition (workbreakdown) is made, how and when reporting on progress is done, how costs should be monitored, what must be verified before deliverables are signed off as completed, how documents should be prepared and catalogued, and so on. ***Templates*** may be specified to provide standard patterns for workbreakdown structures, or to provide profiles of skills or characteristics for the resources needed to carry out particular types of task, and so on. The ***estimation models*** specified may cover effort, duration, cost, quality and/or risk estimates. In each case, they specify the input variables whose values are to be observed, the conditions under which they are to be observed (the domain of applicability of the model) and the rules or algorithms to be used in computing the estimates which are output from the model concerning the quantity of interest, given the assessments which were input to the model.

The purpose of project management activities carried out at takeover time is to enable the manager to understand his project and its constraints (to make the project effectively ***his*** project), to set it up as an entity within its environment and to plan his own managerial activities.

4.2.2. Initial planning of the project work system

While taking over the project meant the official acceptance of project responsibility by the manager, planning is the first implementation of this responsibility. This, however, is not an easy task. A good and viable plan may help the project no end but the existence of a good plan does not necessarily imply that the project will be carried out successfully.

A project plan effectively describes work that needs to be carried out to meet the objectives of the project and how this is to be achieved. It typically consists of

- a workbreakdown (or task decomposition), specifying the tasks that need to be carried out within the project,
- effort estimates required per entry in the workbreakdown,
- number and types (or names) of resources per entry,
- a time schedule,
- a list of deliverables and milestones.

However, there is more than one mode of planning used in a project and more than one type of plans created for it. We describe these in sections 4.2.2.1 and 4.2.2.2, below.

The planning phase also includes additional managerial activities the motivation of which is to set up the scene within which the project will take place (Cooper 1984). For example, the project's history will need to be documented and the interface with the client will need to be specified. The latter will define the facilities, services, contributions, etc. the user/client is expected to make available to the project as well as the change control procedures which will provide the procedures, constraints to, and possibilities for any future change to the client's requirements. The manager also needs to define the constraints under which the software development itself will be carried out. This is likely to involve describing the standards and methods that will be used for the development of the software system and the software quality assurance procedures to be employed.

4.2.2.1. Modes of planning

The mode of planning used in any phase of the project is determined by the amount of knowledge and certainty the manager has concerning the resources that have been or will be made available to the project. Thus, *standard* planning is the mode of planning employed when this knowledge is minimal and uncertainty about available resources is high, while *actual* planning is the mode of planning usually employed after all or most of the resources that will be made available to the project are known.

Standard planning is based on the assumption that standard people are going to be assigned to the project, that they have the standard required capabilities and that they have standard records of absence (sickness and holidays) and characteristics (e.g., learning curves, capabilities, affinities). Standard planning requires prior definition of resources (i.e., how they should be characterised) and of the contractor organisation's calendar throughout the currency of the project. It also requires consideration of how to set margins on estimated task effort which allow the possible deviation of any given resource from the standard measure.

When resources are made known to the manager, the results of standard planning need to be reviewed, re-interpreted, and revised according to the constraints and possibilities offered to the project through the resources actually made available. Now, *actual* planning becomes operational. While in standard

planning, tasks are allocated to resources until all tasks are accounted for, in *actual* planning, resources are allocated to tasks until all resources (which, as defined, meet the requirements for the task to which they are allocated) are used up. Typically, this latter allocation process exhausts before the former is satisfied. Now the workbreakdown structure, scheduling, and resource allocation components of planning are iterated with the objective of finding a workbreakdown structure which can be scheduled in such a way that the resources with the appropriate characteristics are likely to be available in reality when required. If this cannot be achieved, the project manager may seek to restore the coherence of his plan for the project by checking his initial estimates of effort required to carry out particular tasks, to see if these may be "trimmed", thus releasing resources for allocation elsewhere. However, if estimates on a task are trimmed, the chance of achieving the objective for that task in the scheduled time or with the required quality will diminish (Nocentini 1985).

4.2.2.2. Types of plans

While standard and actual planning refer to the process by which a plan is made, different types of plans may be produced by utilising either of these two modes of planning. It is often the case that not all resources which will be eventually made available to the project are known at the beginning of the project (i.e., at the time of the initial planning phase). In this case, a plan may include results of standard planning as well as results of actual planning. Thus, a distinction between plans should not be predicated on the mode of planning.

The first type of plan for the project is what is known as a ***macroplan*** or ***overall*** plan (Parkin 1983). The macroplan, as its title suggests, represents a global plan for the whole project from its start-date to its end-date. Sometimes, a macroplan for a project is inherited by the project manager (rather than being created by him) as a result of organisational processes which took place before he took over the project (e.g., bidding, risk analysis, contract negotiations). In this case, given that senior management or the client expects the macroplan to be followed by the project manager, it functions as an ***additional*** constraint on the manager. However, it often happens in practice and it is always desirable that the senior management involve the prospective project manager (when known) in the construction of the initial macroplan for the project. The conventional wisdom here is that, unless one commits oneself to a plan by such participation, one cannot carry it out with conviction†.

Regardless of its authorship, this macroplan guides development specifying what needs to be done by the project in global terms. It typically includes (a) a workbreakdown whose leaf nodes (activities) may be of 2-3 months' duration,

† The same argument, of course, applies for involving team members in the planning of the work for which they are responsible, at least, by collecting their own estimates of effort needed, difficulty involved, etc.

(b) global estimates of effort per activity or phase, (c) number and types of resources required at the different phases of the project, (d) general scheduling, and (e) a list of external deliverables and milestones.

At the time the macroplan is created, which resources will eventually become available to the project as a whole is (nearly always) unknown, although the manager may know the resources which will be available for the first phase of the project. Often, the macroplan is used to support the manager's application for project resources. It provides the basis for estimating how many resources and of what type will be needed by the project so that it may be carried out within budget, on time and to its specification requirements.

A macroplan may be used to look at the general direction the project should take, and, in this way, it can serve as a prospective summary of the project. Macroplans of this type, even though produced by the project manager, are generally intended for external consumption via presentations to the client, to senior management, and so on. The work that is canied out within the project necessarily needs to follow a plan specified in more detail than this: one that will enable proper monitoring and control of the project's progress. This is achieved through a ***detailed*** or ***specific*** plan (Parkin 1983).

A detailed plan is developed against the background of the macroplan by enhancing and detailing a specific section of it by breaking down the phase or a specific activity or group of activities into distinct tasks. These tasks or groups of tasks are allocated to specific resources (defined by name or type) which, by the time a detailed plan is constructed (2-3 weeks before the phase or activities are due to start), are usually known to the project manager. The detailed plan defines which part of the work will be carried out by which resource and thus provides the basis for allocation of responsibilities within the project team. A detailed plan can still be satisfactory even when resources attached to some tasks (e.g., some project support functions such as secretarial help) are only defined by type (e.g., grade) rather than by name.

Planning carried out during the initial planning phase normally produces a macroplan for the whole project (which may or may not coincide with the inherited macroplan) and a detailed plan for the first phase or phases of the project for which resources to the project are known. From then on, detailed plans for future parts of the project work are developed during the running of the project.

The time horizon within which such detailed planning is made varies. Some project managers create detailed plans of just the next phase (whatever its duration) or plan towards the next major milestone (however far into the future that may be). Others prefer to keep a constant, fixed time horizon (e.g., 2-3 months, 6 months) irrespective of how many phases this covers. The time horizon may be also dependent on the type of the project under consideration (e.g., a loosely defined project warranting a shorter time horizon) or determined by the company-wide project management methodology or a company policy.

Detailed planning within a time horizon has the advantage that, by the time a new detailed plan is made for the next slice of time, the project manager

has gained more knowledge about his project and the quality or potential of his already-existing resources. The detailed plan includes tasks that need to be carried out during its execution, their sequence and duration, resources that will be utilised, and planned internal or external deliverables. In the detailed plan, the duration of leaf tasks tends to be short (2-3 weeks to even one day). Tasks tend to be small enough to enable easy recovery from delays (typically, no more than 5 days in duration, Miller 1978). However, the detail with which a task will be defined should not be too rigid or inflexible. Otherwise, it may thwart creative effort from the person who will carry it out (Nocentini 1985).

Both macroplans and detailed plans are rather likely eventually to be subjected to revision during the running of the project upon the discovery that they are no longer adequate to guide the project or when new circumstances befalling the project necessitate re-planning. The ensuing revisions may be quite radical. But a revised plan is not a different type of plan. It simply further qualifies a macroplan or a detailed plan whether or not that was the original one. We consider re-planning under the project running management phase in more detail in sections 4.2.4 and 4.2.4.3, below.

4.2.2.3. Assessing the quality of a project plan

Both macroplan and detailed plans are defined in terms of the same kinds of entities, like tasks, resources, time, deliverables, and so on. As such, they are open to the same problems in planning. The fact that one type of plan is more detailed than the other will simply define a stricter implementation of what managers take to be essential characteristics of a good project plan. For instance, a macroplan which has most of the resources attached to tasks it comprises identified by type of resource rather than by the names of particular individuals is commonly considered acceptable. However, a detailed plan which fails to identify individual resource allocations is quite likely to be considered unacceptable both to the manager and to any other person who assesses his plan.

There are two principal types of errors that can impinge on the quality of a plan. These are:

(a) ***errors in oversight,*** that is, planning mistakes by the project manager such that the overall plan is not coherent or not internally consistent (e.g., overallocation of a resource, expected deliverables not being produced by any task), and,

(b) ***errors in foresight,*** that is, inability to foresee the possible consequences of some aspects of the planning when implemented (e.g., too many dependencies between tasks).

These two types of e m are diierent, although it is not always easy to differentiate between them and to define which type of error was the source of a particular problem with the plan.

In general, errors in oversight are typically local problems which are often easy to detect, although they may be less easy to resolve. On the contrary, errors in foresight tend to lead to more global, pattern-related problems, which may propagate their effects to other aspects of the project plan. However, something which appears to be an error in foresight when taking an external view on the project plan may only reflect anticipated difficulties in the work of project management. The latter is always constrained by the particular circumstances of the project and, especially, by the amount and quality of resources that have been made available to the manager, and this is the usual reality in project management. In other words, the project manager may not have had any alternative to making his plan that way and preparing to handle the consequences as they emerge.

Lack of contingency planning may also be seen as an instance of error in foresight. Characteristic contingencies are: temporary duties of a resource, inadequate personnel skills, transfer of personnel, late delivery of a resource, turnaround failures, software or equipment failures, lack of information required for further progress, attrition, absence, company meetings, promotion of resource, etc. However, the heuristic which may be of most help with planning for the majority of these contingencies may be expressed as "get someone above you in the organisation to OK your plan"!

Problems affecting the quality of a plan may materialise in any of the aspects which characterise that plan, whether tasks, resources, time, or estimates. However, the major difficulty in resolving such problems is that these aspects are not independent of each other. For instance, lack of understanding of the complexity of a particular task may have led to underestimation of effort needed to carry it out and, hence, to delay in completing it because of an insufficient allocation of resources.

Below, we list some potential problem areas which managers do well to watch out for in their project plans and some related heuristics which are often useful in increasing the quality of these plans. These heuristics apply not only in initial plan reviewing in the present phase but also, and sometimes especially, in reviewing operative, fully resourced, plans and contingency plans in later phases of the project. It is important to note that this list does not profess to cover all issues pertaining to the quality of a plan. It simply represents the implementation of wisdom gained through the experience of the project managers we interviewed. We list them here by the particular area of primary concern to the project manager they address.

Primarily task-related concerns

- External deliverables such as hardware and their expected date of delivery must have been defined in the plan.
- Check whether the client requires to sign off all documents before proceeding with system development in which case appropriate time should be allocated to the signing off procedures (e.g., in governmental bodies signing off may take a long time). Tasks not directly related to the

documents to be signed off should be planned to fill in the waiting gaps.

- If new equipment or new languages are going to be used in the system development, appropriate time should be allocated to learning.
- Too few or too many task dependencies are dangerous.
- Time gaps between a task's finish date and the starting date for a subsequent task which awaits a product from the first task may exist in the plan (it may be a project managerial technique of allowing for the possibility of slippage).
- Tasks specifically designed to review deliverables must exist in the plan.
- Check that all internal milestones are associated with concrete products.
- Tasks should be assigned priority. This becomes vital when two tasks may compete for the same resources at some later stage when delays have occurred.
- Under no circumstances, should more than 6 months' elapsed time be allowed without a product coming out of a task.
- Time should be allocated for handling user queries, if they are anticipated (i.e., create a task for handling user queries).
- Check if hardware has been planned w be delivered before implementation starts.
- Check which tasks may overlap and which may not.
- When tasks are put together into one new task, check the consistency in the definition of the new task with those it has replaced.
- How much can be learned from tasks already completed for similar tasks which are about to start?
- The estimated effort per task must match the effort allocated to it
- External deliverables should match those stated in the contract.
- If the project has not started at the beginning of the life cycle, are there any activities planned to check the quality of the external input from work carried out before the project started?
- What is the sequence of decisions that need to be taken before any new project task is allowed to begin?
- Check that assumptions made in making the plan are clearly articulated within the plan, also the certainties and uncertainties in parts of the plan.
- Provisions must be made for interaction between the various parts which constitute the project, that is, formal meetings with the client, technical discussions, progress meetings and quality reviews; there must also be a time margin resewed for dealing with shortcomings in specifications, difficulties in testing, etc.

Primarily resource-related concerns

- At which stage in the software development life cycle does the project start? Some projects which start at later stages in the normal development life cycle portray a resource utilisation curve which is very obscure.
- If the whole project plan needs to be compressed (due to time constraints), is the plan still viable?
- If all resources have been allocated 100% of their effective time and the project is forecasted to finish just in time, the plan is not satisfactory.
- If the system to be developed is a modification of a previously constructed system, a resource involved in the project which developed the original system will be valuable in this project
- If the system to be developed is a turn-key system, proportionately more time should be allocated to testing.
- To handle time dependencies between tasks, allocate any particular resource to tasks which can be executed sequentially. (However, this may cause extra dependency problems due to dependency on the same resource.)
- Resources provided by the client tend to perform less well than expected, hence, they should not be assigned to tasks on the critical path.
- It is common to estimate total effort available for a resource (when fully employed) by calculating 4 effective days per person per week, or 145-150 hours per month, or 60-80% of the resource's real time.
- Junior programmers should be assigned to less critical activities and be allowed more time to carry them out
- At the testing phase of the project, both analysts' time and programmers' time should be kept in reserve to cope with any changes.
- The resource curve may show a dip at the time of user acceptance trials as when the user needs to prepare the trials.
- Check if tasks assigned to one person have internal coherence.
- If particular tasks are new and difficult, check if they have been assigned to a senior person.
- If a new resource is expected to enter the project at a certain time which usually includes holiday time (e.g., summer), expect a default of 3 weeks' holidays if the resource is not as yet named.
- Juniors should not be given tasks of duration longer than 2 days. (But, note that this may be done internally by their supervisor and not necessarily be reflected in the manager's plan.)
- The workload on a particular resource should be equally distributed throughout a particular period. But the overallocation of a particular resource for a short period according to the plan may still be acceptable provided that the particular resource has lower percentage of utilisation before or after the particular period (i.e., check average for the period).

- Check the *minimum* percentage of a resource's involvement in a task. A normal minimum involvement in a task is about 25-30% (unless that is the ***entire*** involvement of the resource in the project which will naturally lower the percentage of involvement in the task due to learning overhead about the project). This rule does not apply if the resource is involved only in a support activity when involvement as low as 10% is acceptable (e.g., in maintaining a data model, making back-ups).
- Are there any gaps in a person's allocation?
- If a resource is not involved full-time in the project, keep a record of the resource's expected obligations to other competing projects in which he is involved.
- Induction time should be allowed for within the allocation of an incoming resource. This tends to vary according to how much there is for him to learn concerning the project. For example, designers, in general, are allocated more time than coders.
- Ideally, the resources who will start off the project (e.g., functional design) should be the same as those who will close the project.
- Consider that, for long projects, the position of a resource within the project may change (i.e., a person may be given more responsibility later on).
- The person involved in detailed design should be assigned the coding of the corresponding design.
- In a combination of junior and senior programmers consider whether the senior programmer has been allocated in the plan sufficient time for supervision.
- Consider the correspondence between the number of non-human resources (e.g., terminals) required at the time of implementation and the number of human resources who will be needing them.
- Testing of the code should not be carried out by the person who coded it.
- If there are two people assigned to the same task, it may be advisable to split the task (if possible).
- If overtime becomes necessary to be planned, check the budget because of the increased overhead due to overtime (often 50%).
- 15-20% of the project budget is devoted to project management effort. This should be reflected in the plan.
- Consider the involvement of the project manager in activities other than project management and possible influence on his performance.
- The project manager should not assign himself to tasks on the critical path in any other than in his project manager capacity. (This is a contentious statement; one could also argue the contrary.)

The list just given, of course, was not meant to cover the whole range of concerns of a manager when assessing the quality of his plan. Assessing the quality of a plan is a very difficult task which can only be carried out ***prospectively*** on the basis of received wisdom (such as the items listed above) and on the basis of one's own past experience. Plans can be easily and accurately assessed only ***retrospectively,*** that is, after the project has finished, but, by then, the usefulness of the assessment is for future projects and not for the current project.

4.2.3. Launching the project

To a great extent, planning activities are solitary activities for the project manager. He may have needed to interact with other managers in his own or the client organisation to clarify any questionable points in the requirements for the project, he may have discussed the project with colleagues to gain from their experience and improve his planning, he may have asked the opinion of known or potential members of his future team about aspects of his planning where their technical expertise may be needed. But none of these interactions, however advisable, may be essential for plan development. The project manager could still shut himself in his office and come out with an acceptable plan in his hands. Fmm that point on, however, no activity of the project manager is really a solitary activity. Anything he does will involve transactions with the project work system and/or the project environment.

A completed plan needs to be accepted by the client and the manager's own organisation before it is put into operation. The initial planning phase finishes with this acceptance which, sometimes, may only be achieved through complex negotiations with the client and/or his own organisation aimed at fixing an agreed-upon plan for the project work. Once the plan for the project has been approved by the client and the project manager's own organisation, the project is ready to be launched.

The launching phase is effectively very short in terms of time. It simply entails activating the project by communicating to the project team the project plan, allocating work, and setting up and communicating to them the monitoring and reporting procedures that will operate for the project. The latter will depend on the internal and external milestones that have been identified in the plan and on the organisational structure that the manager has created for his project, respectively. The phase itself closes with the communication of this information to the project members and the running the project phase immediately commences.

Just as the planning phase necessitated a view into the future in defining the project work system that will deliver the product (imagining about the composition of the project work system, its productivity, et~.),this phase also necessitates such a view, but, this time, simulating through the real, mostly known, project work system he has constructed, manned with certain individuals, guided by given constraints, in order to determine how to man-manage it

and facilitate its smooth and productive running. Failure of a manager to provide for a facilitating work environment fostering good performance often leads to the failure of the project (Paretta & Clark 1979).

4.2.3.1. Providing for a facilitating work environment

In focusing on developing a plan which will achieve the objectives which have been set externally for the project, within constraints which are also set externally, the personnel engaged on the project tend to be viewed as "human resources", the main planning problem being to ensure that a particular human resource is available at the time the task allocated to him is scheduled in the plan, and that he has the appropriate skills, qualifications and experience to carry out the task.

In resourcing the tasks in the plan, human and machine resources may be treated equivalently in some respects. In the case where a human resource falls short on a required skill for a task which the manager would like to allocate to him, explicit provision can be made in the project plan for a period of training with the aim of gaining the required improvement in the resource's requisite skills. Similarly, a machine, nominated as a resource but lacking a particular characteristic, may be able to be "upgraded or a period of "system development work" be scheduled on it with the equivalent aim.

However, the gaining of desired skills in human resources may also follow the ***apprenticeship model,*** whereby they eventually train themselves as a result of experience on particular tasks. Hence, greater flexibility can be gained when scheduling personnel on tasks through viewing their skills as subject to improvement in particular ways resulting from assigning them to particular sequences of tasks. As we saw in some of the items (listed in section 4.2.2.3) which guide managers in assessing the quality of their project plan, this possibility is always worth considering, as the apprenticeship model holds regardless of whether or not any of the relevant tasks were explicitly planned for the purpose of training or for upgrading personnel.

However, major problems of organisations lie at the interface between taking a resource perspective on human resources versus taking a personnel (human) perspective on them (Leavitt, Dill & Eyring 1973). Within the former perspective, people are seen as "hands", in the latter, as motivated agents. Organisations and managers who adopt ***only one*** of these perspectives to the expense of the other can only fail to achieve their objectives (e.g., high personnel turnover in the first case, undirected or unstructured work in the second one).

The notion of a ***motivated agent*** addresses the fact that a person is guided by his own goals, has his own aspiration level, his own needs for recognition, job satisfaction, personal growth, and so on. Recognising the existence of these needs (which are most often different for different people) is the starting point towards a good work design for any particular person (Hackman & Oldham 1980).

The counterpart of the notion of a motivated agent is that his motivation may be increased or decreased at will by others or by certain external conditions. A great deal of literature purports to define how motivation can be enhanced (e.g., Couger & Zawacki 1978, Baron 1986). The most often quoted example is the work of Herzberg and his colleagues (Herzberg, Mauser & Snyderman 1959). They describe as "motivators": promotion opportunities, achievement, increased responsibility, nature of tasks performed, recognition from management, opportunities for personal growth. These are suggested to affect job satisfaction while job dissatisfaction is suggested to stem from "hygienes" or "maintenance factors" such as company policies, pay, job security, quality of supervision, relations with others, physical working conditions. Any of these variables can be manipulated ***externally***, i.e., by the manager and/or his organisation.

However, the notion of a person being a motivated agent, possessing his own goals and objectives, creates problems for managers planning such manipulations: not every person will be susceptible to the same incentives used to enhance his motivation. The decision by a manager or an organisation to manipulate ***certain*** variables (assumed to enhance motivation) and not ***others*** is usually based on assumptions about what would motivate ***any*** person (e.g., higher pay). This may or may not work for any particular person. Herbert (1976) argues that

> "a superior does not directly motivate his subordinate. He merely ***establishes*** the environment and the opportunities for the subordinate to motivate himself" (p. 249).

This notion of motivation enhancement assigns agency for the degree of motivation to the motivated agent rather than to the manager or the organisation. The latter are seen as contributing to its enhancement through just setting up the conditions under which it can flourish.

This is closer to the approach often utilised by managers of software development projects. Software development people tend to be seen as special types of people (e.g., Weinberg 1971), intrinsically motivated in their work (e.g., Couger & Zawacki 1978, Licker 1985). What is important, then, becomes how to ***maintain*** their motivation and, more specifically, how to capitalise on it within the project in question. Most of the effort project managers spend to that effect is invested in ***creating*** this facilitating environment and in making sure that the project work that will be assigned to their team members, when possible, includes characteristics of work they consider important (e.g., variety, task coherence, discretion; Cheney 1984).

Moreover, the manager can and does affect the quality of life of his staff through his own activities. Software development projects are often subject to strenuous conditions of work necessitating overtime, working at odd times. often working in a crisis mode. This style of life spills over into the private worlds of project team members as well as of the manager, frequently with disastrous consequences (Levinson 1981, Lasden 1984, Cherlin 1984). This often can be avoided through better planning and management of the project and of its relations with the project environment. Flexibility and the fostering of

the ability to respond quickly to all kinds of pressures is a conventionally desired characteristic of current project management practices (British Institute of Management 1985).

However, it is not only considerations of individual needs and desires that guide the design of a facilitating working environment for a project Project work is typically carried out within a team†. Considerations which relate to working conditions and enhancement of motivation in the project work take now another, more complicated, dimension. That is, human resources, drafted into a project work system, become team members with their own goals and motivations, and communication needs, which are not always congruent with those of other members of the team (Weinberg 1971, Brooks 1982). The ways in which these goals and motivations are appreciated in designing the project team can have crucial effects on the capability of that team to operate as a team in meeting the objectives of the project (Bartol 1977, Unger & Walker 1977, Parkin 1983, Licker 1985).

However, a team is more, and different, than the sum of all the individuals who comprise it (Blake & Mouton 1964); it has its own momentum which can be enhanced, motivated, depleted, and so on. Building a team takes time and effort, often starting from a period of less collective productivity than the sum of the productivity of its individual members (Freedman, Gause & Weinberg 1984). Members need to learn to work together by learning about each other; team building is a process, not an event (Dyer 1987). The outcome of this process is, hopefully, a team that functions well. Naturally, when this is achieved, there is a great reluctance to disband the team (although this is nearly always inevitable in time-limited projects).

In software development projects, the nature of the project itself necessitates the creation, expansion, contraction and disbanding of teams at different times during the software development life cycle depending on the skills and kind of organisation that is needed by the particular phase of the project (Keen 1987). To some extent, all positions in the project are temporary. The implication of this would be that the manager will need to invest a considerable amount of his time building teams. In reality, what happens is that project managers invest a great deal of their time at the beginning of the project, building a team for the project, the project team, with an identity defined in opposition to teams of other projects. The original project team expands as more individuals enter the project, are initiated to the team, and learn to function within

† The *structure* of teams in software development projects is often discussed in the literature in terms of two diametrically opposing notions. One is that of an *egoless* team (Weinberg 1971), a democratic structure where group leadership becomes the responsibility of the team member whose skills are currently needed. The other is that of a *chief programmer* team (Baker 1972), a centralised autocratic structure where leadership resides with the chief programmer and decisions are made at the top level. An alternative structure, drawing from both notions, is that of a *controlled decentralised* team (Mantei *1981)* where leadership resides with the project leader who governs a group of senior programmers who each governs and leads a group of junior programmers.

the team. Later on, the team will contract as the project progresses and will be disbanded at the end of the project. The manager has to invest his time in sustaining the spirit and motivation of the project team as it changes its composition and structure through time.

Ideally, designing team work should involve describing unambiguously what is expected of each team member according to his role in the team, demarcating each member's responsibility, authority and discretion. Ambiguity in the definition of team members' roles frequently results in job tension, lack of job satisfaction, a sense of futility and reduced self-confidence (Kahn et *al* 1964). Conflicts between the responsibilities of the different roles that a member has to play within the team can also cause considerable stress. Conflict situations arise, for example, when some of the expectations associated with the various roles allocated to the same person within the project work system are incompatible, or when the status of any particular role a person is asked to play is lower than that of the kind of role the person expects to be asked to play (e.g., a senior analyst may not be happy to be assigned work as a programmer).

Work overload can also create considerable stress (Blacker & Shimmin 1984). For example, a project manager's planning, focusing on the completion of a task on schedule with a fixed set of resources, may require a team member to do too much in too little time or to act on inadequate information, without the requisite material, or without being adequately prepared. Under these conditions, the team member is unlikely to carry out responsibilities delegated to him satisfactorily. Conversely, underutilisation of a team member's capabilities and skills within the project work system can result in frustration and job dissatisfaction for the person concerned (Handy 1981).

In theory, during task allocation, in all but the most authoritarian of organisations, it is possible for a team member to reject the commissions given to him as unachievable (in much the same way as a project manager can reject the commission to manage a particular project). In practice, this does not happen very often, although if the team member does not believe the objectives he is told to meet are achievable, be will not be committed to them and he will almost certainly fail to achieve them (Baron 1986).

4.2.4. Running the project

Running the project is the most difficult phase of project management. From the point of view of all interested parties, it is on the successful execution of this phase that the project's chances to succeed or fail depend. To some extent, one can argue that running the project constitutes project management ***par excellence*** as this phase involves the execution of most, if not all, managerial functions (e.g., planning, controlling, directing).

A good initial project plan may give the project excellent starting conditions for success but cannot guarantee this success. Conversely, a poor initial plan coupled with a skillful project manager may provide the project with all it

needs to meet its objectives satisfactorily (e.g., through negotiating with the client for a better plan, through directing his resources appropriately). When projects are perceived by the contractor organisation to be in great trouble, often *fixers* are called in to act as their project managers, replacing the current manager. These persons are experienced project managers typically employed in the fixer role in different projects which are faced with similar problems as they have the very special kind of skills required for emergency situations (Shaw 1984).

However, it is the norm rather than the exception that something will go wrong during the execution of the project. Typical problems that can befall projects are: overspending (all resources have been exhausted), underbudgeting (often due to incorrect estimating), product failure (due to poor quality assurance), lowering of staff motivation leading to low productivity (and, indirectly, overspending), rapid staff turnover (and, hence, extra effort required for selection and training of new staff, and, therefore, overspending), etc. The source of these problems is not always directly traceable to some shortcoming of the manager. It could be that the constraints under which he had to manage his project were unrealistic and he had no power to change or affect them. He could only attempt to *minimise* their effects on the pmject through his own actions. Conversely, for things which are under his control, timely managerial action could probably have saved the project.

The phases that precede the running phase set up the conditions under which the project will be executed. However, these conditions rarely remain stable as the project progresses. Changes in, or the resolution of misunderstandings about, the project and the software system it is developing may lead to new requirements and demands within the project, or to an unexpected reception for project deliverables, etc. The early discovery of misunderstandings and anticipation of changes in requirements and the management demands associated with them eases the task of the project manager, greatly reducing the risks of the project and the need for emergency re-organisation of project activities (Bunyard & Coward 1982). This requires that the manager exhibits continuing awareness of issues which can affect the project determined only by looking

- ***outwards*** from the project towards the different stakeholders who constitute the project environment (client, general management of the contractor organisation, other subcontractors, regulatory agencies) as they often have different conceptions of the project and different expectations of what is to be achieved through it, and,
- ***inwards*** to the project towards the project team members in order to maintain the team spirit, make sure they work within a shared vision of the project (a key element to its success, Powell & Posner 1984), and so on.

Most of the problems that arise during this managerial phase tend to depend on the skill of the manager at playing the role defined by his position in the interface between the project work system and the project environment This role necessitates him acting as a *buffer* between them, interpreting their demands of each other, communicating the results of these interpretations to the

relevant parties and acting on them to guarantee that these demands will be met. All these activities bring to the fore the importance of the transactional aspect of this phase and, hence, the need to deal with the uncertainty about how people will behave during these transactions that the manager faces (Wynne 1983). Managers tend to spend as much as 80% of their time relating to people, mostly in informal, verbal interactions (Mintzberg 1973) and this is particularly true during this phase. Project management methodologies can not offer much help in this aspect of management (Waldrop 1984).

The following two sections discuss what defines the transactions of the manager with the project work system and the project environment and the importance of their successful conclusion for the project. We then discuss (in section 4.2.4.3) how the results of these transactions are used by the manager to manage any resulting need for change within his project.

4.2.4.1. Transactions with the project work system

Transactions with the project work system are essentially of two kinds: one determined by the need of the manager to know about the progress of the project and one determined by the need of the manager to ***foster*** progress in the project work. The project work system acts as the provider of information to the manager and as recipient of the results of his managerial activities. The project manager acts as the interpreter of the information provided by the project work system and of demands made on it by the project environment and as a facilitator of project progress given these interpretations.

The project work is carried out within the terms of reference of the (current) project plan. The macroplan guides its general direction while the detailed plan describes who is to do what and when during the period of its application. Monitoring progress of the project is set against the detailed plan and its results are fed into the macroplan by the manager in order to enable forecast of system development cost and resources that will be required, show the target dates of remaining checkpoints and deliverables, and such like (Parkin 1983). To ensure that all goes well, the manager needs to monitor the status of the project regularly. In order to keep track of progress he requires information on

- estimates of work needed to be done on all active tasks (provided by team members, and, if possible, by an independent source),
- warnings concerning new tasks which have surfaced during the execution of the project (e.g., external or internal change requests),
- resources consumed (e.g., time, financial, physical resources, contingencies) against their predicted consumption at this stage.

However, a balance must be found between the effort involved in collecting this information and its accuracy. Usually, this results in periodic status data collection, the time intervals being dependent on the size of the tasks involved and the project manager's confidence in the project's progress. Normally, the

setting up of the monitoring procedures that has taken place during the launching of the project will have defined these time intervals.

However, measuring progress in terms of effort-spent and effort-to-be-spent does not address issues which relate to the quality of the work being carried out. If a task is discovered to need re-working after it has been assumed to have been completed, its progress measured in terms of effort at an earlier stage is, with hindsight, seen to constitute a misleading measure. However, there is a marked lack of precise and unambiguous scaled criteria for measuring quality (Jones 1978). Nevertheless, actual product quality needs to be monitored as objectively as possible. The responsibility for promoting and monitoring quality in products emanating from the tasks of the project is often assigned to an independent quality assurance representative and quality assurance reports are then presented to the manager as part of the process of monitoring progress on the project. The information in these reports must address technical issues but must also help the manager interpret these issues in management terms in such a way that he can take corrective action (Lybrook 1982).

Gathering all these various types of information is a means to an end for the project manager, who must, on the one hand, report to his superiors on the project's progress, and, on the other, act upon the problems he has uncovered during the process of monitoring. Monitoring progress not only informs the manager about the progress of the project but also provides him information about the *future* progress of the project. For example, he can use it to ascertain the actual productivity of his resources, their actual range of skills, the difficulty involved in some tasks, and so on. This can facilitate any necessary re-planning of the work. Indirectly, it also provides him with insight as to the quality of his current plan in terms of any concerns which have not been addressed within it. For example, the plan may not have taken into account learning curves and trial-and-error methods which can result in unproductive time (Gordon & Lamb 1984), or not scheduled resource requirements such as keypunch, test time, typing, printing, etc., which often cause problems while the project is running (Keider 1979).

Walkthroughs, reviews and inspections may be seen both as a means of controlling the project work and as a means of fostering its progress. In the former case, they provide a quality assurance function for the software (Weinberg & Freedman 1984, Fagan 1986). In the latter case, together with project audits, they entail project evaluation from an external person's point of view thus encouraging project identity, identification of team members with project results, and alertness about where the project is heading (Cleland 1985).

It is unwise to see fostering progress in the project as a *side-effect* of monitoring in the sense that it becomes imperative only after progress is found to be slow or lacking. Instead, the manager should be on the lookout for ways of fostering progress in the planning phase, when he assesses his plan, and in the launching phase, when considering his team and the needs of his team members. On both those occasions, the project manager needs to simulate the project plan to define the conditions under which progress can be fostered. In the running phase, the ability to foster progress is med out in reality: it consists

of creating the real rather than hypothetical conditions under which his team can flourish.

In fostering progress, the manager's style of management plays a very important role. This can range from an *authoritarian* style, through a *free-rein* style, to a *democratic* style of management (Herbert 1976). The authoritarian style ensures high productivity over a short period of time, entails high consistency and uniformity of purpose but it also endangers progress. This is because it entails a conventional attitude, not open to change, and an inability to relinquish authority in cases where the required competence lies with subordinates rather than the manager. For example, problems frequently arise when managers whose previous careers focused on the development of technical skills but who now hold a purely managerial role in the project, attempt to control the technical part of the work as well. As such projects progress, the technical work usually suffers as the manager/technician becomes more distant from the technical content of the work as his time is all consumed in meeting his assigned management responsibilities (Freedman, Gause & Weinberg 1984). In contrast, the democratic style is flexible and enabling, integrating individual and formal motives. However, it imposes a great time commitment on the manager and its success depends on the subordinates' contributions (Herbert 1976).

The free-rein management style, on the other hand, is based on delegation. It provides for autonomy and freedom that may be needed by a task thus generating motivational intensity and enabling personal growth. However, it may create coordination difficulties for the manager and is dependent on each subordinate's ability to take responsibility for the task assigned to him (i.e., it assumes a certain level of self-management or sub-team management ability on the part of the delegatee). It involves assigning responsibility and authority to the delegated person while, at the same time, making him accountable for the results of his activities to the project manager (Benge 1976, Raudsepp 1981).

The ability to delegate is necessary in project management else the manager may find himself overwhelmed. Jenks and Kelly (1985) recommend that the manager should delegate everything but ritual, policy making, specifically personal matters (e.g., evaluation, discipline, resolution of disputes, delegation itself), the handling of crises and confidential matters. Delegation presupposes that a pre-defined set of objectives, plans and standards have been set up. These provide a framework wherein the delegatees are allowed latitude to make decisions of greater or lesser importance given the degree of discretion they are allowed.

While all the above indicates that the style of the manager is very important in fostering or hindering project progress, it would not be right to assume that this means that there is a single, correct, managerial style (Gregson & Livesey 1983). Each particular style seems to have its positive and negative aspects. The most effective style to be adopted in transacting with the project work system needs to be decided on each project afresh, rather than the manager adhering to some fixed, preferred, style irrespective of the particular needs and situation of the project. Ideally, a manager will need to be able to adapt his style to the situation, to the particular project and to the project

environment.

In practice, however, this is a lot to ask of any particular manager, requiring chameleon-like abilities and consummate acting skills. This is why, in chapter 3, we suggested that one of the most important risk management issues to be determined on the basis of a project risk profile was that of identifying the required management style and skills for the project before the project manager is assigned. Then, given that the contractor organisation has a pool of available managers, one can be selected whose preferred management style is consonant with that required for the project to which he has been assigned.

However the appropriate management style is achieved, the result should be to ensure that

> "in all interactions and all relationships.. each member will, in the light of his background, values, and expectations view the experience as supportive and one which builds and maintains his sense of personal worth and importance.." (Likert 1961, p. 103).

Moreover, it should facilitate constructive interaction between project team members, stimulate enthusiasm for meeting project objectives and facilitate the achievement of these objectives through planning, coordinating, providing resources, and so on (Bowers & Seashore 1966).

4.2.4.2. Transactions with the project environment

Optimising the nature and style of the transactions between the manager and the project work system can not in itself guarantee the success of the project. The manager, in running the project, needs also to be successful in his transactions with the project environment. This is necessary in order to make sure that the requirements the project is working towards are the ones agreed upon by all parties involved and that the deliverables of the project will meet these requirements from the point of view of all external stakeholders.

Major difficulties in carrying out these transactions stem from the fact that the project environment is seldom stable. In this environment, stakeholders' views may shift as, for example, when the client changes his requirements or finds that the deliverables, while meeting the terms of the contract, do not solve the problem their application was designed to handle. Moreover, this problem itself may have changed in nature since the project's conception. On the other hand, the project manager may need to respond to changes originating in the environment of his own organisation as, for example, when there is a change in the availability of resources or in the priority his project had been assigned. In general, changes in the project environment produce challenges to the project manager involving new potential risks which can only be obviated or kept under control through the manager's own activities.

Any of the requirements imposed by the environment on the software system under development, the project and the project manager are liable to change at any time during the lifetime of the project. Changes in the

requirements for the software system are a frequent occurrence as we discussed in section 2.1. Changes in the requirements for the project and the project manager tend to be related to the way stakeholders in the project environment perceive the progress or lack thereof in the project work. The typical source of this information on progress is the project manager through his reporting activities. The recipients of the manager's reports in the project environment have then to interpret this information in the light of their experience and the degree to which they can foresee the project being able to deliver what they require.

There are two basic kinds of reporting:

(1) ***reporting on exception*** which is desirable since senior management would like to assume all goes according to plan unless explicitly told otherwise (this minimises the information flow between project and environment), and,

(2) ***reporting on progress*** which is necessary if senior management feels the need to monitor the project closely (as when the project is viewed as risky in some domains, or as having a relatively inexperienced project manager) and, thus, requires explicit confirmation that the project is, indeed, running according to plan. In this case, the project manager is forced to monitor the project progress more regularly and more comprehensively.

In reporting, project managers usually desire to avoid being the "bearers of bad news". As their reports are made on the basis of the (currently valid) project plan, deviations from that plan m, for the most part, "bad news". A strategy often adopted by project managers in dealing with "bad news" is to act fast to remedy the deviation but to take time in making a report, in the hope that, by the time the report is made, all will be back to normal or worked out satisfactorily.

The transactions described so far between the project manager and the project environment deal with the need of the project work system to gain information through its manager about the requirements it is expected to meet, and the need of the project environment to find out about the progress of the project (i.e., whether its requirements have been or will be met). However, the majority of transactions the manager has with the project environment stem from the need of the manager to negotiate ***changesin*** the particular constraints he and his project have to meet (e.g., insufficient resources, time, unimplementable requirements). We discuss how such changes may be negotiated and managed in the following section.

4.2.4.3. Managing change

Most software development projects involve the management of change at some point during their development. The need for the change comes from the project environment (e.g., changed requirements) or the project work system itself (e.g., illness) or the state of the progress in project work (e.g., delays). Some of the questions to be answered before the decision on how to manage the change

is made relate to the ***cause*** of the need for change. For example, is the problem proposed to be solved by the change local to some tasks (i.e., they are late) or does it affect the whole project (i.e., the whole pmject is late)? does the need for change originate in motivational problems with particular personnel or incorrect estimates having being made about the amount of work involved in the tasks to which they have been assigned? (Turner 1984).

Most often, the need for change is a natural consequence of the lack of knowledge about the project at the time when it was initially planned (expressed, for example, as optimistic estimates) and the increase in the amount of knowledge about it while carrying it out. In the latter case, the manager has a better idea of the actual parameters applicable to the various resources and tasks (e.g., a team member is less effective than anticipated, some tasks have turned out to be easier or more difficult than expected). A new, revised, plan which could reflect this increased knowledge would, by definition, be more realistic for the project at hand. However, each new, revised, plan should always be challenged with the question: "what is being done differently that will cause this plan to succeed while the previous one failed or would have failed?" (Miller 1978).

However, changing to a revised plan can itself produce disturbances in the reality of the project work system and the project environment. A revised plan is likely to have implications for the work of the team, and, thus, it may be necessary to negotiate with team members concerning the changes in the tasks to be allocated to them. This may be welcomed where difficulties involved in switching to new work are perceived as less than the difficulties associated with trying to keep alive the fiction of a now impossible task schedule. On the other hand, bringing a plan closer to the current reality of a project may also necessitate changes in the timing and nature of future resource requirements, milestones and practically achievable deliverables. Such changes are likely to have repercussions within the project environment (in the world of the client and the contractor organisation) regarding the immediate responsibilities of the project manager.

Hence, the project manager is not free simply to invoke the new plan on his own and report that the project is running to plan. Estimates of the external impacts of the proposed project changes have first to be made and included in the project manager's reports. These changes have then to be negotiated with the stakeholders they affect, often at the level of management to which the project manager reports (or at even higher levels, in severe cases). When the repercussions of these changes are great, often issues initially considered and resolved when establishing the project (as discussed in chapter 3) may have to be re-considered in the decision of whether to accept the implication of the new project plan and, thus, enable the pmject manager to continue to exercise his responsibilities.

Most often, changes in the project plan are local changes. That is, they are made on parts of the plan for the purpose of solving or alleviating a particular problem discovered during the running of the project. For example, on discovering that one particular resource has more requisite knowledge for the work on

a particular task than has the resource currently allocated to it, the manager may swap the resources between tasks. This often constitutes a change in the project manager's detailed planning but one which, usually, has no repercussions for his macroplan. Alternatively, it may be deemed necessary to include in the plan a new task for which resources can be acquired easily from other tasks.

Local changes rarely need to be communicated to the client although they are often presented in the manager's reports to his superiors. In any case, the repercussions of even local changes need to be considered for the project as a whole as well as for the problems they purport to alleviate at the local level. For example, the use of a resource in a task which he has more knowledge about may increase performance on that task but may create problems for the task from which the resource was withdrawn due to re-allocation

4.25. Closing the project

Typically, closing a software development project implies handing over the product to the client, having it accepted, arranging for its future maintenance, and final reporting on the project and its results to one's own organisation. It is a good idea to make use of the experience gained through the project while it is still fresh in the minds of the people involved in some way. Project post-mortems are important in providing the team with a better understanding of the software development process and how they can improve their practice, as well as providing the manager with a better understanding of the project management process and how to improve his own practice. Thus, debriefing the project team, archiving information on the project (e.g., plan), and anticipating future use of the project are also desirable project closing management activities.

However, not all organisations insist that such activities are required from their project managers and, hence, although managers appreciate how important they are, they rarely carry them out (Keider 1979). The main reason for this is lack of a source of funding for such activities (unless the organisation itself has a policy to fund such undertakings). Projects are charged to the client up till and including delivery and acceptance of the product (unless the contract included maintenance as well); once this has been done, the project manager's time needed to carry out these extra activities cannot be charged to the project but to the contractor organisation's internal budget. Moreover, as software development projects tend to run late in practice, they tend to use up the manager's slack time between projects. Thus, the particular project manager may have already started working on his next project by the time the closing project finishes, having no time to spare for such activities.

Keeping information on past projects and capitalising on the experience gained through any particular project, however, should be a requirement for any organisation which takes a long-term perspective on its own development. It should be seen as an investment rather than wasted potential project management time. Such information can provide the organisation with a reliable and

sound basis for future bidding for successful projects and a basis for transferring valuable information from one project to another (Hales 1979, Keider 1979).

4.3. Anticipating possible threats to the success of the project

From the time the project manager takes over the project to the time that the project closes, he needs to exhibit continuous vigilance to cope with threats that his project may face in fulfilling its objectives, and to determine strategies he needs to adopt were these threats to realise. This vigilance is expressed in two ways: through the manager (a) watching the project and acting when something happens, and (b) simulating what might happen and taking preventive measures to avoid it from happening or making contingency plans to cope with it. Both are important means of vigilance and they should both be exercised. If only the first is exercised, this can pave the way to a constant crisis management mode of operation and, indeed, events that lead to crisis are the norm rather than the exception in many projects. If only the second kind of vigilance is observed, the manager may lock himself up in loops: simulating all that can ***possibly*** go wrong in the project forgeting to observe what ***actually*** does go wrong as it is carried out

In this section we describe what is involved in anticipating possible threats to the project's success through carrying out ***what-if simulations,*** that is, performing the second type of vigilance. How the manager watches the project has been described in the previous section and how he can act to cope with adversity he has come across while watching the project will be addressed below in connection with the simulated rather than real occurrences of these problems.

What-ifs are an essential part of the manager's planning whether this planning relates to planning his relationships and transactions with external stakeholders to the project (e.g., answering questions such as "how should I behave towards this particular client?", "what kind of reporting would satisfy my senior management?", "what can I do if other project managers in my organisation create difficulties for me and my *staff?")* or to planning relationships within the project work system itself (e.g., answering questions like "what would happen if I do not allow for enough reading time for my team?", "what could I do if my project team does not work well together?"). Carrying out what-if simulations effectively leads to a better planning of these relationships.

The purpose of carrying out such simulations is to ascertain the consequences of an anticipated event occurring on the project (as it stands at the time of the simulation or at some future state) for the purpose of deciding on a course of action to be taken if and when the anticipated event occurs or how it can be avoided by taking precautions now. This implies that the project manager has to

(1) define the anticipated event,

(2) determine the aspects of the project on which the event is going to impinge directly or indirectly,

(3) speculate on the nature, magnitude, and desirability of these effects, and,

(4) decide on possible course(s) of action.

There is a great number of such possible events on whose occurrence the project manager may decide to speculate and carry out what-if simulations. However, not all imaginable events are likely to happen. This implies that the assessed likelihood of the event happening will determine whether the simulation will even commence. Since such simulations are quite cumbersome processes, it is often the case that imaginable events which have been assigned low probability of occurrence are omitted, even though there might be serious consequences should they materialise.

The counterpart of what-if simulations is the *assumptions* the manager makes about his project. For example, assuming that a particular resource is going to be available by a particular time biases the project plan on all pieces of work which depend on the availability of the particular resource. All project plans are based on assumptions of this kind as, otherwise, the plan will never be completed since everything in the plan can, potentially, be uncertain. The problem, however, is that such assumptions are often implicitly made and not documented in the plan. This makes tracing back elements in the plan which have been based on such implicit assumptions very difficult, in the case that it is suspected that a problem with a plan is due to unspecified assumptions having been unfounded.

What-if simulations are the cornerstone of contingency planning. In such cases, the plan includes indicated courses of action if certain events do occur for real. Contingency planning helps the manager avoid panic-led situations and crisis management Here again, the number of anticipated events for which the manager could, in theory, cater within his contingency planning is great. Hence, he has to make some choices about which possible events he will include in his contingency planning and which he will leave out. As we mentioned above, one criterion commonly used for excluding events from a simulation is its judged low likelihood of occurrence. However, the likelihood of an event occurring is only part of what determines its significance for the simulation. The other part is determined by the magnitude of its consequences, were it to occur. For example, keeping back-up copies of software under development is an instance of contingency planning for a possible event (i.e., accidental erasing of files) which otherwise might have catastrophic consequences. Another example is the insistence on using a standard method for documenting software development work to cater for the possible event that any particular resource working on a particular module leaves the project and the replacement may find it impossible to understand his code.

Furthermore, anticipated events (especially negative events) do not only happen one at a time. For example, consider the case when a resource who was the contact point with the client is withdrawn at a time when the client starts being difficult. What-if simulations which had assumed the ***independent***

occurrence of these two events, could have given rise to courses of action based on assumptions about the other part such as "the client is being difficult but the resource who is the contact with the client can manage this difficulty" or "our relationship with the client is stable enough not to suffer from the withdrawal of this resource". In that case, we are dealing with assessing conditional probabilities. Here again, we have the problem of a great number of combinations of anticipated events that may occur simultaneously.

What-if simulations on the occurrence of a certain event may not even start if

(i) the event is assessed as unlikely, or

(ii) its consequences have been underestimated (assuming that it will be easily handled if and when it does occur).

Carrying out a what-if simulation which yields reassuring results, however, does not necessarily ensure that the threat to the project's success has been neutralised. Even if the manager carries out a what-if simulation on a particular event, he may

(a) not have determined all the project elements which its occurrence will affect; or,

(b) not have taken into account the ***snowball*** *effects*, such as the project elements which are indirectly affected by the event by the propagation of this event's effects through elements which are directly affected by it; or,

(c) not have defined the event properly (for example, define the event as "slippage by 1 week" rather than "slippage by 1 week at the testing phase of the software development life cycle"; the seriousness of an anticipated event happening at some stage of development will depend on which stage it is: slippage of 1 week during early stages is less serious than slippage of 1 week nearer the end of the project); or,

(d) not have considered other anticipated events which are likely to occur in conjunction with the particular event.

In all these cases, the simulation may have been performed but its results will not provide the correct picture of the threat to the project on which the manager can decide how to act to neutralise the threat.

In the following, we discuss the different types of events which may be used as the object of what-if simulations in project management. We address these events under five categories starting from those which often may fall outside the project manager's responsibility, to those which are an essential part of his work. The discussion of events which may be outside the scope of the project manager's concerns aims at outlining the effects of such events on his work when they do happen even if he has no control over them.

Events in the first category are those over which the project manager has no control and has no power to stop them from happening or even make contingency plans for them, however likely they may be and however high their consequences might be. These events are rarely the object of what-if simulations by the project manager because their consequences are usually outside his

own responsibility. An example of such an event is when the client or the contractor organisation, due to some change in internal policy, cancels the project In this case, it is not the manager's job to find alternative employment for his team members unless he is located at such a position within the organisation which defines this as his responsibility (as, for example, having the responsibility to bid for another contract). Another example would be the simulation of government changes and results of elections to anticipate the effect of policy change on a government-funded project

In the second category, we encounter events which affect the way the project is planned to run. These events relate to fundamental assumptions on which the plan was based (e.g., a change in the development life cycle model or in the hardware or the development language used). Here, we are dealing with high consequence, low-probability events which may be simulated only when their probability of occurrence is considered likely under the particular circumstances (e.g., having experienced the client as somebody who often changes his mind *or* the likelihood of a hardware needed not being available is high). In such circumstances, simulation of these events occurring leads to either not undertaking the risk of facing such problems by opting for safer options (e.g., choosing an existing or more certain hardware and, if the client insists on the particular uncertain hardware, formally warning him of the consequences) or by binding the probability of the occurrence of such events by strict change control procedures (in the case of the client changing his mind over language to be used *or* the functional requirements of the product). However, even at this level, we are discussing problems of managers who have the responsibility and the authority to negotiate the project at the time of its inception. This is not often the case for the majority of project managers who are handed a project, together with a fixed brief already negotiated and are asked to manage it as best as they can.

Simulation of possible events classified under the third category facilitates better estimates being provided to the project and helps in defining a realistic project plan. The object of the what-if simulation in this instance is a particular variable aspect of the project as a whole but on which the project is dependent for its successful execution. Examples here are what-ifs related to time ("can we do it in x man-months?"), budget ("can we do it with fewer people?"), productivity/performance ("what if the rate is less than average?"), reliability, function and responsiveness of the system, machine availability, and so on. These what-ifs are carried out usually before the project starts and lead to information for the bidding for the contract They can also be carried out later on, during the very early stages in the project's life, and they normally lead to a negotiation process between the project manager (or his superiors) and the client in changing some aspects of the contract

All the what-ifs we have described in the above three categories facilitate setting up the conditions under which the project will run. Thus, by the time they have been camed out, they define and fix the overall constraints within which the manager will have to operate to carry out his responsibiities. All the what-ifs that are carried out after that are carried out within this fixed project

structure and facilitate project planning or re-planning. There are located within categories 4 and 5, as described below.

In the fourth category, there are possible events which are considered during the planning stage and are those which may happen (or tend to happen) during the project's life. What-ifs carried out on these events help set up a project plan which can cater for such occurrences and test its tolerance to adversed conditions. Examples here would be: anticipated difficulties with the client, project resource withdrawal, machine unavailability, etc. once the project has started. To cater for the occurrence of such events, contingency plans are made which specify what to do if and when these events occur. However, there are a great number of what-ifs that can be carried out to cater for all such possible events and, since these events may not ever happen, project managers do not habitually carry them out Instead, during the planning stage, what-ifs tend to be related to specific aspects of the project plan and simulation of the ***plan*** with different configurations. For instance, "what will happen if I decompose this particular function into two rather than three subtasks?", "what will happen if I combine the work of an experienced resource with an inexperienced one?", "what will be the consequence of scheduling this particular task earlier?", and so on. The effects of these what-ifs are simulated throughout the plan and the plan which appears more reasonable at the time is chosen.

The most frequent occurrence of what-if simulations takes place once the project starts running. They constitute our fifth and final category of what-ifs. These are normally triggered by some occurrence (an initiating, triggering, event that has ***already*** happened) and they reflect responses of the manager to the particular occurrence. There are two types of occurrences that need to be considered within this category: (a) an occurrence which affects the whole project and which normally involves the falsification of some initial fundamental assumption on which the project plan was based, and, (b) an occurrence which affects only a small part of the project the effects of which the manager can easily contain within his project without necessarily involving higher management to solve. All these what-ifs are carried out ***given*** the occurrence and simulating effects of alternative solutions to the problems arising from the occurrence. We consider the nature of each of these types of occurrence in more detail below.

(a) Classified under this type of occurrences are events which are instigated from the project environment rather than from within the project work system. The project manager has no agency in affecting them, he can only minimise their effects. The effects of these events may impinge across the whole project rather than only on parts of the project. A good example here is when the manager has been given human resources who, as a whole, are below the ***standard*** resource he had been basing his planning on and whom he is not allowed to or cannot change. In this instance, if his estimates were based on standard resources or on resources who perform above average, the project manager is going to have great difficulty meeting his deadlines. Another example of such an occurrence would be the discovery that a particular software product on which the project was

relying is not made available by the time it is needed. Examples of what-ifs in that instance would be: "what if we use some other specific software instead?", "what if we develop it ourselves?". Also, another example of such an occurrence would be the discovery that the complexity of the system had been underestimated when the contract was negotiated or when the plan was made. Examples of what-ifs that can be carried out in this instance would be: "what if I employ more resources?", "what if I negotiate a delay in delivery?". The occurrences described in this category are what had been simulated (i.e., had been the object of what-ifs) under category 3, as described above.

Typical ways of coping with such events when they occur are renegotiating earlier estimates (if it is possible) and getting more realistic deadlines, agreeing on a lower functionality of the system (if the client accepts it) or reducing the degree of testing that can ensure the quality of the software with *or* without the client's knowledge, and so on. These courses of action will be the outcome of what-if simulations on the project as a whole (e.g., "what will happen if we reduce the functionality of the system?"). The decision of which particular avenue to take will be based on a trade-off between the consequences of each course of action. For example, lowering the quality standards for the development of a system may not be considered a desirable alternative if the contractor organisation will also be responsible for maintaining it after its delivery to the client.

(b) The second type of occurrences which trigger what-if simulations in category 5 are events which are instigated from within the project work system itself. Examples here would be: slippage, withdrawal of a human resource or unavailability of a machine, discovered lack of expertise in a particular needed area, etc. These are the most frequent causes for carrying out what-ifs simulations by project managers. Each type of such an occurrence triggers what-ifs which may result in resource re-allocation, task or workbreakdown re-definition, re-scheduling, re-estimation, or in actions which have been described under category 5(a) above (e.g., reducing functionality or quality).

What-if simulations are, effectively, means to an end: their value is determined not by the fact that they have been carried out but by their results being reflected in the manager's subsequent planning and actions which consequently will lead the project to its successful conclusion. They provide valuable information to the manager, increasing his confidence that, if and when anticipated threats to the success of his project materialise, he can cope with them. Of course, there is always the danger that one may spend a great deal of time simulating what ***might*** go wrong, leaving little time available to devote to finding out what ***has gone*** wrong, or whether the plan that has been designed on the basis of all possible contingencies is workable or whether it is actually being used as the basis for project work.

Chapter 5

Goals, activities and perspectives as organising concepts

The previous chapter outlined the process of project management from the time a project manager takes on the responsibility for a particular software development project to the time when the project is completed (hopefully successfully). We discussed this process in a chronological order (i.e., in terms of project management phases), describing the activities that need to be carried out within each phase and providing some indications about how the successful completion of each phase may be achieved. Thus, our focus was on what to do, when to do it and how to do it.

Still, throughout our discussion, one may detect another, latent, strand of argument which could not be explicitly addressed within a chronological account of the process. That is, that project management is a ***motivated*** work engagement, where activities which comprise it may be performed at times other than one would have expected them to, depending on the emerging needs of the project For example, making a project plan could be an activity which is performed at any time throughout these phases (apart from the closing phase), negotiating working conditions may be an activity that needs to be performed again and again throughout the process till the conditions become satisfactory, and so on. Thus, the question of ***why*** a particular activity is performed and what the manager tries to achieve through it should be addressed as well as what it entails, when and how it is done.

To address this issue of motivation in the project management process and explain concerns which were merely touched upon in the previous chapter, we shall use the concept of ***project management goals.*** These goals are described in section 5.1 below. They will be seen to provide the rationale for carrying out a particular project management activity ***and*** the testing ground for the success or failure of its execution. This relationship between goals and project management activities which implement them is discussed in section 5.2. However, for a project management activity to be carried out successfully, information is needed from the project work system and the project environment. How this

information may be gained is discussed in section 5.3, where we introduce the concept of the ***perspectives*** that the manager can take on the project work system and the project environment.

The concepts of ***project management goals, project management activities,*** and ***perspectives are*** used in this book as ***organising concepts:*** they organise, in an informal way, the issues raised in chapter 4. As such, they provide the required conceptual link between the discursive discussion of the project management process in chapter 4 and the pre-formal and formal modelling of this process which will be the concern of chapters 6 through 10.

5.1. Project management goals

Project management goals are not coincident with the goals of the project manager as an individual (e.g., achieving recognition, making more money. enjoying job satisfaction). These are the manager's goals within the context of the role he plays within the project. This role we have already defined as being played out at the interface between the project work system and its environment. This leads to the identification of two global goals to be achieved through his management of the project: one addressing the project work system (i.e., ***manage the project well***) and one addressing its environment (i.e., ***transact with the project environment well***).

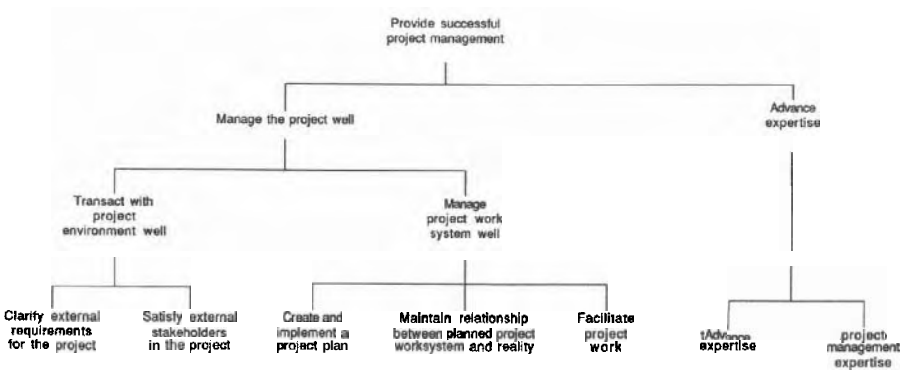
However, the successful completion of any particular project always depends on the technical and managerial experience which is injected into the project through its resources and enriched and enhanced through the project work. A third global goal, to ***advance expertise,*** can guide managers in planning the utilisation of their resources in a way that their technical and/or managerial skills may be increased through working on the project and in documenting the project experience as a valuable reference for use in future projects (e.g., for estimating, diagnosing problems). This goal guides the manager in looking beyond the current project, and contributing to the long term objectives of his own organisation.

The three global goals we have just discussed (that is, to transact with the project environment well, to manage the project work system well and to advance expertise) are too general to enable us to determine how any of them may be actually accomplished through the activities of the project manager. This necessitates refining them into more specific goals the accomplishment of which, collectively, could indicate the accomplishment of the global goal they collectively comprise. Figure 5.1 shows the results of this refinement into seven specific goals which we will discuss in this section.

It must be emphasised that refining global goals in this way and listing them one at a time is not taken to imply that, in reality, the means by which each goal may be achieved are independent of the means by which another goal may be achieved or that the achievement of one goal does not also contribute to the achievement of another. For example, whenever a manager mes to satisfy the goal to create and implement a project plan, he will, at the same

time, further the attainment of the goal to facilitate pmject work just because the existence of a good project plan facilitates project work no end Listing them as if they were independent enables a more detailed discussion of what their achievement would entail (in sections 5.1.1 through 5.1.3 below). It also draws attention to the distinction between goals which trigger one activity (i.e., goals which are at the forefront of the manager's considerations and the motivation for carrying out the particular activity) and those goals which are accomplished through it as side-effects in the way that we discuss in section 5.2 below.

Figure 5.1: Refinement of project management goals



5.1.1. Transacting with the project environment well

The project environment considered under this goal consists of the external stakeholders to the project (such as the client, other managers within the contractor organisation)† from whom the project receives information on the requirements it has to meet and to whom the results of project work are aimed at as their consumers. The project manager has therefore a dual mediating role in the interface of the project with its environment: (a) bringing into the project requirements from its environment, and, (b) returning to the environment the project's products. We express this dual nature of the relationship between the

† Subcontractors and suppliers who also comprise the pmject environment (in the sense that they can affect the project externally) as it was discussed in the previous chapters are considered in this chapter under the goal to manage the project work system: they may be seen as resources to the project (suppliers as providers of needed facilities, subcontractors as providers of needed services) rather than pmviders of requirements to the project and who expect deliverables from the pmject Their requirements from the project and their expectations of deliverables from the project are similar to those of the pmject team (being paid, being provided with necessary facilities, etc.).

project manager and the project environment by refining the goal to ***transact with*** the ***project environment well*** into the goals to ***clarify external requirements*** and to ***satisfy external stakeholders.***

5.1.1.1. Clarifying external requirements for the project

External requirements for the project stem from two main sources: (a) the client organisation whose desire for a particular software product has initiated the project, and, (b) the contractor organisation whose desire to accept and carry out the project has established it. These organisations may have different views on what the project is to achieve, or how it should achieve it, thus imposing often contradictory demands on the project manager and the project at large. Furthermore, their requirements may or may not be clearly stated in formal documentation or during meetings, or what is clearly stated may not precisely correlate with requirements which would enable all stakeholders' expectations to be fulfilled. Thus, clarifying external requirements for the project, as and when they arise, is a major goal for the project manager, which is often promoted through negotiation with the relevant stakeholders (client organisation and/or own organisation). If it is not achieved, the project has little chance of being successful as the goal to ***satisfy external srakeholders*** will not be met either.

Typical project management activities which are motivated by this goal include: analysing risks, specifying requirements, confirming requirements and deliverables, negotiating changes, and setting up liaison.

5.1.1.2. Satisfying the external stakeholders in the project

The clarification of external requirements for the project is only one side of successful management across the interface between the project and its environment. Satisfying external stakeholders involves more than just producing deliverables which meet expressed requirements; deliverables should meet or exceed the expectations of stakeholders, and demonstrably so. For instance, reports which indicate to the project manager that requirements have been met are not sufficient if the results reported are expressed in a language the client does not understand, resulting in misunderstandings and womes amongst external stakeholders concerning the project.

Furthermore, producing the required deliverables even in a form which is completely acceptable to the client is not often enough to fully achieve the goal of satisfying external stakeholders: good relations with the client, maintaining good will both with the client and other managers within one's own organisation and such like are important elements of achieving this goal. Publicity is also important especially in maintaining a good image for the public, thus making it easier for external stakeholders to justify their support for the project, or the delivery of desired resources required by the project team. If this goal is not achieved, even the best planning may add up to no more than a cancelled

project.

This goal is ever-present in the manager's considerations. It provides a kind of filter through which the success or failure of nearly all his activities pass. All managerial activities are at least partially guided and motivated by this goal.

5.1.1.3. Means of ensuring successful transactions with the project environment

The match between the requirements of external stakeholders from the project and the deliverables from the project to these external stakeholders must be ensured if the goal to transact with the project environment well is to be achieved. Ensuring this, however, is not always easy as it is not the result of ***one*** successful transaction (that is, "get the requirements at the beginning and provide the results at the end of the project"). Requirements often change in the process of project evolution, and results have to be provided to these external stakeholders continually throughout the lifetime of the project. To ensure that requirements are fully satisfied in the products of the project, provisions must be made by the project manager for ***verification*** and ***qualify assurance,*** which can guarantee that requirements for the project are satisfied through the deliverables of the project.

Verification and quality assurance take place in various ways a different phases of development. At the early stages of a project, client involvement is crucial to define and flesh out adequately the requirements and specifications for the project (Licker 1985); once the project has been established, the continuing use of prototyping, or other techniques (e.g., using a prospectively written user's manual; Howes 1988) which enhance the development of requirements specifications can be very successful in shaping the manager's planning. Some issues which need to be checked in connection with verification and quality assurance within the manager's planning are as follows:

- whether adequate provision has been made for reviews, walkthroughs and/or inspections during the design and development stages in order to avoid pitfalls later on;
- whether testing and system audits have been planned in order to provide verifications at the earliest appropriate points in the development cycle as it is far less expensive and much easier to remove defects in the early stages of product development than later on;
- if there is a continuation of verification activities through the stages of product development and testing in order to ensure that the product corresponds to what the client actually wants rather than to what he said he wanted,
- what the possibilities for accessing and correcting the product after the delivery date are.

Considering whether such concerns have been taken into account in the manager's plan or not is an essential element in achieving the goal to transact with the project environment well.

5.1.2. Managing the project work system well

Unless the current external requirements for the project are clarified to a satisfactory degree, the project is unlikely to be successful (e.g., Schwartz 1975). However, if they have been clarified, the success of the project will exclusively hinge on the manager's ability to meet the goal to ***manage his project work system well.*** This effectively entails the achievement of three goals: (a) to ***create and implement a project plan*** which will translate the requirements of external stakeholders into deliverables from the project, (b) to ***maintain the relationship between this plan*** *and* ***the reality of the project*** during the execution of the project, and, (c) to ***facilitate project work*** in all its facets. What achieving these goals entails is discussed below.

5.1.2.1. Creating and implementing a project plan

The project manager is usually afforded considerable discretion on how he creates and implements a project plan which meets the constraints imposed by the achievement of the goal to transact with the project environment well. The goal of plan development is to identify the operational tasks which will meet the requirements through transforming the types of resources ***available to*** the project (mechanical, financial, human, etc.) into the types of resources required ***from*** the project by other stakeholders (i.e., its deliverables). Constraints on time, finance and resource availability (including resource levelling) are usually set and need to be observed in order to meet this goal while also meeting the goal to transact with the project environment well. Since individual tasks that may be identified have to be linked together in order to create the plan, the goal to create a project plan also reflects the need to identify the products which characterise the essential contents of the inter-task links.

Implementation of the plan requires the existence of a ***coherent task decomposition*** (workbreakdown) on different levels and ***consistency*** in the developed workbreakdown structure. This is essential for the project manager to be able to outline the make-up of his project and assure that all details are attended to in an ordered, efficient, and timely manner thus providing a good basis for the project as it is being executed. The workbreakdown structure used in the planning should reveal the relationships between the variables the interdependence of which needs to be taken into account in estimating (e.g., Putnam 1978) and pave the way for reasonable staffing and scheduling estimates (Howes 1984).

It is also important that, through what-if simulations, the manager examines whether the implementation of the plan is likely to satisfy constraints and

requirements over the range of situations it is envisaged to cover. Appropriate real resources must be allocated to the various tasks (meeting availability constraints) for the plan structure which represents the planned project work system to form the basis for the project work which will be carried out in reality.

Some of the issues pertaining to a good project plan that the project manager should watch for in assessing his plan (and, thus, attaining this goal) are as follows†:

- if there are omissions or discrepancies in the requirements specification used as a basis for the task decomposition;
- how complete the task decomposition used as a basis for planning is;
- how well the complexity of individual tasks and the interfaces between them has been estimated;
- how great the interdependencies between parallel paths in the task schedule are;
- how accurate the estimated completion times for activities are likely to be (including time reserved for handling errors);
- how good the resource estimates are (e.g., what provision has been made within them for people going sick; how much dependency there is on individual knowledge owners; if there will be time-sharing of resources with other projects which could result in conflicts).

Typical project management activities motivated by this goal include: specifying requirements, negotiating changes, standard and actual planning, tailoring the management process, confirming requirements and deliverables, and activating the project (triggering the plan).

5.1.2.2. Maintaining the relationship between planned project work system and reality

Under the goal to ***create*** and ***implement a project plan,*** the ensuing plan is viewed as a ***changeable*** entity (i.e., its structure may be developed and reused, its parameters may be manipulated in what-if simulations and so on). However, under the goal to ***maintain the relationship between the planned project work system and the project reality,*** the structure and parameters of the plan are ***fixed*** in a single, coherent, instantiation which can be used as a criterion against which observables within the actual project work system can be mapped and compared. The aim of the mapping is to find sufficient and appropriate observables to permit an adequate test of the relationship between plan and reality at any point of interest‡. This comparison aims to identify the degree of conformity evinced by the tests carried out since an observed

† The issues listed here relate to a more global level of assessment of the project plan than that described in section 4.2.2.3 above.

‡ We will address this problem in detail in chapter 9.

discrepancy may result from more fundamental, less observable problems in the relationship between plan and reality. Furthermore, discrepancies thus observed need to be diagnosed and action taken to remedy the encountered problems if this goal is to be attained.

Managing milestones provides the project manager with an accurate means to track progress and make any necessary adjustments to schedules and budgets. As milestones are tangible (the outputs of tasks, not the tasks themselves), they can facilitate tighter quality control as intermediate products (Sweet 1986, Turner 1984). Furthermore, a milestone-based monitoring method provides a way of periodically evaluating the risk of the project as it proceeds and helps the project manager make appropriate decisions about its continued development. In attaining this goal, the manager needs to consider, for example,

- how well the global history of the project can be monitored and reviewed (e.g., so that the overall estimates, and effort spent, can be examined as a function of time);
- if adequate intermediate checking mechanisms have been placed on long timescale activities;
- if provision has been made for adequate monitoring of activities which are likely to go wrong;
- if the recording of progress is likely to remain realistic (rather than be a pretence);
- whether the reporting and review procedures are likely to be effective in giving an idea of what is likely to occur in the future (e.g., reporting exceptions and predictions as well as confirmations).

Anticipating future problems is also part of this goal. This is achieved through simulating future states of the planned project work, starting from observed discrepancies between the plan and reality at a particular point in time. However, it remains within the discretion of the project manager to decide whether problems arising due to lack of conformity between the planned project work system and reality should remain within the focus of this goal (in which case, he would act directly on the real project work system), or whether to re-plan or revise observed values (rather than retain the previously planned ones) for the relevant parameters thus attempting to attain the goal to ***create and implement a project plan*** instead.

Typical project management activities motivated by the goal to maintain the relationship between the planned project work system and reality include: negotiating changes and resources, setting up Liaison, actual planning, quality assurance planning, setting up product change control procedures, setting up monitoring procedures, team building, reporting on progress, diagnosing and remedying exceptions, identifying and predicting discrepancies and exceptions, and handling team problems.

5.1.2.3. Facilitating project work

Project team members with capability for self-management or subteam management can be accorded discretion on how they carry out the work under their responsibility provided their work meets its requirements (typically covered in task briefings, specification documents, etc.). The focus of this goal is on managing the interface between the parts of the project work system explicitly formalised in the plan and those the precise implementation of which is left to the discretion of self-managed units (e.g., individuals, subteams) and coordinating the results of independent and self-directed actions (Brech 1967).

These self-managed units may also consume resources and, thus, need to be provided with an adequate range and auantitv of such resources. In the absence of precise planning for their utilisation, this need may be interpreted in various forms such as provision and coordination of adequate communication links without specifying their precise usage, designing an appropriate work environment (i.e., one able to support and facilitate the execution of the whole range of tasks involved), negotiating agreement with team members on the requirements for their work, and other attempts to motivate team members to adopt a style of self-management which enables tasks to be carried out in a way which meets their requirements effectively.

Project management activities motivated by this goal comprise most of the project management activities which are initiated with the start of the project (including activities which are motivated by the goal to create and implement a project plan). Project management activities which are carried out at the closing of the project phase (e.g., anticipating future use of the project, project archiving) are motivated by this goal not in connection with the particular project but in connection with future projects in the sense that these activities will facilitate project work in future occasions (e.g., through providing information which can help in a realistic planning of the future project).

5.1.2.4. Means of ensuring successful management of the project work system

A major problem in software development which can endanger the achievement of the goal to manage the project system well stems from the fact that project requirements often change during the development itself (either by the client/user or even by the development team members) resulting in unanticipated demands made on the project. Effective ***product change control*** is a means of limiting this kind of disturbance on the project work system. It requires the manager setting up criteria, before the project starts, for allowable changes and providing for adequate coordination, communication and documentation of all changes. While decisions on allowable changes must be made on a sound economic basis (this means considering the economic impact of the re-planning necessary to achieve these changes; Cooper 1984, Ferrentino 1981), economic factors alone should not control this decision. One will also need to

consider, for example,

- what is the likely impact of a proposed change on the technical team (in negative terms, this may involve disruption of their work schedules and personal frustration, as in the case where a deliverable of which the team is proud is scrapped; in positive terms, a proposed change may allow members of the team to escape from working on a deliverable in which they, as well as the client, have lost confidence);
- if, after trading-off potential benefits and disbenefits, a specific change is decided upon, whether it is likely to be done under tight control so that system quality will not be degraded-,
- to what extent the impact of all changes may be reflected in the project schedule;
- what is the nature and degree of on-going client involvement in the development process. (This can be helpful as when the client has a real understanding of the characteristics of the product and its application, together with an understanding of the work requirements of the project team, he is more likely to become an ally in the effort to keep changes under control.)

Another essential ingredient in the achievement of the goal to manage the project work system well is the quality of the manager's leadership of his team(s). Effective leadership requires interaction and communication between the leader and the people he leads. As such, leadership is successful when the manager is considerate of the people he leads while they are committed and enthusiastic about their work (Robertson & Secor 1986).

Leadership behaviour is often described along two dimensions: a task-related one and a relationship-related one (Fiedler 1967, Fiedler & Chemers 1974, Hersey & Blanchard 1977). The former relates to the leader's ability in forming relationships between himself and his co-workers in terms of initiating structure of work, providing patterns of communication, working methods for the group, setting Standards, scheduling work assignments, etc. The latter relates to the nature of his relationship with his subordinates in terms of friendship, mutual trust and respect.

The degree to which either of these behaviours should be exhibited is described as dependent on the maturity of the subordinate. For instance, when the subordinate exhibits low maturity, the manager should tell him what to do; with increased maturity of the subordinate, the manager needs to "sell" to him decisions already made. At the next higher level of maturity of the subordinate, the manager needs to discuss and agree with him what his task shall be via participative management. Finally, a highly matured subordinate will necessitate an interaction where the manager defines the problem that needs to be handled but leaves the analysis and solution to the subordinate, effectively delegating work. Ability to delegate successfully is an important element of the leadership skills of a manager (e.g., Raudsepp 1981, Jenks & Kelly 1985).

However, the degree to which a subordinate is mature is difficult to define or ascertain for sure nor are all persons mature in the same way in all facets of

their personality (Bergen 1986). Furthermore, the way managers behave as leaders is also determined by the constraints imposed upon them by, and the very nature of, the organisation within which they have to work (Gregson & Livesey 1983, Kakabadse, Ludlow & Vinnicombe 1988).

The particular leadership activities required of a software project manager vary during the phases of software development (Kugel 1984, Brooks 1982). When involved in the initial planning and organisation of a project, the manager must demonstrate the ability to forecast, to plan for contingencies, and to make well-reasoned judgements. During the design period, the project manager should lead obliquely and enthusiastically to encourage ideas while gently rejecting inappropriate ones. During the development stage (coding and testing), more control is necessary while, at the same time, the project manager should be shielding his staff from distractions while not allowing his presence to be oppressive as he ensures that nothing is overlooked.

5.13. Advancing expertise

The goal to ***advance expertise*** relates both to the improvement of the ***technical skills*** of the project manager and his team members and to the improvement of ***project management expertise*** through carrying out a particular project

Advancing technical expertise necessitates that members of the project team learn through the project and become and remain conscious of what they are learning from it However, the possibilities of advancing technical expertise that may be available for the team members are often curtailed by the nature of the project itself (i.e., its requirements may demand that team members spend much of their time on work which they consider to be mundane; Fife 1988). The project manager can increase these possibilities by planning the work of his team members in such a way that they are allocated increasing more complex or challenging tasks as the project progresses. Personnel may also be sent on courses which can increase their knowledge about some aspects of the work that they are doing or are about to do on the project Advancement in technical expertise can also be achieved by documenting relevant information on the execution of the project as the project is carried out and archiving this information for the purpose of enhancing future performance on the part of the resources of the contractor organisation.

Advancing project management expertise involves abstracting material from the experience gained through running the particular project which can be used in the future either by the same manager or by other managers when they are faced with similar projects. Experience gained through running a particular project can be communicated both in a semi-formal way (e.g., through a well-documented project history), and in an informal one (e.g., through personal communication with interested parties). However, it is not often the case that the project management experience thus gained will be incorporated within the company prescribed standards or procedures unless it can be described as an exception to an already existing instance. In the absence of facilities for specific

tailoring of management methodologies to suit particular projects, it is more likely that this experience will be used as statistical data to improve company managerial standards or as a source of information for future projects rather than part of prescriptions for future projects.

Typical project management activities particularly motivated by these two goals include: documenting the project as a case study, debriefing the project team, anticipating future use of the project, project archiving, and actual planning of the particular project (the last in conjunction with the goal to advance technical expertise).

5.2. Achievement of goals through activities

It would be possible to proceed to refine project management goals further in order to impose more structure in the discussions of the individual goals compared with those given in the previous sections. However, goals, on their own, only represent hopes and exhortations. Refining them does not bring us any closer to achieving them in practice. To do this, we need to consider the project management activities which are motivated by them and which are aimed at meeting them. Moreover, the relative importance each goal will have in the various phases of the project will vary as will the activities that could appropriately serve each goal.

Essentially, goals represent ***desired ends*** and management activities represent ***intentional means*** to achieve those ends. However, it is not guaranteed that the means, when implemented, will bring about the desired ends. Thus, it is important, at the completion of an activity, to test for the achievement of the goal which triggered it.

Moreover, although an activity may have been motivated or triggered by a particular goal, its successful completion may have also served some other goal. For example, successful creation of a project plan can also facilitate project work, as it enables the team members to have a better idea of the significance of the particular tasks and responsibilities assigned to them.

Conversely, when the project manager is preoccupied with achieving the goal which triggered an activity, he may inadvertently carry out that activity in a way which creates a side-effect which prejudices the achievement of another goal. For example, devoting all his time to monitoring the project tightly in order to maintain a close relationship between the plan and reality may lead him to overlook the need of reporting about the project progress to the client thus subverting the goal to satisfy external stakeholders. Hence, at the completion of an activity, it is important to ***test*** for the achievement not only of the goal which motivated or triggered it, but also of the achievement of other goals as well which were achieved or subverted as a result of carrying out the activity.

Table 5.1: Summary of the relationships between goals and activities

Activities	Goals						
	1	2	3	4	5	6	7
Actual planning		?	T	?	?	?	
Analyse risks	T	?			?		
Anticipate future use of project						T	?
Confirm reqs and deliverables	T	?	?	W	?		
Debriefing project team		?				?	T
Design working environment		?			T		
Develop case studies						?	T
Diagnose & remedy exceptions	W	?	W	T	?		
Handle team problems		?	W	?	T		
Handover results		T					
Identify & predict discrepancies		?	W	T	?		
Negotiate changes	?	?	?	T	W		
Negotiate resources		?	W	T	?		
Negotiate working conditions		?	W		T		
Project archiving		T				?	?
Project initialisation	T				W		
Publicise project		T		W			
Quality assurance planning		?		T	?		
Report on progress		T		?	?		
Set up change control procedures		?		?	T		
Set up liaison	?	?		?	T		
Set up monitoring procedures		?		T	?		
Specify requirements	T		?		?		
Standard planning		?	T		?		
Tailor management process	?	T	?	?	?		?
Team building		?		?	T		
Trigger plan		?	T	?	?		

Key to ruble:

Goal 1	Clarify external requirements
Goal 2	Satisfy external stakeholders
Goal 3	Create and implement a project plan
Goal 4	Maintain the relationship between plan and reality
Goal 5	Facilitate project work
Goal 6	Advance technical expertise
Goal 7	Advance project management expertise
T	Triggering goal
W	Watch for the possible subversion of this goal
?	Goal which may be achieved as a side-effect

Table 5.1 names some typical project management activities which we have discussed in the previous sections as related to the achievement of the various goals. We will discuss how each of these activities may be modelled in chapter 7. However, for our present purposes, table 5.1 merely indicates the goals which could mgger each activity, other goals which may be advanced through the activity's execution, and goals the achievement of which needs to be safeguarded against potentially damaging side-effects of the execution of the activity.

Achieving the project management goals we described in this chapter is not an easy enterprise. All the previous chapters of this book have amply demonstrated this by discussing problem areas where risk to the achievement of project management goals may materialise. We could list more problem areas but still fail to address ***everything*** which could upset their achievement. Rather, when identifying and developing models for understanding project management activities, our objective should be to devise tests for ascertaining the risk drivers† which can upset the achievement of project management goals and provide suggestions for associated risk management activities aimed at ensuring that the manager ***can cope*** with their threats as and when they arise.

5.3. Perspectives on the project work system and the project environment

No project management activity has much chance of being carried out successfully in the absence of information about relevant aspects of the project work system and the project environment. However, material which can describe the present reality of these systems will exist in diverse locations in the real world. The material to be found at any locality is likely to be organised in a form appropriate for the purposes of the person who is responsible for a particular part of this reality. For example, person A knows he works on task X which has to fulfil particular functions within a particular timeframe. This organisation of the material is sufficient for the work purposes of person A (so that he knows what he has to do when) but it is only *locally* sufficient for the purposes of the manager, since it structures information for him only about the progress of person A on task X (e.g., it provides information on its actual start-date, planned end-date, percentage of work on the task completed so far).

The way the manager needs to organise information from the real world of the project work system and the project environment, his ***internal*** organisation of this material, is shaped by his own own range of responsibilities and consequent concerns, which extend across the entire project work system and take in part of the project environment The goals which the project manager wishes to achieve both shape the activities he decides to carry out and guide the ***views*** he takes on the project work system and the project environment to collect and interpret the required information for his own activities. Conversely, each view

† The concept of a *risk driver* is discussed in detail in *sections* 3.2.2 and 8.2

taken on the project work system and project environment needs to organise the relevant information in such a way that the project manager's activities can be carried out successfully and his goals achieved.

To reflect the idea of the ***view*** that the manager needs to take on the project work system and the project environment, we employ here the concept of ***perspective.*** This concept capitalises on the analogy from visual perception: the project manager, viewing his project, is likely, at any point in time, to take only a partial view on the full workings of the project work system and the environment within which it is situated. However, what is viewed is seen ***in perspective.***

The notion of perspective employed here is essentially non-geometric. That is, it does not refer to a two-dimensional projection plane of a three-dimensional physical system. This is *so* because we are interested here in ***conceptual*** rather than physical modelling of the systems in question (Checkland 1981). Nevertheless, the analogy is useful in that it holds in terms of certain properties the notion of perspective implies. It enables:

(a) ***Reduction of complexity,*** in that the description of the view of a complex system seen in perspective is less complex than would be a description of the full system. Yet, information is lost ***selectively*** across all dimensions of the system, rather than information in some dimensions being completely excluded (as would be the case of a section through a complex system).

(b) Salience of the foreground, in that, in a scene represented in perspective, objects in the foreground (i.e., closer to the viewpoint of the observer) take up a disproportionately large amount of the field of view. In the non-geometric interpretation of the idea of perspective, this may be expressed in terms of ***saliency.*** That is, objects in the project work system and the project environment which come to the fore due to the current concerns of the project manager tend to be more salient to him. In focusing on the more salient objects, however, the manager may overlook the fact that they mask consideration of other objects which may be important to consider as well *so* that his management activities achieve their goals.

(c) ***Gaining information through changing the viewpoint. In*** practice, the perspective initially adopted by a project manager in carrying out a management activity will generally depend upon his immediate goals in selecting that activity. However, adopting a single viewpoint on the project work system and the project environment will invariably lead to the prioritisation of operations on certain objects in these systems and to the neglect of others. It is quite possible, however, for the manager to change his viewpoint, thus gaining a new perspective on the scene addressed by the current management activity. This new perspective may provide additional information about objects "visible from another angle" in the first perspective such as information which refers to different characteristics of those objects. The new perspective may also provide information about relevant objects which were "invisible" fmm the viewpoint of the first per-

Although, in theory, any number of perspectives may be possible as there are an infinite number of viewpoints one can take in viewing a system, the discussion in the earlier chapters of this book points to ***four reference perspectives*** which seem to be commonly adopted by managers in describing objects of interest in the project work system and the project environment. These four reference perspectives are:

(1) the ***project function perspective*** which relates to the function of the project as a whole, including the product which is the end result of the project, issues related to client satisfaction with the product, the reputation of the contractor organisation concerning quality of product, the contract and the binding conditions it reflects concerning the functional specifications for the product, delivery time, quality standards expected, available budget, etc.;

(2) the ***activity perspective*** which relates to the work that needs to be carried out within the project work system for the project to achieve its aim (i.e., to produce specified products and deliverables within the constraints identified within the project function perspective);

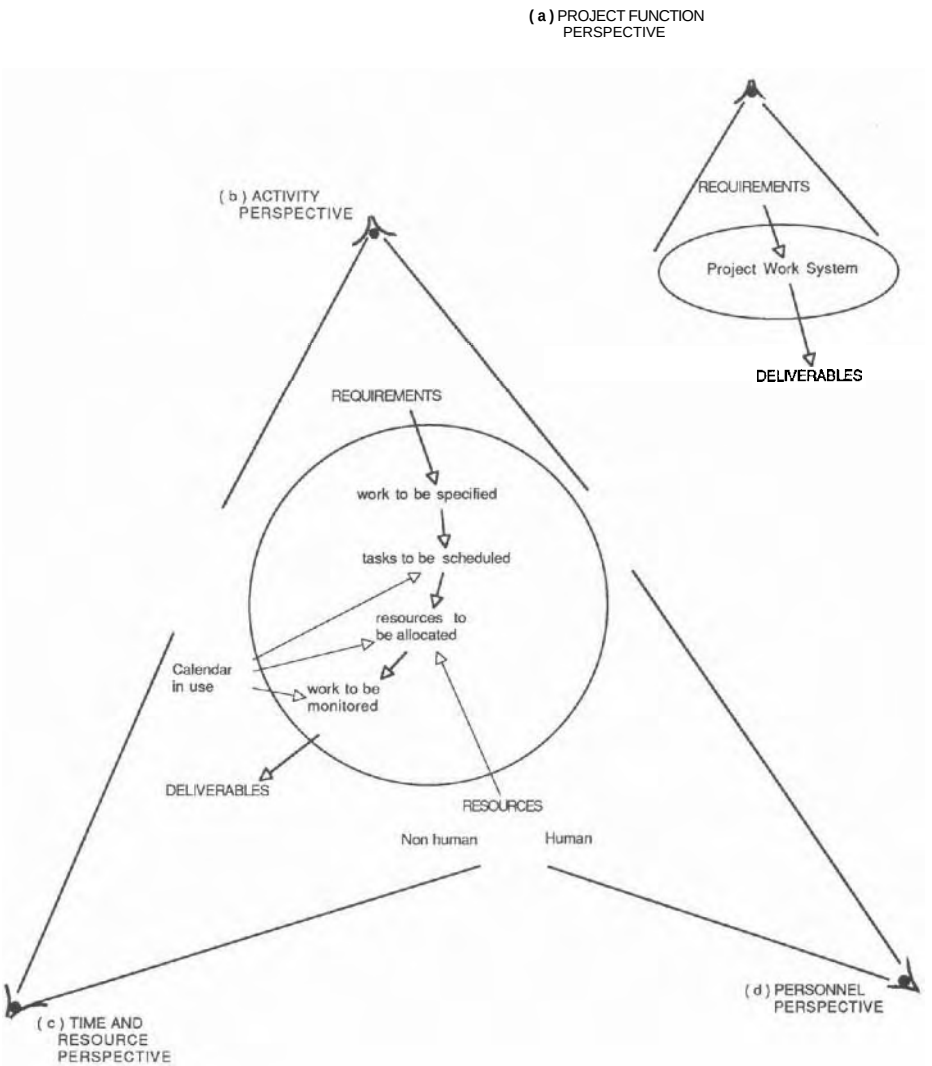
(3) the ***time*** and ***resource perspective*** which relates to resource aspects of the project in terms of time (in calendar terms and effort terms), and in terms of physical resources such as finance, personnel, machines and machine time, space, etc. (instrumental to achieving what has been identified within the project function perspective); and,

(4) the ***personnel perspective*** which relates to the concerns of human resources as they carry out the roles allocated to them within the project work system (e.g., focusing on the satisfaction or stress they may experience through working with certain other people, on a particular set of tasks, etc.).

These perspectives function as organising principles for information which may exist in unorganised form within the real world of the system of interest to the project manager. All four can provide the full range of information he needs to carry out his project management activities. Figure 5.2 illustrates what may be viewed from within each perspective. However, not all four perspectives need to take in both the project work system and the project environment, for the reasons we discuss below.

Within the ***project function perspective,*** the project work system may be viewed in terms of requirements from the project environment and deliverables to the project environment (figure 5.2a). In this sense, the project function perspective takes in aspects of the project work system as well as aspects of the project environment, so that the manager can compare the requirements set in the project environment with anticipated deliverables from the project work system. In general, taking this perspective informs all the project management activities which involve the manager in transactions with both the project work system and the project environment.

Figure 5.2 (a, b, c, d): Perspectives on the project work system and the project environment



Within the activity perspective (figure 5.2b), the foreground is occupied with the interpretation of the requirements from the project environment. This sets the scene for the work to be specified within the project in terms of tasks which, further in the background, will be scheduled and have resources allocated to them, so that deliverables can be produced and then be delivered to the project environment. The deliverables may serve as resources utilised by other systems in the project environment, but the organisation of information on how this is done is not the responsibility of the project manager. Therefore, the activity perspective is principally a perspective on project work and does not need to focus in any way on work to be done in the project environment.

Within the time *and* resource perspective (figure 5.2c), the foreground is concerned with the definition of the calendar to be used in timing project work resource availability. It brings into view resources in the project environment as well as the deliverables from the project, here seen as resources delivered into the project environment at specified times. Taking the time and resource perspective on the project environment thus allows the manager to locate potentially available resources that exist therein (whether within the client organisation or the contractor organisation), define the budget for the project (i.e., financial resources provided to the manager), and the timescale of the project. This information enables all his project management activities which have to do with negotiating resources that may be eventually made available to his project. Taking the time and resource perspective on the project work system also helps the manager identify actual, rather than potential, resources to the project enabling his activities which have to do with planning of how work may be carried out within the project work system, whether deadlines can be met, etc.

Within the personnel perspective (figure 5.2d), the foreground is concerned with the human resources assigned to various tasks and the roles they play within the project work system. In this perspective, whatever makes a human resource different from a non-human resource is the focus of interest. It is a perspective which needs to be applied in viewing the project work system but does not take in the project environment as the concerns of the manager which could be expressed within this perspective are limited by his range of responsibility (i.e., the persons whose quality of life he can really affect).

In chapter 4, we discussed in some detail issues specifically related to project personnel which need to be taken into account in managing a project (e.g., issues to consider in team building, needs and expectations of project team members). All these issues may be brought into focus within the personnel perspective. The success with which they are actually taken into account by the project manager is the major determinant of his perceived ability to manage a project really well from the viewpoint of all personnel concerned. We described, for instance, in section 4.2.3.1, the disastrous consequences that can result from viewing personnel working on the project merely as resources, that is, taking only a time and resource perspective on the project work system rather than a personnel perspective as well.

In summary, perspective, as an organising concept, facilitates our understanding of the way a manager needs to, and does, view the project work

system and project environment in order to carry out his activities and meet his goals. As an **organising** concept (like the goals and activities we discussed above), it necessarily reflects a simplified view of what actually happens in reality. It organises our understanding of the reality of project management although, in that reality, the **information** needed by the project manager is organised according to what could be seen if all the perspectives were simultaneously employed. The advantage of treating each perspective separately is that we can identify particular elements in the reality of the project work system and the project environment which may be focused upon at any particular time within the perspective actually employed by the project manager. Thus, perspectives provide a starting point, together with goals and activities, in the **scoping** of the modelling enterprise which we shall describe in the following chapters.

Chapter 6

Modelling the project management process: Approach and concepts

The previous chapters presented an outline of the process of software development project management, starting with a discussion of how constraints are placed on the project and the project manager during project establishment, describing what is involved in managing a project, and concluding with a discussion of the project management goals which motivate the activities of the project manager. This outline was, by necessity, informal and discursive as it aimed at scanning through, rather than detailing, the domain which concerns us here†. It described material only at the degree of detail that was necessary and sufficient for providing information needed for the development of a model of the project management process.

This chapter provides the transition from such a discursive discussion of the project management process to the modelling of this process. Its purpose is to introduce the reader to the modelling concepts used in this book and to describe their use in developing models of the various systems relevant to project management and its concerns.

6.1. Models and model building

Model and model building are terms which have become familiar in the literature of a great deal of diverse disciplines ranging from the "soft" Social Science disciplines (e.g., a model of a social system) to the "hard sciences such as Physics (e.g., a model of the atom) or Computer Sciences (e.g., a database model). In all instances, the term *model* is used to refer to some ***abstraction*** of certain elements in the reality of the system the model is meant to represent (the ***object system*** of modelling) and a representation of the relationships

† See, for example. Keen (1987) or Sneed (1989) for more detailed expositions of this domain.

between them. The difference in the nature of these models is usually located at the degree of detail they use to address the object system, the model representation formalism they employ, and the method through which the model is arrived a t

Models, as pure abstractions, are without intrinsic meaning, as is any general calculus (Carnap 1938). Meaning is achieved through ***interpreting*** the model and, thus, restricting the generality of the elements and relationships in the model through making references to objects, or classes of objects, which are familiar (or, at least, identifiable) in the domain of application of the model. These objects, and their relationships are thus considered to constitute the ***object system*** being modelled. ***A*** representation of part or all of the object system, showing the relationships between particular objects at a particular instant (in model development, rather than in real time) is known as an ***instantiation*** of the model. Hence, a model of an object system can have any number of instantiations.

Generally, the motivation behind model building is to provide a means and a pathway to the understanding of the real phenomenon or object system of interest which is represented through instantiating the model. However, each model is necessarily built on the basis of certain assumptions about which aspects of the system it aims to model are ***essential*** and therefore should be included in the model, and which elements of the system are ***not essential*** and therefore should be left out of the model. Cleland and King (1972) argue that,

> "The primary value of a model is that it leaves things out. If all models were "perfect" in the sense that they included ***all*** aspects of the real system, there would be no models but, rather, simply reproductions of real-world systems." (p. 48)

Since assumptions guide the decision of what to leave out of the model and what to include, in any model building enterprise, the resulting model is no more valid than are the assumptions on which it is based. If any of the assumptions are called into question, so is the validity of the model.

Moreover, on the basis of different assumptions, which may appear equally valid, various models may be generated, instantiated and accepted as addressing the same modelled phenomenon, but be incompatible with each other, that is, it is not possible to make an exact mapping between the formalisms used and the elements chosen in different models. This is particularly true for models which purport to explain social phenomena where the modeller's assumptions about an essential part of the model, ***man,*** necessarily underly the process of model development. Here, it is especially important that the underlying assumptions be made explicit at the outset.

In modelling the project management process, we start from the basic assumption that an essential part of this process is the human agent who carries it out (i.e., the project manager), as are the human agents who influence how it is carried out (e.g., the project personnel, the client, other managers in the organisation). Our discussion of the project management process in earlier chapters sewed to refine this basic assumption and shape the process of model

building by addressing these agents as active interpreters of the social world rather than mere recipients of its influence.

Ultimately, model building is guided by the ***purpose*** for which the model is intended to be used. For example, a model built for the purpose of ***replacing*** the project manager in his activities through a computer-based system would aim at developing a system specification which, when implemented, would automate all the activities which make up the manager's practice. Its focus and the selection of the elements to use to model the project management process would be rather different from those of a model built for the purpose of discovering how best to ***support*** rather than replace the project manager in his activities and transactions (Landry, Pascot & Briolat 1985, Humphreys 1990).

6.1.1. Precision, refinement and formality

Models are often qualified by adjectives which reflect some characteristic or some aspect of the model. For example, we find models described as structural, mathematical, conceptual and so on. The choice of the adjective may be made by the modeller, or by a user or reviewer of the model, according to the particular characteristic of the model he wishes to stress to differentiate it from other models of the same phenomenon.

One could argue that all models are conceptual, rather than real, and that all models are structural by definition (as a model implies the existence of some form of structure), although not all models are mathematical models. However, the word ***conceptual*** is usually used in contrast to the word ***mathematical.*** The distinction which is actually intended concerns the use of a ***low precision*** modelling formalism employed in conceptual models which mes to capture the form in which these models were "intuitively thought up". This is contrasted with a (mathematical) model whose initial form has been worked over and precised to the stage where its degree of coherence and consistency permits a mathematical treatment of the relationships between the elements of the model.

The concept of ***precision*** in modelling should be carefully distinguished from that of ***refinement.*** Refinement involves decomposing a unitary element of the modelled representation of the object system to show its internal structure. Thus, ***refining*** a model refers to the process of developing a more ***fine-grained*** structure in some part of the model in order to capture more detail in subsequent instantiations of it. This process of refinement should not be confused with the natural language meaning of the term as a kind of purification of the substance of the model (by analogy with chemical refinement of sugar, etc.), which does not apply here at all.

Refinement of the model structure should also be distinguished from refinement of the language in which the model is expressed. To avoid confusion, we will always refer to the latter process as ***increasing*** the ***degree of formality,*** rather than refinement. Increasing the degree of formality of the language used to express a model may offer the possibility for building future instantiations exhibiting greater precision than that possible with the current

language. However, translating a current instantiation into a more formal language does not, in itself, alter the degree of refinement at which the instantiation is represented.

It is important to realise that the process of refining the structure of a model may effectively ***lower*** the overall degree of precision of that model. This will be the case when the modeller does not have enough information for particular elements, at the detailed level required by the refinement, to specify the internal structure of all elements at the same degree of precision as was previously possible in modelling their external relationships at the "coarser" level of modelling. Conversely, the process of refinement, on its own, does not increase the degree of precision of a model: it only reveals greater detail.

6.13. Static versus dynamic models

A distinction which is sometimes made between models is one between ***static*** models and ***dynamic*** models Typically, the qualification ***static*** is used for models developed to represent states of an object system while the qualification ***dynamic*** is used for those which have been developed to represent processes of the system, the latter, by definition, being assumed to be a simulatable model.

In fact, a dynamic model always entails a static model. Any model which offers the opportunity to simulate processes occurring in the object system must distinguish, in a fundamental way, between passive elements represented by ***places*** or ***states*** in the model and active elements which are accorded the agency to effect ***transitions*** between those places or states. The active elements may be described as activities, channels, functions, transformations, and so on, according to the process modelling formalism employed.

Sometimes, further divisions into element ***types*** are made according to the particular roles that different types of elements may play in the model, but such distinctions only refine, rather than supplant, the basic distinction between active and passive elements. However, models which correctly identify the active and passive elements in the object system to be modelled, but do not provide precise ***rules of change*** to describe the conditions under which, and the manner whereby, the active elements transform the contents (or "markings") of passive elements, are adequate as static models but not as dynamic models, even if the relationships modelled indicate ***possible*** transformations and transitions.

A model deserves the label "dynamic" only in the case where the rules of change are sufficiently well worked out and coherent to indicate exactly ***which*** transitions and transformations will occur ***when*** (i.e., under what enabling conditions, set ***a priori*** through the "initial marking" and transformations and transitions consequently executed in the model). Thus, moving from a static model which is adequate to represent the static properties of the object system to one with dynamic capabilities involves, essentially, an increase in the ***precision*** of the modelling formalism which is consistently employed in creating the modelled representation.

6.13. Generative versus generic models

A further distinction that is often made between models is the one between ***generative*** models and ***generic*** models. Following Brodie (1984), we can consider a generative model as a collection of well-defined concepts which help one consider and express the static and dynamic properties and constraints for a range of uses of the model within the domain of the object system. Each use will involve the generation of a particular ***instantiation*** of a representation of the model. While each representation is, of necessity, temporarily closed (so that it can be represented), the union of the set of representations which could possibly be generated through the use of the modelling concepts remains open.

In purely generative modelling, each instantiation of the model is built from scratch through the use of the appropriate modelling concepts according to immediate requirements. Often, though, it is found in practice that a number of applications will involve building instantiations of the model some parts of which have the same structure as similar parts of a number of other instantiations of the model. In such cases, it is useful to identify these recumng parts and define them as ***pre-structured components.*** These components can then be stored in a library and may be incorporated (or "tailored") into any particular instantiation as and where required, thus obviating the need to re-generate them every time they are needed from scratch.

For a generic model, however, the ***core*** of the representation is pre-structured. This is usually achieved through grouping elements that may be instantiated into classes defined in terms of ***pre-structured entity types.*** Then, each instance that is made of the modelled entity described by the class will necessarily share the same generic specification of its attributes, relationships and constraints. A particular instantiation of the model can thus be made by assembling instances drawn from the classes describing the entities to be represented within the instantiation. As the relationships between the various entities are pre-specified (as part of their class definitions), there is no need to employ modelling concepts to generate the model structure each time: the structure is automatically given as the instantiation is made. It may be necessary, though, to verify the coherence of the structure of the resulting instantiation by checking the consistency and reciprocity of the relationships between the entities incorporated in it.

Direct comparisons of different instantiations of a generic model may appear to reveal that they have different structures, but this is simply a result of having instantiated different parts of the generic structure, leaving the non-instantiated parts "invisible" within the structure of the instantiation. Hence, the union of the set of representations which could possibly be built through instantiation within a generic model is closed at the level of the generic model itself.

The development and use of a generic model in this way is usually satisfactory in cases where the generic structure can be discovered ***a priori*** and remains stable across all applications in the domain of the object system. Sometimes, however, this requirement is only approximately met in reality, leading to the need to ***customise*** the generic model to address the requirements of a

particular type of application better. This requires the use of modelling concepts to re-generate specific components within the generic model. However, these can then no longer be considered to be truly generic as the model representations built by instantiations within differently customised versions of the same generic model may be only partially compatible.

One way out of the potential confusion that this can create is to use the generic model to build instantiations over the whole range of applications employing the same structure described at a coarse level then, through refinement, to generate the internal structures of model elements in a form which is specifically appropriate for a particular application, leaving the external (unitary) representation of each element unchanged within the generic model.

Generative modelling is most appropriate in cases where there is likely to be unforeseeable variation in the structure of the instantiation of the model which will subsequently need to be built. Conversely, the generic approach to modelling is more appropriate in cases where the generic structure of each instantiation of the model can be ascertained ***a priori.***

However, it is often neither desirable nor necessary to adopt a purely generative or purely generic approach as each can contribute different advantages to the modelling endeavour. Thus, when developing an instantiation of a model top-down within a generative approach, generic features can be brought in bottom-up through incorporating pre-structured components selected from a library, eliminating the need for unnecessary repetition of effort. Similarly, when developing an instantiation of a model top-down within a generic approach, generative features can be brought in bottom-up through making customised refinements, thus counteracting the necessary restrictions imposed by a generic model.

6.2. The initial stage in modelling the project management process

The initial stage in any modelling enterprise involves collecting observations, information, knowledge, etc. from the real world of the phenomenon of interest up to the point that one can describe how the phenomenon is initiated, what influences it, how it happens, what its consequences are, and so on. This provides for a discursive account of the phenomenon of interest organised in a more or less linear way (usually through time), like a script (Schank & Abelson 1977).

The previous chapters of this book can be viewed as providing, to some extent, a script of this kind. This script helped us scope the domain of interest (i.e., project management), identify the various systems of interest to the project manager (i.e., project environment and project work system), define what motivates the project management process (i.e., the project management goals) and describe the various project management activities which comprise it. Our aim is to model the project management process, and, as such, our primary object system of interest is the project management system. Only to the extent

that for it to function it needs information from other systems will these systems be considered as object systems of interest as well.

6.2.1. Defining the scope of project management process: modelling

In chapter 2, we identified the object systems relevant to our task as:

(1) the software development ***project work system*** which represents the means by which the project will be carried out and produce the required results;

(2) the ***project management system*** which operates on the project work system and manages it so that it can fulfil the project objectives; and,

(3) the ***project environment*** which comprises all the external systems that may affect the project, such as

   (a) the ***contractor organisational system*** within which the project is constituted and which provides the resources the project needs and has requirements concerning the way in which the project may be implemented and managed;

   (b) the ***client organisational system*** which provides the requirements of what is to be developed and expects satisfactory returns from the project;

   (c) any ***supplier organisational system*** which provides the project with necessary external resources for carrying out its activities;

   (d) the ***subcontractor organisational system*** which produces deliverables needed by the project and may need deliverables produced by the project in order to do so; and,

   (e) other ***external organisational systems*** which often provide regulations and guidelines which serve as project requirements and resources.

In chapter 5, we discussed the following seven goals which the project manager seeks to achieve through management activities involving transactions with these object systems:

(1) to clarify external requirements for the project;

(2) to satisfy the external stakeholders in the project;

(3) to create and implement a project plan;

(4) to maintain the relationship between the planned project work system and reality;

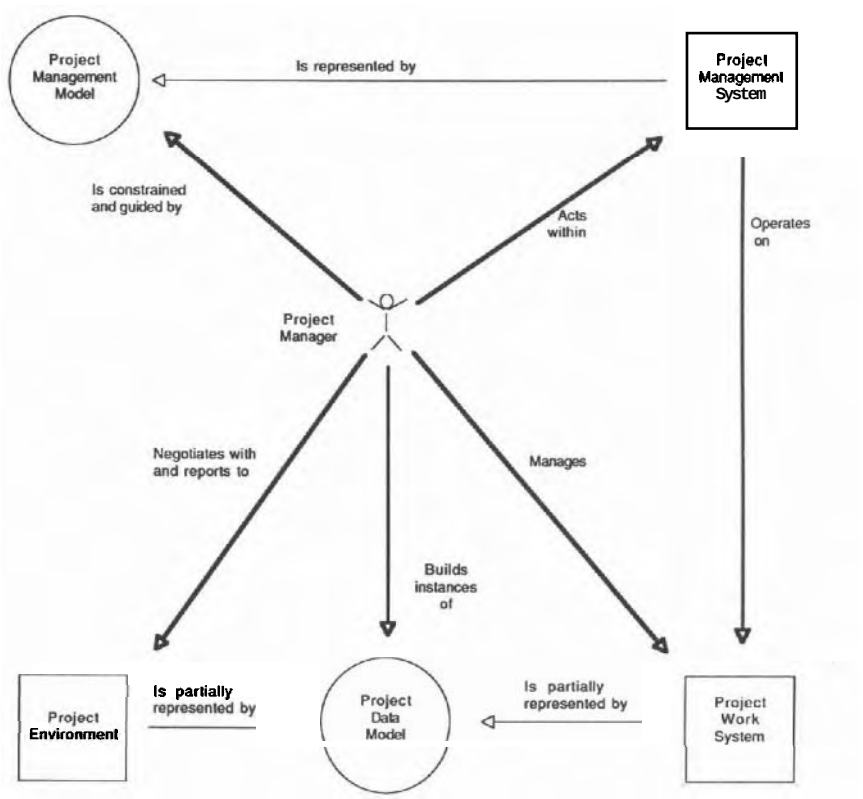
(5) to facilitate project work;

(6) to advance technical expertise, and,

(7) to advance project management expertise.

Figure 6.1 illustrates the role of the project manager in terms of the general nature of his transactions with the project management system and project work system in seeking to achieve these management goals. The object systems within the project environment are not specifically identified, as it is necessary for the project manager to have access only to those parts of the environment which have a direct impact on the project. These can be generally interpreted as either potential ***resources*** (humans, machines, finance, information, contacts, office space, etc.) ***or requirements*** (for the software system to be developed, standards to be observed, constraints on the management methodology to be employed, etc.).

Figure 6.1: Roles of the project manager in relationship to the principal object systems and their models



The manager's direct, behavioural, transactions with the project environment are limited to reporting and negotiating, as he has no agency to act unilaterally on elements which comprise it as these are, normally, outside his responsibility. Conversely, his responsibility to manage the project work system is achieved through his management activities which, collectively, constitute the project management system. They ***operate*** on the project work system, controlling, coordinating and re-organising its elements, and they are represented in the ***project management model.***

Any model purporting to offer a comprehensive, integrated representation of processes and states of an object system has to offer facilities for both a ***static*** view, showing the general structure of its elements, and a ***dynamic*** view, showing the nature of the transitions from one state to the next. Thus, in developing the project management model, we will need to model both the state and the process parts of the project management system since the manager has responsibility for both carrying out the ***process*** of project management through his activities within the project management system, ***and*** being well informed about the ***states*** of the project management system (e.g., what activity he carried out last, what activity he can possibly carry out now).

However, in developing a model of the project management process, our concern with the other object systems of interest (i.e., the project work system and the project environment) need not extend to building a fully comprehensive model of each object system, offering both static and dynamic views on its processes and states (Yeh ***er al*** 1984). In fact, what we describe as ***project data model*** in figure ***6.1*** reflects our concern with only the ***state*** parts of the project work system and the project environment. This is because primary concern about the states of the project work system is in regard to their implications for the objectives of the project. This involves concentrating on the ***outcomes*** of processes within the project work system, rather than the processes themselves. Additionally, the project manager has the responsibility to keep himself informed about the current states of requirements and resources in the project environment but, again, he has no direct control over the processes that transform their states. Hence, for project management activities to be effective, a structured way of thinking about only ***states*** of these object systems needs to be available to the project manager, rather than about the ***processes*** by which these states have been achieved?.

---

† **The exception is when something goes wrong and the project manager needs to act in a technical capacity rather than a managerial capacity, to try to put it right. However, modelling such exceptions is outside the scope of this book, where we address the activities of the project manager acting in *role* (i.e., as a project manager), rather than investigate the technical processes in which the same person may be engaged, out of role, in an emergency.**

6.2.2. Anchoring project management model generation

The previous section described the general understanding of the role of the manager in relation to the various systems relevant to the modelling of the project management process and to the models we shall need to develop. This discussion provided the scope for the modelling enterprise. However, even within this specified scope, modelling of the project management process needs to be provided with some ***anchors*** to guide model generation according to **constraints** which explicitly relate to the real-world phenomena of interest.

The fundamental constraints identified in earlier chapters related to the goals the manager wishes to fulfil through his project management activities (shaping his motivation to carry them out), the project management methodology he has to use, the structure of the state of the project work system he is managing and the project environment Correspondingly, project management model generation is here anchored ***from above,*** through identifying and shaping the activities to be generated as components of the model, according to

(1) the **goal(s)** that the project manager may actualise through their execution, and,

(2) the external constraints imposed on him (defined either individually or as a set) by a project management methodology.

Project management model generation is also anchored ***from below*** by the structure of the project data model, which holds the entities on which the modelled management activities operate. In section 6.4, we will examine the project data model development process, indicating how this model may be pre-structured in terms of entity classes to serve the double purpose of

(a) constraining the project management model generation process from below, and,

(b) providing a structured view of the object systems to which its elements refer in reality, that is, the states of the project work system and of the requirements and resources in the project environment.

6.3. The development of the project management model

In chapter 5, we discussed how project management activities can be seen as ***means to achieve*** project management goals (the desired end of an activity). This ***means-end relationship*** between project management activities and goals aids model development as the concept of a project management goal can now be used to anchor each activity at the point of its activation (thus indicating what needs to be achieved through it, cf. Jungermann, von **Ulardt** & Hausmann 1983) and to test its execution at the point at which the activity has been completed (thus ascertaining whether it has been successfully carried out).

As we described in chapter 5, the particular project management activity which a manager may consider most appropriate to carry out at any particular time will depend upon the goal which the project manager wishes to **prioritise.**

Table 5.1 named some typical project management activities, indicating the goals which could trigger them, other goals which may be advanced through their execution, and goals the achievement of which needs to be safeguarded against potentially damaging side-effects of the activity. The purpose of table 5.1, in indexing the ***informal*** descriptions of typical project management activities given in chapters 4 and 5, was to set the agenda for the ***pre-formal*** modelling of each of these management activities in the way that will be described in chapter 7.

Project management activities themselves are carried out in the light of information about the objects on which they have to operate. The required information may have been the output of some previously carried out activity or gained directly from, or through inferences made about, states of objects within the project work system. Alternatively, it may reflect the states of requirements (instructions, guidelines) from the project environment or it may refer to states of resources (availability, characteristics) in the project environment. What information is needed by each project management activity, where it should be found and retrieved, and stored again when transformed through the operation of the activity, is thus an important issue in modelling the project management process.

In order to address this issue in a consistent way in pre-formal modelling of project management activities, a necessary first step is to precise the distinction between the active and passive elements contained in the informal scripts which describe the relevant activities (provided in earlier chapters). In section 6.3.1, we introduce the net modelling conventions that will be used to examine how this can be achieved. In section 6.3.2, we show how these passive and active elements can be consistently structured into pre-formal descriptions of Project Management ACtivities, or PMACs, expressed at the required degree of refinement†. Then, in section 6.3.3, we discuss what is involved in building instantiations of a project management model which have the capability for dynamic simulation of complex project management processes. Pre-formal descriptions of the relevant PMACs serve as initial building blocks whose specifications can be further precised in order to permit dynamic process simulation within the instantiation in which they are incorporated. In section 6.3.4, we describe the role of external constraints such as the implementation of a project management methodology in shaping and guiding the instantiation of a project management model.

† In this book, we use the term PMAC a refer to a pre-formally or formally modelled activity and preserve the term *project management activity* for an activity which is only informally described.

6.3.1. Net modelling techniques

The net modelling techniques we describe in this section allow one to ***generate*** the internal structure of any PMAC at whatever degree of refinement and precision is appropriate in pre-formal and formal project management process modelling and simulation. This generative approach to modelling has taken some ideas from earlier generative modelling schemes such as the **TOTE** systems described by Miller, Galanter and Pribram (1960) and Minsky's Frame-system theory (1977). However, these earlier accounts described pre-structured units, while, in our approach, the internal structures of PMACs can be generated ***during*** the operation of the project management system. Our approach can thus be considered ***evolutionary*** rather than static. This is in line with the reality of project management practice. A non-evolutionary project management model would only make sense in the case where an all-embracing and rigid project management methodology is faithfully and slavishly followed by the project manager throughout all his activities. This case, we have argued, does not hold for the majority of project managers.

The net modelling conventions we use are those appropriate for the specification of both approximate and exact predicate-transition nets (Genrich 1986) although, in this book, we give examples only of approximate net models (Reisig 1985). However, a major extension to the standard form of predicate-transition modelling is made here. That is that, while predicates may be tested as pre-conditions and generated as post-conditions in the usual way, in our version of the modelling technique, they can also be incorporated within an instantiation of the internal structure as it is developed within the PMAC. Hence, predicates can serve both as ***passive*** elements transferred between PMACs and as ***active*** elements within PMACs. This extension to the standard predicate-transition modelling technique is closely related to that made for ***coloured Petri nets*** (Jensen 1986) where predicates constitute ***expressions*** when they act as pre-conditions for an activity, and constitute ***functions*** when they ***are*** incorporated within the internal structure of the activity itself.

In addition, the ability of our modelling technique to express PMACs at any required level of refinement can be expressed in terms of hierarchies in coloured Petri nets (Huber, Jensen & Shapiro 1989). This is achieved through representing subPMACs as ***subpages*** within the page hierarchy describing the structure of the refinements instantiated at the various levels in the current coloured Petri net model.

According to the fundamental rule of net modelling (Petri 1982), a net is generated by connecting two types of elements: places and links. In the applications of net modelling we will describe here, ***places*** are interpreted as ***states*** of the project management model being generated, and ***links*** are interpreted as the project management activities which effect ***transitions*** between these states.

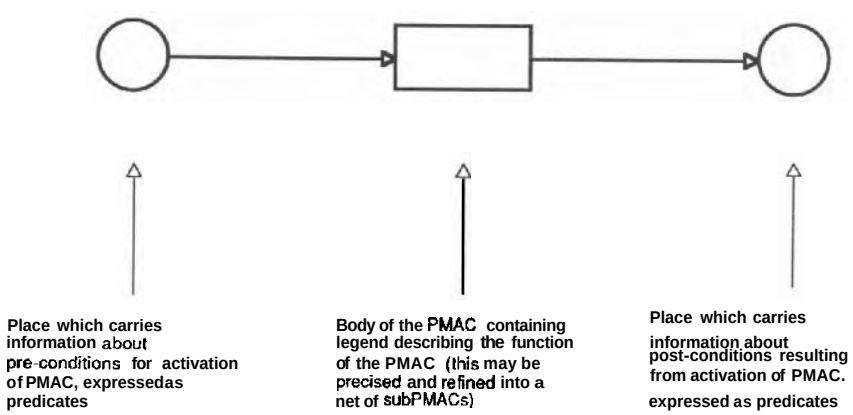
A place is depicted as a circular symbol. It may carry ***tokens*** which are individual objects with properties. The set of tokens on a place at any particular time is called its ***marking***. A place may have a ***legend*** (i.e., an informal description in natural language) as an integral part of it The legend given to a

place should describe the state achieved through its marking. More formally, an inscription on a place may be used to define the number or types of tokens (which may carry predicates) which can lie on it.

The places in a place-transition net are connected through PMACs. These links are shown as arrows where each arrow tail defines an exit point from a place and each arrow head defines an entry point to a place. A rectangular box, placed at the junction of arrowheads and arrowtails making up the link, may contain a legend naming the function of the PMAC. More formally, an inscription may be employed to define operations which are effected through activating the linkage.

In the interpretation of nets we use here to express PMACs and their relationships, the contents of a rectangular box describe the ***body*** of the PMAC. Tokens are used to carry the predicates which instantiate the ***pre-condition*** place linking into the body of the PMAC and the ***post-condition*** place linking outwards from the body of the PMAC, as shown in figure 6.2.

Figure 6.2: Linkage conventions in net modelling of PMACs



The simplest tokens, used in ***state-transition*** nets are merely distinguished by their presence or absence. For purposes of dynamic simulation in such a net, boolean states may be defined as the markings for a place (i.e., states defined in terms of predicates which evaluate to t if the place is marked by a token and ***nil*** otherwise).

In ***predicate-transition*** net modelling, as employed here, ***pre-conditions*** (activation conditions) for PMACs are tested as predicates on the properties of the tokens which appear on the places which are linked inwards to the PMAC. Activating the object may, in some net structures, lead to more than one (alternative) result. Here, predicates may be employed to determine which tokens, with what properties, may mark which of the places linked outwards from the PMAC, hence defining its ***post-conditions.***

More complex tokens may carry denotative information describing their enduring properties, or may carry information which reflects the history of the transitions in which the token has been involved in the net. A complex token of this type may be considered to be an instance of a class of tokens (predicates defining project management states) which has a particular set of attributes on which the information carried by each of its instances (i.e., tokens) is defined.

Like in board games, there may be several tokens with the same relevant properties (expressed as predicates) on places marked in terms of pre-conditions and post-conditions within the game (like pawns on a chessboard). Tokens can only leave places at exit points and enter at entry points during the course of a simulation. The only resmction for a link is that it must not have ***several*** exit or ***several*** entry points with the ***same*** place although it may have both an entry and an exit point with it.

During the course of a simulation, some of the places linked within the net will alter their markings as tokens appear on or disappear from them reflecting changes in project management model states within the net. A set of alterations of markings which occur together is called an ***atomic change*** because it is a specification of a change which is ***simple,*** that is, it cannot be decomposed within the net any further into component specifications of sub-changes.

The occurrence of an atomic change is subject to a ***rule of change***† which comprises two subrules:

- the ***enabling rule,*** prescribing the pre-requisites for the occurrence of an atomic change, and
- the ***occurrence rule,*** prescribing the effects of an occurrence on an enabled atomic change.

Nets may be used for both approximate and exact descriptions of the structural and dynamic properties of PMACs. The distinction between approximate and exact nets refers to different kinds of net models rather than to the level of detail addressed in the model. Exact net modelling requires nets with precisely structured inscriptions and the existence of a smct rule of change whereas ***approximate*** net modelling need not specify a strict rule of change and may use legends rather than inscriptions. A ***legend*** serves to explain the meaning of a place or link in terms of the modelled world. It may usefully be expressed in the natural language of the actual model builder. It is not

† Section 2.3 of Richter et *al* (1987) give a detailed discussion of the rule of change with some specific examples.

executable, and so, can not influence the actual behaviour of instantiations of the model which incorporate the place or link to which it refers. For this reason, an approximate net does not contain a proper specification of how it may be simulated.

However, an approximate net may be precised into an exact net through

(1) increasing the degree of structure in the legends on the places in the net until they form a complete and coherent set of inscriptions which effects the markings and atomic changes in the net. (This is achieved by precising their logical content into predicates which can be delivered as tokens, tested as pre-conditions and/or generated as post-conditions, as and when appropriate.)

(2) developing a strict rule of change from the process constraints described in the legends on the transitions in the net. (This is achieved by precising the rules of change which are implicit in these legends to specify the precise nature of the transformations effected in terms of tokens produced and consumed, and operations on entities of the project data model.)

6.3.2. Refinement of a PMAC

Figure 6.2 gave the coarsest possible view of the structure of a PMAC. This structure needs to be refined m give a more detailed PMAC specification template in order to allow pre-formal descriptions of individual instantiations of PMACs to be made. The necessary refinement, shown in figure 6.3, employs net modelling conventions to represent the basic structure of any PMAC as comprising three active entities:

(a) a pre-condition test of input predicates,

(b) a functional body consisting of a structure of subPMACs effecting operations on entities instantiated in the project data model, and,

(c) a post-condition test of goal achievement.

Three types of predicates are input to the pre-condition test:

(1) those in set P1 which result from previous PMAC use (that is, they have been produced as post-conditions by other previously instantiated PMACs),

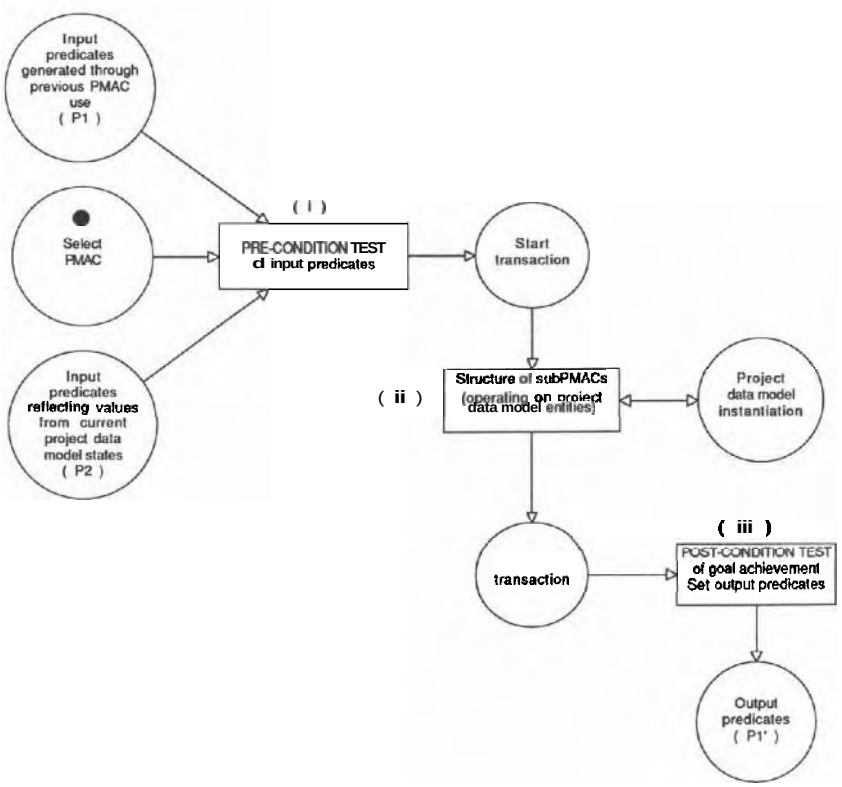
(2) those in set P2 which reflect the values of current states of entities modelled within the project data model†, and

(3) a single (boolean) input predicate which is marked (i.e., set *true)* when the manager selects the PMAC as something he would like to do now.

---

† Note that establishing the value of a typical predicate within this set. such as *actual* plan *is ready,* may require an examination of the current values attached to states of a variety of different entities represented within the project data model such as tasks having been defined, resources having being allocated to tasks, and so on.

Figure 63: The basic PMAC structure



The predicates which are output from the post-condition test of goal achievement refer exclusively to current project management model states; their only purpose is to serve as pre-conditions on ***subsequent*** PMAC activations.

In theory, of course, it would be possible to produce also output PMACs in a hypothetical set P2', reflecting current (post-PMAC operation) states of the project data model, but this would be very dangerous in practice. For example, the predicate ***actual plan is ready*** might be output by one PMAC (e.g., actual planning), and, sometime later on, be taken at its face value as input to the pre-condition test of another PMAC (e.g., identify/predict discrepancies). In the meantime though, the manager may have made some incomplete modifications to his planning with the result that his actual plan is now ***under revision*** rather than being *ready*.

The only really safe strategy here is to ***consume*** all the predicates in set P2 through the activation of the PMAC, and not to produce any. This means that the values of predicates in set P2 will always be computed on the basis of the ***current*** state values of the relevant project data model entities, rather than being pre-set on the basis of possibly out-of-date values which may no longer be accurate.

In chapter 7, we give pre-formal descriptions of the various PMACs named in table 5.1 of chapter 5. For each, we describe, in general terms, the operations on the project data model carried out within it. We also name some representative input and output predicates which serve as pre-conditions and post-conditions, respectively, and identify the goals addressed in post-condition tests within the PMAC. Then, in chapter 8, we show how the body of the PMAC indicated in figure 6.3 may be refined in terms of a structure of sub-PMACs when generating specific instances of PMACs. However, the basic (coarse) PMAC structure shown in figure 6.3 remains the same, regardless of the level of refinement at which a PMAC is represented.

6.3.3. Building instantiations of a project management model capable of dynamic simulation

Pre-formal descriptions of PMACs can serve as "building blocks" which may be linked together according to net modelling conventions to form project management model instantiations. These instantiations, when represented at a sufficient level of precision, permit the simulation of project management activities according to how and when they could actually be carried out in the project management system.

In developing project management model instantiations, it is usual to start the modelling enterprise at the level of approximate nets as (i) their legends are likely to be closer to the natural language of the person supplying the relevant knowledge concerning the functioning of the project management system (i.e., the project manager), and, (ii) they provide a clear and compact view of the static structure of the system components and their linkages. This is the approach which we adopt in chapter 8.

In developing instantiations where the capability for dynamic simulation is required (as when, for example, the net will form a specification for the automated part of a project management training simulation), the approximate model may then be precised into an exact net at the appropriate level of refinement?. This final stage in model development is difficult to achieve solely on the basis of paper-based work, but is made much easier through the use of interactive graphical tools for coloured Petri net model development and simulation. Tools like Design/CPN (Albrecht, Jensen & Shapiro 1989) already offer interactive aids to net refinement, consistency checking and maintenance of a subpage hierarchy. Facilities like these allow the model developer to test the dynamic process simulation properties of instantiations of the project management model as required during this stage of the model development process.

6.3.4. The role of external constraints in instantiating the model

In theory, offering the project manager access to a full project management model, comprising both processes and states, would provide him with the possibility of generating instances of this model in exactly the form required for managing his particular project (i.e., through identifying project management activities at the desired level of refinement, specifying how they are to operate and linking their activation through the pre- and post-conditions that have to be met in practice). Taking advantage of this possibility would involve the manager playing the role of the ***management expert.*** The resulting instantiation would then be available to guide and constrain him in his activities in ***managing*** the particular project in the manner indicated in figure 6.1. In practice, the ***management expert*** role is often performed by senior management within the organisation, or by quality assurance personnel under their direction.

In the absence of comprehensive interactive support for project management modelling of the type that will be described in the later chapters of this book, the ***management expert*** role may not be fully exploited in practice. Not much of the structure of the project management model may end up being generated in the integrated way which is appropriate for the management of a particular project. Instead, standards and pre-defined methodology to be followed in managing the project tend to be prescribed piecemeal in the project environment with the aim of regulating how and when particular management activities should be carried out in operating on particular objects in the project work system. The actual process of project management then becomes a matter of intuition (seat-of-the-pants expertise) on behalf of the project manager, partially

† In practice, exact net modelling is often done with formal inscriptions, and informal text is used to provide legends in approximate net modelling. However, this does not always have to be the case. For a net to be an exact or an appmximate model, it does not depend on whether the language used is formal or informal. Thus, terms like "formal nets" or "pre-formal nets" are not quite appropriate to distinguish exact net models fmm approximate net models.

constrained by the prescribed management standards and methodology.

Major problems can arise when the prescribed methodology is expressed in terms of a fixed structure, set in an organisational handbook, as discussed in section 4.2.1, above. The project manager may then be only allowed to tailor its prescriptions, rather than being encouraged to organise the project management process in such a way as to meet the goals and standards of the prescribed methodology. The reason usually given is that the latter activity, based solely on the manager's intuition, would defeat the methodology's intended purpose, that is, impose a *fixed* view of what constitutes good project management practice. The result is often that the manager is needlessly over-constmined on some of his activities, making contingency planning and creative management solutions difficult, while being left entirely to his own devices in regard to other activities where some (non-prestructured) methodological guidance would have been helpful.

Within our approach to modelling the project management process we have described so far, externally prescribed management methodologies may be viewed as a ***set of partial constraints*** on project management model ***instantiation,*** serving to regulate the conditions under which particular management activities may be executed (according to the instantiation), shaping the project management process so that it conforms to the specific, current requirements for "good management practice". Thus, some of the pre-conditions for particular management activities may be used to reflect constraints imposed by senior management while others may be fixed ***a prwn*** by the manager himself, acting in a ***management expert*** role. Conversely, the ***interpretation*** of the prescribed methodology in actual project management practice is achieved through the way the instantiated pre-conditions shape the project management process throughout the execution of the project.

6.4. Development of the project data model

In section 6.1.3, we contrasted the generative approach taken in project management model development with the generic approach to be employed in project data model development. In the final part of this chapter, we describe how we follow this approach, starting in section 6.4.1, where we define the scope of the project data model in terms of what can be "seen" through the perspectives that we identified in chapter 5 to identify the entities that will comprise the generic core of the project data model. Then, in section 6.4.2, we show how entity-relationship modelling conventions (Chen 1976, 1980) can be used to develop *class* specifications for the entities represented in the project data model. The resulting specifications for all the entities in the generic core of the project data model are presented in chapter 10†. We also indicate how

† There are many published variants of entity-relationship modelling conventions, and no single standard for *them.* The conventions we will use are designed to maintain consistency with the conventions used in net modelling.

an instantiation of any part of the project data model can be developed through creating instances of entity classes, assigning values to their intrinsic properties and relating to instances created in other entity classes. In chapter 10, we will provide a detailed example of how the instantiation process is actually controlled through the operations defined within a specific PMAC.

In this book, we give detailed specifications only for entities within the generic core of the project data model, thus presenting in a pre-structured way the part of the model which could reasonably be employed by a modeller, a project manager, or even a project management support system, across a wide range of projects, given the way they are managed in a wide variety of organisations. However, in the context of managing any particular project in a particular organisation, the generic project data model is likely to need to be ***tailored,*** that is, additional context-specific entities would need to be added to the generic core and the specifications of some of the core entities would need to be revised to meet the precise information needs of the project manager and to match the language he wishes to employ in describing project data model entities.

We do not discuss the generic model tailoring process in any detail here, but merely remark that it should involve the use of the same entity-relationship modelling concepts as those described and illustrated through the examples employing core project data model entities which we give in section 6.4.2 and chapter 10 of this book. It must be born in mind, however, that the resulting, tailored, project data model should be considered to be context-specific and, thus, no longer generic. Moreover, great care must be taken to ensure that the tailoring process does not introduce inconsistencies between specifications of the project data model entities defined within PMAC operations and the specifications of the entities with the same names in the tailored project data model on which the PMACs now operate. Project data model entities like Task may be the subject of operations defined within many different PMACs, and tailoring their specification will necessitate a revision of the relevant operations in *all* of the many PMACs involved.

6.4.1. Defining the scope of the project data model

In section 5.2, we described four ***reference perspectives*** commonly adopted by project managers in describing objects of interest in the project work system and the project environment. These four perspectives were identified as:

(1) the ***project function perspective,*** which relates ***to*** the function of the project as a whole;

(2) the ***activity perspective,*** which relates to the work which needs to be carried out within the project to achieve its aim;

(3) the ***time*** and ***resource perspective,*** which relates to resource aspects of the project in terms of time and in terms of the physical resources which are instrumental to achieving the aim of the project; and,

(4) the ***personnel perspective,*** which relates to the concerns of human resources, viewed as human beings working in various roles.

Each perspective helps in identifying and providing a structured view on the objects which need to be represented by the specification of entities in the project data model, as they will be the subject of the manager's activities and hence of operations modelled within PMACs.

We do not propose, however, to build separate parts of the project data model, each comprising those entities which can be seen in a particular perspective. Just as different perspectives result from adopting different viewpoints on the same real world of the project work system and the project environment, so may perspectives offer ***virtual views*** on the set of entities, and the relationships between them, which constitute the project data model. Not all the entities in the project data model will be visible in each of the perspectives adopted by the project manager, but many of the entities will be visible within more than one perspective, though different attributes of an entity will be revealed in different perspectives.

Thus, in scoping the project data model, we will be interested in characterising each of its constituent entities in terms of ***all*** the attributes (properties and relationships) which it may be seen to possess as a result of adopting ***all*** the perspectives taken by the project manager.

The resulting project data model will, of course, be more complex than would be an equivalent model of what could be seen through taking any single perspective on the project work system and the project environment. This, in itself, creates a problem for the project manager when reviewing the contents of the project data model: human beings cannot apprehend highly complex representations in their totality due to their inherently limited capacity for processing information (Miller 1956). If they are to gain a full understanding of complex representations like those built in project data modelling, they need to be provided with a coherent and consistent set of structured partial, and, hence, less complex, views on any such representation (Larichev, Moshkovich & Rebrik 1988). Moreover, each of these views needs to be structured in a way that is intuitively understandable by the viewer (i.e., the project manager), and to employ language and representation concepts which are familiar to him through his transactions with the real world objects represented by the entities viewed (Larichev 1984).

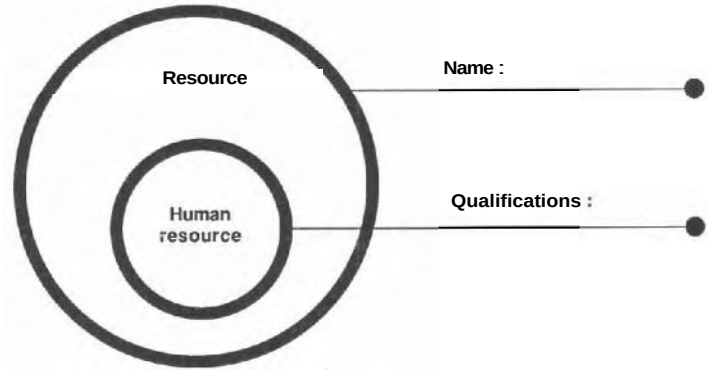
Hence, it makes sense to develop means of viewing complex representations of project data model entities ***in perspective.*** In later chapters, we will examine how this might be achieved in practice. Here, though, we should remark that the complex entities viewed in perspective will be ***virtual entities,*** each of which is synthesised as a collage of partial descriptions (i.e., involving some, but not all, of the atuibutes of one or a number of different entities which are specified in the project data model). Any such ***virtual entity*** will not, in itself, have a denoted, specific identity within the project data model. Instead, it is ***temporarily*** synthesised for the sole purpose of being viewed within the perspective on the project data model which is currently adopted by the project manager.

6.4.2. Entity-relationship modelling of project data

For any entity modelled in a pre-formal way within the project data model, the entity-relationship modelling conventions we employ are as follows.

An entity class is represented by a circle drawn with a heavy outline, with the class name (given to each instance within the class) inscribed inside the circle. Nested circles represent specialisations of the entity class into subclasses. The inscriptions within the nested circles then describe the nature of each specialisation. As an example, figure 6.4 shows the project data model class entity **Resource** with its specialisation **Human resource** nested within it.

Figure 6.4: Nesting of entity classes



An entity class may have any number of class attributes attached to it Each attribute should characterise, to varying degrees and with various values, all the instances of the project data model entity with the given class name.

Attributes are divided into two types: ***property*** atmbutes, which describe intrinsic characteristics of the entity class, and ***relationship*** atmbutes, which define the way in which instances in the entity class may be related to particular instances of another entity class within the project data model.

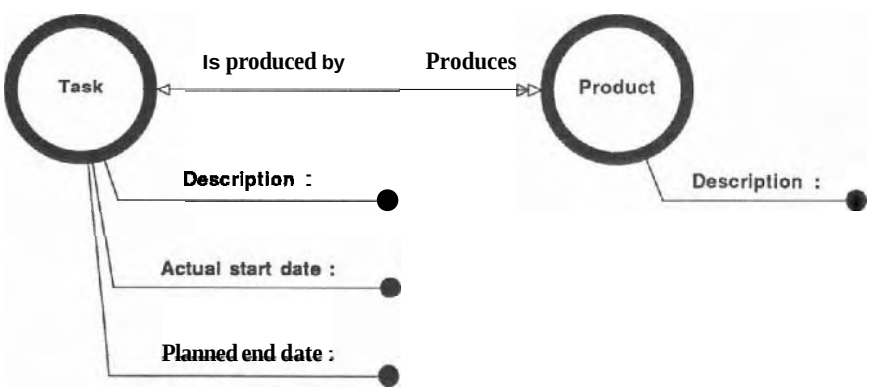
Attributes are represented by segments attached to the circle defining the class to which they apply. In the case of a property atmbute, the segment is not oriented towards any other class and so is shown as terminating at the class on which it is defined. For example, in figure 6.4, ***Name*** is an attribute of the **Resource** entity class: it can take on different values for different instances

within the class, but each value will identify the (specific) ***name*** by which that Resource instance is called.

The attribute ***qualifications*** is shown in figure 6.4 as a property of Human Resource, but not of Resource in general. This implies that human resources may have their own individual names and qualifications represented within the project data model, but that other (non-human) resources represented within the project data model may have their own individual names, but not qualifications, represented (at least, not within the ***generic*** project data model, as not many organisations and institutions currently are in the habit of conferring qualifications on machines, etc)†.

Figure 6.5 gives an example of two entity classes liked through oriented segments. There, the entity class named Task is shown as having the property attributes ***description, actual start date*** and ***planned end dare.*** It is also shown as having the relationship attribute ***produces,*** indicated by a segment oriented towards the entity class named Product. The ***cardinality*** of the relationship between particular instances in a pair of classes may be one-to-one, one-to-many or many-to-many. In figure 6.5, the double arrow on the relationship attribute oriented towards the Product entity class indicates that many Product instances may be ***produced*** by a single Task instance.

Figure 6.5: Partial definition of the Task and Product entity classes

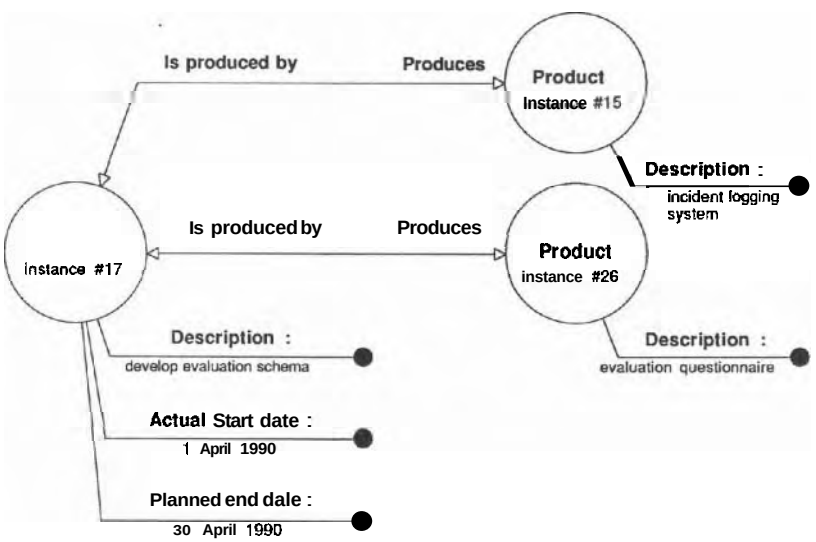


† Note, however, that all the examples given in this chapter show only a subset of the attributes which are defined for each entity class in the generic project data model. A more extensive list of the property attributes that may be defined for each entity class is given in the specification of that entity class in chapter 10.

The reciprocal relationship which may exist between **Product** and **Task** instances is also indicated in figure 6.5: the entity class named **Product** is shown as having the relationship attribute ***is produced*** by oriented towards the entity class named **Task.** However, the single arrowhead on the segment oriented in this direction indicates that any **Product** instance ***is produced*** by a single task, although which **Task** instance produces which **Product** instance (and vice versa) is not defined until the particular instances are created and linked.

Instantiations of entities are shown in the same formalism as the entity classes. We distinguish here the representation of particular instances from that of the entity class to which they belong by drawing the circles which define instances with a light outline, rather than a heavy one, as illustrated in figure 6.6.

Figure 6.6: Example of an instantiation of Task and Product entities



In the example shown in figure 6.6, a particular instance of the **Task** entity class has been created. It has been given the identifier #17, but this identifier has no intrinsic meaning: its purpose is simply to supply the instance with a unique means of identification which will distinguish it from all the other instances which may be created within the same entity class.

The pmperty attribute ***description*** has been given the value "develop evaluation schema"; ***actual start date*** has been given the value "1 April 1990 and ***planned end dare*** has been given the value "30 April 1990". In fact, it is not mandatory to give each attribute a value at the moment an instance is

created; attributes of instances can exist with their values "undefined". Existing values assigned to an instance's property attributes can be changed without having to create a new instance of the entity.

In figure 6.6, the relationship attribute ***produces*** has been assigned *two* values: it points towards Product instance #15 ***and*** towards Product instance #26. This is permitted here, as the cardinality of the Task ***produces*** Product relationship was defined as one-to many?.

To find the meaning of Product #15 and Product #26 (i.e., to find out ***what*** may be produced), we can examine the values currently assigned to these instances of the Product entity class. This examination reveals that the ***description*** property of Product instance #15 has been assigned the value "incident logging system", and that the ***description*** propeny of Product instance #26 has been assigned the value "evaluation questionnaire". Thus, the example instantiation shown in figure 6.6 has modelled the fact that, in the view of the person who made the instantiation, that is, the project manager,

> the task that is described as "develop evaluation schema" produces the product described as "incident logging system". It also produces the product described as "evaluation questionnaire". ,nr VS 14

Similarly, this instantiation has modelled the reciprocal fact that, in the view of the project manager,

> the product described as "incident logging system" and the product described as "evaluation questionnaire" are both produced by the task described as "develop evaluation schema".

It is important to remember that, in building the entity class definitions shown in figure 6.5 and in making the instantiation shown in figure 6.6, the ***process*** by which Task instance #17 ***produced*** Product instances #15 and #26 was not and, indeed, could not be modelled in following the entity-relationship approach in building the project data model. The presented examples organised information about ***what*** may be produced by the task described as "develop evaluation schema", but not about ***how*** this may be achieved.

This limitation is a general feature of entity-relationship modelling where attributes indicate the relationships which ***may*** be effected between instances of entity classes but there is no means of providing dynamic simulation capabilities which would reveal how and when a particular process, if activated, would actually effect the transformation or transition indicated in a particular relationship between particular instances. This is in contrast with exact net modelling which, as we described in section 6.3.1, does offer this capability (Reisig 1986).

However, as we discussed earlier, this limitation of entity-relationship modelling does not matter to the project manager as he is interested only in information about ***stares*** of the project work system and the project

† In order to maintain coherence in the instantiation of the project data model, these two instances of the Product entity class would have to be created at the time they were linked to instance #17 of the Task entity class, if they did not exist already.

environment, which are interpreted here in terms of values assigned to attributes of instances of entity classes in the project data model. The processes which actually effect the transitions between those states are not his responsibility, and thus lie outside the concern of our project data modelling enterprise.

Chapter 7

Pre-formal description of project management activities

As described in the previous chapters, a project management activity is constrained both by the goals the manager wants to achieve through it and by what needs to be done by him at the current stage the project is in. Thus, a project management activity includes a ***stable*** referent to its understanding (i.e., a specific goal which does not vary in its aim) and a ***variable*** referent (i.e., the phase of the project) which affects what needs to be taken into account in carrying out the activity, what information needs to be collected by the project manager before he can carry it out, and such like. For example, planning before the project has started is based on a rough knowledge about the software system to be developed, and the outcome of the manager's planning is itself coarse and general. On the contrary, re-planning necessitates a great deal of information on the project's progress up to that point so that the revised plan will handle the problems encountered with the plan it replaces.

Table 7.1 relates the project management activities we discussed in chapters 4 and 5 to the project management phases during which they are most ***typically*** carried out Like table 5.1, which related the same list of project management activities to the project management goals that they may achieve or subvert in the process of their execution, table 7.1 is ***descriptive,*** rather than ***prescriptive.*** The indication in table 7.1 that a PMAC may be activated in a particular phase should not be taken to imply that it *has* to be activated in that phase. It may be advisable to do so, in general, but a particular project or a particular management methodology used may not require it.

However, in developing pre-formal descriptions of project management activities, the notion of project management phases, on its own, is insufficient for forming a basis for such descriptions. Taking a phase-approach could indicate only which PMACs should be carried out in a particular phase ***before that phase is completed.*** It would not indicate whether the preconditions are appropriate for activating a PMAC (i.e., depending on the full complexity of the current state of the project, not just which phase it is in), and would not

take any regard of the post-conditions of particular activities which contribute to indicating which PMACs may be carried out next. As such, a purely phase-driven approach might, in theory, meet *external* requirements (i.e., those from the manager's superiors, the client, and the project team) in terms of what may be expected to be produced by the end of a particular phase. It would, however, take no account of the fundamental management requirement *internal* to the project, which is to choose the most appropriate PMAC, given the current managerial goal and state of the project.

Table 7.1: Summary of the relationships between phases and PMACs

Activities	Phases				
	1	2	3	4	5
Actual planning		x		x	
Analyse risks	x	x		x	
Anticipate future use of project				x	x
Confirm reqs and deliverables		x		x	
Debriefing project team					x
Design working environment		x	x	x	
Develop case studies				x	x
Diagnose & remedy exceptions				x	
Handle team problems				x	
Handover results				x	x
Identify & predict discrepancies				x	
Negotiate changes	x	x		x	
Negotiate resources	x	x		x	
Negotiate working conditions	x			x	
Project archiving					x
Project initialisation	x				
Publicise project			x	x	x
Quality assurance planning		x			
Report on progress				x	
Set up change control procedures		x			
Set up liaison	x		x		
Set up monitoring procedures			x		
Specify requirements	x	x		x	
Standard planning	x	x		x	
Tailor management process	x				
Team building			x	x	
Trigger plan			x		

Key to *table:*

Phase 1 **Taking over the project**
Phase 2 **Initial planning**
Phase 3 **Launching the project**
Phase 4 **Running** the **project**
Phase 5 **Closing the project**

We have argued in previous chapters that taking a purely external view on the manager's activities and their results excludes the possibility of developing adequate criteria for testing the success of either anticipated or actualised management activities. For example, the mere production of a project plan, indicating the completion of the initial planning phase of the project, is, on its own, a misleading criterion for assessing the success of a planning activity. Instead, it is also necessary to determine that the ***contents*** of ***the plan*** satisfy all those directly or indirectly involved in the project, in other words, that the planning activity has achieved the management goal set for it Hence, it is important to test for goal-achievement, not merely phase-achievement, in evaluating the success of any management activity.

In developing an instantiation of the project management model from scratch, one would start from a particular goal and a particular phase-context and generate the PMAC which best reflects the desires and capabilities of the manager to act effectively on the project work system, given this goal and phase-context. However, actual software development project management practices are not entirely idiosyncratic. That is, the constraints set by goals and phases are fairly stable in coarse terms, and the PMACs developed to act as means to particular ends often contain similar patterns of operations (at least, when viewed at a coarse level of refinement).

In practice, project managers use a mixture of PMAC generation techniques and PMAC selection and tailoring techniques. In the latter case, the PMACs selected would have been modelled ***a priori*** (at least, in part) in some kind of pre-formal way. For instance, following a standard management methodology may involve the selection of a PMAC which is partially pre-structured to represent the constraints and prescriptions imposed by the methodology. The project manager's task would then be to ***tailor*** the pre-structured parts or generate additional parts to meet current circumstances without infringing the predefined methodological constraints.

This strategy is quite common in the case of PMACs dealing with, for example, standard planning or setting up monitoring procedures, but is less in evidence in the case of PMACs where greater flexibility is required. This is the case with PMACs entailing a high degree of uncertainty about the nature and sequencing of consistent operations which will enable their successful completion and, correspondingly, ***a priori*** rather than on a contingency basis, the low degree of control that a manager can have over their results. This kind of flexibility is usually required in PMACs which are carried out at the ***boundary*** of the project management system with other systems, involving ***transactions*** between the manager and other systems (other managers in the contractor organisation, the client organisation, project team members, and so on). Typical examples of ***boundary-spanning*** PMACs are all those which involve negotiation (e.g., negotiating changes or resources) or unstructured interactions (e.g., handling team problems).

Other PMACs, however, can be considered as purely ***internal*** in the sense that, although their execution may need information from other systems (e.g., on progress of work), carrying out the activity does not necessitate any

transaction with other systems once this information is available to the manager. Typical examples of such PMACs are those which focus on project planning and reporting. It is true that, in PMACs of this kind, the manager will also need to have a view of other systems which will be affected by his activities or which will be the recipients of their results in order to ensure that these results will be satisfactory to them. However, the responsibility for their outcome (and, hence, possibility for executive control) lies entirely with the project manager.

In general, internal activities present the manager with a lower degree of uncertainty about how to control their outcomes than do boundary-spanning activities (White 1986). In the execution of boundary-spanning activities, the manager has control only over his own behaviour and may have precious little knowledge of the potential behaviour of the other system with which he has to interact during the execution of the activity. Uncertainty about the intentions, actions and responses of other people with whom he must transact is likely to be high, and this may constitute a serious threat to the effective management of the project. This is particularly true of boundary-spanning activities involving transactions with the client or other managers in his own organisation.

In contrast, the project manager can exercise his executive power over the project work system to control the outcome of his transactions with this system. This can reduce the uncertainty involved in those boundary-spanning activities where the boundary is located between the project management system and the project work system. However, the way in which he chooses to exercise this control can foster or deter progress in the project in the way we have discussed in sections **4.2.4.1** and **5.1.2.4.** For example, an authoritarian style of management reduces this kind of uncertainty virtually to zero but it also is likely to engender a great deal of dissatisfaction and resentment within the project team This, in turn, may lead to grave problems for the project.

In summary, as the success or failure of boundary-spanning activities is predicated on the success or failure of managerial transactions with other systems over which the manager has no control, the refined, local, structure of PMACs modelling these activities is much more fluid than for internal activities: much depends upon ***ad hoc*** refinements to generate local procedures to respond to contingencies. In such cases, the internal structure of the relevant PMACs (detailing which operations will be carried out on what project data model entities under what conditions) is usually generated afresh on an ***ad hoc*** basis to handle each situation as it arises.

Hence, for boundary-spanning PMACs, the pre-formal descriptions given in section 7.1 should only be taken as a guide for what a typical PMAC of this type might involve. In chapter 8, we do not discuss typical refinements for any of these types of PMACs, as, for them, there is no such thing as a typical refinement. In chapters 8 and 9, we discuss the further refinement of the ***standard planning, analyse risks, actual planning, set up monitoring procedures,*** and ***identify and predict discrepancies and exceptions*** PMACs. The intemal structure of each of these PMACs is more likely, in practical applications, to be selected and tailored (at least at the fairly coarse level described in chapter 8)

rather than generated from scratch. So, it is more reasonable to describe these PMACs in terms of typical refinements.

7.1. Descriptions of typical PMACs

In developing pre-formal descriptions of typical project management activities (i.e., PMACs), we took into account our informal descriptions of management activities in chapters 4 and 5, together with the conventions for PMAC modelling described in chapter 6, and the distinction between boundary-spanning and internal project management activities described in the previous section.

The pre-formal description of each of the typical PMACs which will be described in this section is organised according to:

- its ***function,*** that is, what it purports to achieve;
- its ***type*** , defining whether it is a boundary-spanning or internal activity;
- the ***typical input predicates*** to the pre-condition test. Those input predicates which are marked ® will usually ***be required to have the value shown*** for the pre-condition test to pass and the PMAC to be activated.
- the ***typical output predicates*** which are produced as a result of its activation. Not all the results (i.e., changes in instances of the project data model) will necessarily be reflected in output predicates. The main function of output predicates is to hold information about the results achieved through the activation of this PMAC which may subsequently serve as useful input to other PMACs.
- the principal ***operations*** carried out within it (i.e., the process by which it is carried out, thereby operating on entities in the project data model and transforming input predicates into output predicates);
- the ***perspectives*** which may be employed to view the entities on which the PMAC operates within the project data model; and,
- the ***goals*** addressed in the post-condition test which indicates whether the purpose of the activity has been met or not (the first goal listed being the triggering goal).

In the descriptions of the PMACs that follow, we have identified typical, rather than mandatory, input predicates for each PMAC. In any particular modelling application, ***which*** of these predicates will be specified for evaluation within the PMAC's pre-condition test will depend on

(1) the precise nature of the operations specified in refining the PMAC's internal structure, and,

(2) the constraints imposed by any prescribed project management methodology which seeks to achieve "good project management" by ensuring that the particular PMAC is only carried out when the pre-conditions exist for its activation to yield successful results, seen in the wider context of sequences of PMAC use in the overall process of managing the project.

The nature and sources of these predicates will be examined in more detail in section 7.2.

It is important to bear in mind that the PMACs listed below do not constitute a fully comprehensive list of ***all*** the potential activities a project manager may need to carry out to fulfil his responsibilities. They simply represent a reasonably good sample of PMACs which, taken together, address much of the range and complexity of the work of a project manager. As such, they are designed to provide a starting point for the modelling of particular management activities in a more refined way. In chapter 8, we will show how such refinement may be achieved for some of the PMACs which are described pre-formally here. The pre-formal descriptions for each of the PMACs identified in tables 5.1 and 7.1 are listed below alphabetically.

Actual planning ***(detailed in section 8.3)***

Function: to plan on the basis of knowledge about actually available resources to the project at any particular time.

Type: internal.

Typical input predicates: standard plan is available@, resources committed are known®; calendar is known@, risk advice is available; cost, duration and effort estimation models to be used are defined, resource skill profile is available.

Typical output predicate: actual plan is ready.

Principal operations carried out within PMAC: coordinate tasks with calendar, control allocation of defined, individual resources across time; test for adequacy of the actual plan.

Perspectives employed: activity; time and resource; personnel.

Goals addressed in post-condition tests: create and implement a project plan; satisfy external stakeholders; maintain relationship between plan and reality; facilitate project work; advance technical expertise.

Analyse risks ***(detailed in section 8.2)***

Function: to analyse the risks of the project

Type: internal.

Typical input predicates: project has been initialised®; product requirements are available@, risk model is available@; coarse standard plan is available; feasibility study results are available; information on previously archived projects and case studies are available.

Typical output predicate: risk advice is available (i.e., project risk profile is available together with its implications for management).

Principal operations carried out within PMAC: organise information on project risks into project risk profile; coordinate risk profile patterns with risk management rules; test adequacy of risk analysis; coordinate information needs with information available from organisational sources.

Perspective employed: project function.

Goal addressed in post-condition tests: clarify extemal requirements; satisfy external stakeholders; facilitate project work.

Anticipate future use of project

Function: to decide on what parts of the experience of the current project need to be used for future projects.

Type: internal.

Typical input predicates: management reports are available@; lists of confirmations, exceptions, and remedies are available; lists of cost, effort and duration estimates are available; information on the project and its environment is available.

Typical output predicates: material from the project to be archived is identified, future uses of material from the project are defined.

Principal operations carried out within PMAC: organise confirmations and exceptions; coordinate exceptions with remedies; coordinate estimates with monitoring results.

Perspective employed: project function.

Goals addressed in post-condition tests: advance technical expertise; advance project management expertise.

Confirm requirements and deliverables

Function: to gain acceptance of the specification of the requirements and of the timing of extemal deliverables as developed during the use of standard planning and actual planning PMACs.

Type: boundary-spanning.

Typical input predicates: product requirements are available@; list of deliverables is available®; standard plan is available; project risk advice is available; organisational priorities, constraints and objectives are known; product design information is available.

Typical output predicates: requirements and deliverables are confirmed; list of external stakeholders to communicate with is available; acceptance criteria are defined, information related to the requirements of the stakeholders is organised.

Principal operations carried our within PMAC: coordinate product requirements with project deliverables; organise reporting to stakeholders.

Perspective employed: project function.

Goals addressed in post-condition tests: clarify external requirements; satisfy external stakeholders; create and implement a project plan; maintain relationship between plan and reality; facilitate project work.

Debriefing of project team

Function: to gain information about team members' experience from the project

Type: boundary-spanning.

Typical input predicates: actual plan is available; reports from team members are available; lists of confirmations, exceptions and remedial actions are available.

Typical output predicate: information about team members' performance is related to particular situations in the project for future use.

Principal operations carried out within PMAC: organise reports and scenarios provided by team members, coordinate to specific instances.

Perspectives employed: project function; personnel.

Goals addressed in post-condition tests: advance technical expertise; advance project management expertise; satisfy external stakeholders.

Design working environment

Function: to enable the project team make best use of working conditions when working on the tasks described in the project plan.

Type: internal.

Typical input predicates: standard plan is available®; resource skill profile is available; product requirements are available; personnel requirements for working conditions are known; project risk advice is available; resources' availabiiities to the project are known.

Typical output predicates: working conditions are established.

Principal operation carried out within PMAC: coordinate task requirements with resource skills/preferences.

Perspective employed: personnel.

Goals addressed in post-condition tests: facilitate project work; satisfy external stakeholders.

Develop case studies

Function: to create case studies from experience gained in the current project for purposes of future organisational use and learning.

Type: internal.

Typical input predicates: standard plan is available®; actual plan is available@; project risk advice is available®; lists of confirmations, exceptions and remedial actions are available®; material from the project to be archived is identified; working conditions are established; information on the project and its environment is available; reports from team(s) are available.

Typical output predicate: case studies are ready.

Principal operations carried out within PMAC: organise project progress information into episodes; coordinate episodes into stories.

Perspectives employed: project function.

Goals addressed in post-condition tests: advance project management expertise; advance technical expertise.

Diagnose and remedy exceptions

Function: to diagnose exceptions (i.e., severe discrepancies) and choose a course of remedial action.

Type: internal.

Typical input predicates: list of exceptions is available®; actual plan is available@; product requirements and project risk advice are available.

Typical output predicate: potential remedial actions are defined.

Principal operation carried out within PMAC: coordinate diagnoses with remedies.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condition tests: maintain relationship between plan and reality; facilitate project work; clarify external requirements; satisfy external stakeholders; create and implement a project plan.

Handle team problems

Function: to handle organisational problems of the team whether occurring within the team or between the team and external stakeholders.

Type: boundary-spanning.

Typical input predicates: complaints and problems are known (information from team members)@, lists of confirmations, exceptions and remedial actions are available; management and technical reports are available; working conditions are established.

Typical output predicates: list of remedies to the problems is ready; working conditions are changed.

Principal operations carried out within PMAC: organise reports; test remedies.

Perspective employed: personnel.

Goals addressed in post-condition tests: facilitate project work, maintain relationship between plan and reality; create and implement a project plan; satisfy external stakeholders.

Handover results

Function: to deliver the products of the project to the project environment.

Type: boundary-spanning.

Typical input predicates: product to *be* handed over is completed@, corresponding project and product documentation is available (as indicated through list of confirmations).

Typical output predicates: product is handed over, product documentation is delivered.

Principal operations carried out within PMAC: organise results of project; coordinate produced deliverables with planned deliverables.

Perspective employed: project function.

Goal addressed in post-condition tests: satisfy external stakeholders.

Identify/predict discrepancies and exceptions ***(detailed in section 8.5)***

Function: to identify and predict deviations from the plan given the results of monitoring and assess the severity of discrepancies.

Type: internal.

Typical input predicates: actual plan is activated®; entities to be observed and reported on are identified@; estimates of effort, cost and duration are available®; risk advice is available; monitoring procedures are established.

Typical output predicates: missing reports are identified; lists of confirmations and exceptions are available.

Principal operations carried out within PMAC: control timed reporting; coordinate planned and reported values for relevant entities in the project data model; control forward tracking as effects of identified discrepancies.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condition tests: maintain relationship between plan and reality; facilitate project work; satisfy external stakeholders; create and implement a project plan.

Negotiate changes

Function: to negotiate changes in the project with external stakeholders brought about through simulating or implementing a project plan or by the need to clarify the requirements to the project.

Type: boundary-spanning.

Typical input predicates: proposed changes and reasons for proposed changes are available (e.g., lists of exceptions, remedies)@; product change control procedures are available@; organisational objectives are known; product requirements are available; list of deliverables is available; product design information is available.

Typical output predicates: product requirements are reviewed; product design is changed as agreed; revised list of deliverables (if any) is available.

Principal operations carried out within PMAC: organise issues to be negotiated; coordinate product requirements with project deliverables.

Perspective employed: project function.

Goals addressed in post-condition tests: maintain the relationship between plan and reality; clarify external requirements; satisfy external stakeholders; create and implement a project plan; facilitate project work.

Negotiate resources

Function: to negotiate the amount and type of resources assigned to the project by the contractor organisation and the client.

Type: boundary-spanning.

Typical input predicates: (coarse) standard plan is available (with estimates of required resources)®; required resource skill profile is known®; organisational objectives are known; product design is ready; information on resource pool is available.

Typical output predicate: actual resources available to the project are known.

P ncipal operations carried out within PMAC: organise product requirements; organise reports; coordinate reports, requirements and resources offered.

Perspectives employed: activity; time and resource; personnel.

Goals addressed in post-condition tests: maintain relationship between plan and reality; satisfy external stakeholders; create and implement a project plan; facilitate project work.

Negotiate working conditions

Function: to negotiate working conditions for project team with the client and the contractor organisation.

Type: boundary-spanning.

Typical input predicates: actual resources (potentially) available to the project are known®; working conditions are established@; information on the project and the project environment is available.

Typical output predicate: working conditions are agreed.

Principal operations carried out within PMAC: control project requirements with available resources; control personnel requirements with task requirements.

Perspectives employed: project function; personnel.
ri

Goals addressed in post-condition rests: facilitate project work; satisfy external stakeholders; create and implement a project plan.

Project archiving

Function: to archive project information for future use.

Type: internal.

Typical input predicates: management reports are available®; material from the project to be archived is identified®; product and project documentation (referenced through list of confirmations) is available; future uses of material from the project are defined; information on the project and the project environment is available; project risk advice is available; case studies are available.

Typical output predicate: project is archived.

Principal operation carried out within PMAC: organise project documentation.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condirion tests: satisfy external stakeholders; advance technical expertise; advance project management expertise.

Project initialisation

Function: to collect all information related to the project from activities that preceded the establishment of the project and document all information (e.g., initial requirements, contract).

Type: internal.

Typical input predicates: information on the contract and the initial requirements (i.e., work definition) is available@; management methodology and standards to be employed and constraints on budget and resources are known (from project environment).

Typical output predicate: project is initialised; estimation and risk models to be used are defined; standards to be used are defined (e.g., for monitoring, quality assurance, reporting, documentation); initial requirements are defined.

Principal operation carried out within PMAC: organise the information collected into a project file.

Perspective employed: project function.

Goal addressed in post-condition tests: clarify external requirements; facilitate project work.

Publicise project

Function: to make external stakeholders aware of the project and its progress.

Type: internal.

Typical input predicates: list of contacts is available@; product requirements are known; information requirements of external stakeholders are known.

Typical output predicates: list of publicity actions to be taken is available; presentations are prepared and given.

Principal operations carried out within PMAC: coordinate stakeholders' requirements with project anticipated and completed deliverables; organise presentations.

Perspective employed: project function.

Goal addressed in post-condition tests: satisfy external stakeholders; maintain relationship between plan and reality.

Quality assurance planning

Function: to safeguard the quality and standard of project deliverables.

Type: internal.

Typical input predicates: quality assurance, documentation, and reporting standards are available@; product acceptance criteria are known®; product requirements are available@, information on the project and the project environment is available; coarse standard plan is available; project risk advice is available.

Typical output predicates: quality control procedures are defined, quality assurance plan is ready.

Principal operation ca ed out within PMAC: coordinate quality assurance standards with required quality control procedures.

Perspective employed: project function.

Goals addressed in post-condition tests: maintain relationship between plan and reality; satisfy external stakeholders; facilitate project work.

Report on progress

Function: to report to external stakeholders on the progress of the project.

Type: internal.

Typical input predicates: list of contacts is available®; states of deliverables (in preparation, finished, acknowledged) are known@, list of confirmations, exceptions and potential remedial actions is available.

Typical output predicate: report on progress is prepared.

Principal operations ca ed out within PMAC: coordinate planned state with actual state of tasks, resources, etc.; organise reports.

Perspectives employed: project function; activity; time and resource; personnel.

Coals addressed in post-condition tests: satisfy external stakeholders; maintain relationship between plan and reality; facilitate project work.

Set up change control procedures

Function: to provide the procedures for controlling the introduction of changes in the product requirements.

Type: internal.

Typical input predicates: standards for change control procedures are available@ information on the project and the project environment is available; coarse standard plan is available; project risk advice is available; product requirements are available.

Typical output predicate: change control procedures are established.

Principal operation carried out within PMAC: coordinate change control procedures with anticipated changes in the product requirements.

Perspective employed: project function.

Goals addressed in post-condition tests: facilitate project work; satisfy external stakeholders; maintain relationship between project plan and reality.

Set up liaison

Function: to establish and maintain communication links with external stakeholders.

Type: boundary-spanning.

Typical input predicate: list of contacts is available®.

Typical output predicate: communication channels with the project environment are established.

Principal operation carried out within PMAC: organise project communication needs with key stakeholders and contacts.

Perspective employed: project function.

Goals addressed in post-condition tests: facilitate project work; clarify external requirements; satisfy external stakeholders; maintain relationship between plan and reality.

Set up monitoring procedures ***(detailed in section 8.4)***

Function: to set up an observation and reporting structure in such a way that information made available through direct observation or reports on particular nodes in this structure can be compared with the anticipated values at equivalent nodes in the planned project work system.

Type: internal.

Typical input predicates: actual plan is available®; monitoring standards are available®; project risk advice is available.

Typical output predicates: monitoring procedures are established (referenced through time-linked monitoring predicates); entities to be observed and reported on are identified.

Principal operations carried out within PMAC: organise observation and reporting system; coordinate reports with entities to be reported on; control allocation of reporting responsibilities; simulation test for quality control and coverage.

Perspectives employed: project function; activity; time and resource', personnel.

Goals addressed in post-condition tests: maintain relationship between plan and reality; facilitate project work; satisfy external stakeholders.

Specify requirements

Function: to specify the current requirements of the project environment on the software system to be developed in order to provide (or revise) the current definition of work required to produce it.

Type: boundary-spanning.

Typical input predicates: current requirements are available@;product change control procedures are established@; information on project and project environment is available.

Typical output predicates: work definition is specified.

Principal operations carried out within PMAC: organise current requirements from project environment into a work definition.

Perspectives employed: project function.

Goals addressed in post-condition tests: clarify external requirements; create and implement a project plan; facilitate project work.

Standard planning *(detailed in section 8.1)*

Function: to structure the activities involved in the project for the purposes of creating a plan sufficient to estimate and size the scope and contents of the project work system together with the quantities and types of resources required.

Type: internal.

Typical input predicates: product requirements (work definition) are available@; outline product specification is available; estimation models to be used are available.

Typical output predicates: standard plan is ready; resource skill profile is identified.

Principal operations carried out within PMAC: organise work definition into task hierarchy; coordinate products between tasks; control allocation of standard resources; test adequacy of standard planning.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condition tests: create and implement a project plan; satisfy external stakeholders; facilitate project work.

Tailor management process

Function: to tailor the standards and procedures recommended by own organisation to the needs and requirements of the project.

Type: internal.

Typical input predicates: information on standards and management methodology is available@;information on the project and the project environment is available; project risk advice is available.

Typical output predicates: management method and standards to be used in the project are defined.

Principal operation ca ed out within PMAC: coordinate estimated management requirements of the project with recommended management procedures.

Perspective employed: project function.

Goals addressed in post-condition tests: satisfy external stakeholders; clarify external requirements; create and implement a project plan; maintain relationship between plan and reality; facilitate project work.

Team building

Function: to build the project team(s) in a manner which facilitates project work and maintains motivation of team members.

Type: internal.

Typical input predicates: actual plan is ready@, working conditions are set up; information on the project and the project environment is available; interpersonal requirements of personnel are known (from personnel); knowledge about organisational policies, constraints, and objectives concerning personnel is available; resource skill profile is available; demands on team members from other projects are known (from project environment).

Typical output predicates: project team organisation is set up; delegation procedures are organised; communication links are established; training requirements are determined.

Principal operations carried out within PMAC: coordinate task requirements with resource requirements; coordinate team members' preferences; organise working relationships.

Perspectives employed: activity; time and resource; personnel.

Goals addressed in post-condition tests: facilitate project work; satisfy external stakeholders; maintain relationship between plan and reality.

Trigger plan

Function: to put the plan into operation.

Type: internal.

Typical input predicates: actual plan is ready®; quality control procedures are ready®; project team organisation is set up®; quality assurance plan is ready.

Typical output predicate: actual plan is activated.

Principal operation carried out within PMAC: control utilisation of resources.

Perspective employed: project function.

Goals addressed in post-condition tests: create and implement a project plan; satisfy external stakeholders; maintain relationship between plan and reality; facilitate project work.

7.2. Nature and sources of predicates needed for execution of PMACs

From all the elements we have used in providing the pre-formal descriptions of project management activities listed above. only the ***perspectives*** on the project work system and the project environment and the ***goals*** that guide the execution of the activities have been discussed in some detail in earlier chapters in terms of the roles they play in modelling the project management process.

The notion of a ***perspective*** was presented as a particular way of focusing on the project work system and the project environment, prioritising only on some of their aspects, and, as a means for obtaining structured partial views on an instantiation of the project data model, reducing the complexity of its representation so that it can be apprehended by the viewer.

The notion of a goal was presented as a guiding principle in the initiation and execution of any project management activity and the achievement of it should be a criterion for the post-condition test on the success of an activity.

In chapter 8, we will discuss the progressive refinement of PMACs into structures of subPMACs which, at the highest level of refinement, comprise operations on entities in the project data model. First, though, we will investigate briefly here the nature and sources of the predicates, the existence of which functions as a precondition for PMAC activation *or* is a result of its activation (i.e., its post-condition).

In chapter 6, we divided the predicates which serve as input to the precondition test for activation of a PMAC into two sets (P1 and P2), according to how they are formed. Predicates in set P1 were defined as originating within the project management model as a result (i.e., produced as output predicates) of the prior activation of other PMACs. Thus, for example, generation of output predicates for future use as input predicates can model sequential conditions like:

> "Product change control procedures are designed as a result of the activity of setting up change control procedures and are needed for the activity of negotiating changes".

Input predicates in set P2 do not originate within any instantiation of the project management model. Instead, they are formed and evaluated afresh each time the manager desires to activate the PMAC for which they serve as preconditions. The values assigned to these predicates are themselves instances of ***virtual entities,*** constructed from entities instantiated in the project data model in any of the following three ways:

(a) they may be imported from the project environment (e.g., demands on team members from other projects which need to be known in building the team); *or,*

(b) they may be produced as a result of the work carried out within the project work system (e.g., product design that needs to be available for negotiating any necessary changes and needed resources); or,

(c) they may be the result of inferences about the status of project work on the part of the project manager (e.g., reporting on progress involving making inferences about the quality and quantity of the project work actually having been carried out).

In case (a), the virtual entities can be interpreted as representing the constraints under which the manager has to operate. They are constructed on the basis of entities instantiated in that part of the project data model described in chapter 10 which relates to the resources provided by the project environment and the requirements imposed by this environment on the project. In case (b), the virtual entities are constructed on the basis of entities in the project data mode1 whose current instantiation describes the states of products of the project work system. In case (c), the virtual entities need to be constructed through inferences made on the basis of a consideration of the states of a variety of entities in the project data model in the manner we will discuss in chapter 9.

Chapter 8

Generating a project management model

The previous chapter has provided the pre-formal descriptions of a number of typical PMACs which can serve as building blocks in generating and instantiating a project management model for any software development project. Management methodologies often attempt to formalise the description or selection of these PMACs into a prescriptive model of fixed structure which is said to represent a particular management *methodology*.

However, software project managers rarely adhere to fixed management methodologies for long; these are too inflexible to cope with the rapidly changing and often unanticipated organisational requirements and contingencies which characterise such projects. Hence, as we argued in chapter 6, modelling the project management process needs to provide for the *gener on* of a project management model in order to offer a useful and comprehensive support for management activities on such projects. While some elements of this model (such as those corresponding to enforced management standards or stable management procedures) may be still imported into the model pre-structured, other parts will need to be generated from primitive building blocks.

In this chapter, we show how these primitive building blocks (i.e., PMACs and the subPMACs that comprise them) can be modelled. The internal structure of each of the five PMACs chosen as examples is generated in detail through refining the basic PMAC structure (shown in figure 6.3) into subPMACs. These subPMACs comprise operations which perform transformations on instances of entity classes defined in the project data model (i.e., creating, modifying, deleting them). Thus, operations are identified as activities transforming project data model instantiations. Each operation may be defined in terms of a *local process* model. *In* chapter 9, we will provide examples of further refinements within two subPMACs which would reveal the local process models they comprise.

We do not claim here that the particular subPMACs described within the internal structures we detail in this chapter must be exactly the subPMACs which a project manager will *always* use, according *to* the linkage shown. Rather, they illustrate modelling results that might typically be obtained through using the techniques for refinement of PMAC internal structures which we

described in chapter 6.

Sections 8.1 through 8.5 describe typical internal structures that may be generated for five PMACs: ***standard planning, analyse risks, actual planning, set up monitoring procedures*** and ***identify/predict discrepancies*** and ***exceptions,*** respectively. We then show, in section 8.6, how these PMACs may be linked through the input and output predicates they utilise and produce to instantiate part of the project management model.

8.1. Developing the internal structure of the standard planning PMAC

The pre-formal description given in chapter 7 for the ***standard planning*** PMAC was as follows:

Standard planning **(SP)**

Function: to structure the activities involved in the project for the purposes of creating a plan sufficient to estimate and size the scope and contents of the project work system together with the quantities and types of resources required.

Type: internal.

Typical input predicates: product requirements (work definition) are available@; outline product specification is available; estimation models to be used are available.

Typical output predicates: standard plan is ready; resource skill profile is identified.

Principal operations carried our within PMAC: organise work definition into task hierarchy; coordinate products between tasks; control allocation of standard resources; test adequacy of standard planning.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condition tests: create and implement a project plan; satisfy external stakeholders; facilitate project work.

The principal operations carried out within this PMAC define the sub-PMACs which constitute it. These are:

SP1 - organise work definition into task hierarchy;

SP2 - coordinate products between tasks;

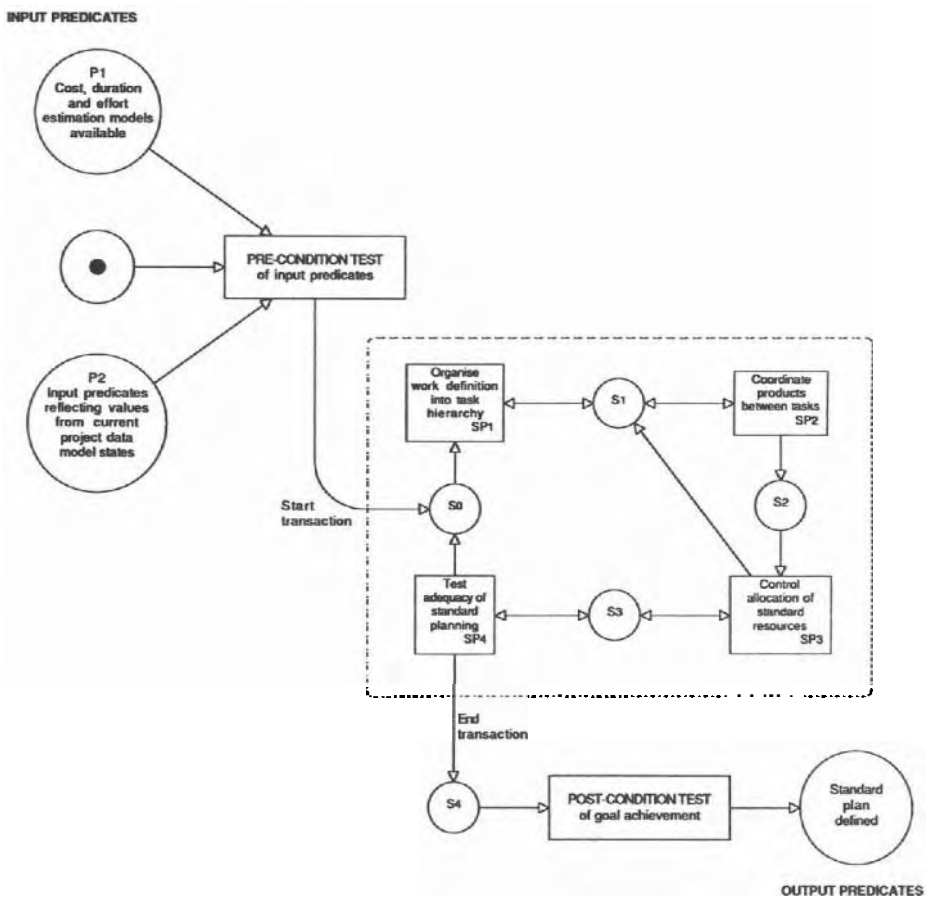
SP3 - control allocation of standard resources;

SP4 - test adequacy of standard planning.

Figure 8.1 shows a refinement of the basic ***standard planning*** PMAC structure. The PMAC's body (shown within the dotted lines) is refined to show the four principal subPMACs listed above, linked within a place-transition net. In making this refinement, the subPMACs linked in the place-transition net do not comprise, in a fixed way, the operations on the project data model that a project

manager will always carry out according to the linkage shown. Rather, they represent structures which will enable the required operations to be carried out in a coherent and functionally appropriate way, consistent with what has been described in chapter 4.

Figure 81: A standard planning PMAC refined to show its internal structure



In the place-transition formalism employed in figure 8.1, the subPMACs shown effect transitions in the way we described for predicate-transition net modelling in chapter 6. However, this place-transition net representation (of what is really a predicate-transition net) does not show the details of how output predicates are obtained and evaluated in activating the subPMACs. Instead, subPMACs effecting transitions are simply shown as being linked through a place which does not carry a fixed inscription. This indicates a typical linkage which could be instantiated through output predicates from one subPMAC (which the place links from) serving as input predicates the evaluation of which can enable the activation of *another* subPMAC (which the place links *to).* Double headed arrows, where shown, indicate a link that can be traversed in either direction.

The resulting picture (as shown in figure 8.1, and in subsequent figures employing the same place-transition net formalism) gives a rather restricted and static view of the full set of possibilities for PMAC activation offered by the predicate-transition linkage between the subPMACs in the structure. This kind of restriction is forced upon us by the need to present a view on the internal structure of a PMAC which can be presented in the static form of a book illustration. Nevertheless, we hope that the place-transition views into PMACs we provide in this chapter will convey the essential features of their internal structures adequately f a the purpose of illustrating our discussion of the refinement of particular PMACs†.

In order f a the project manager to carry out effectively the operations identified withii any subPMAC the internal structure of which has been sufficiently refined for this purpose, a view needs to be provided on the structure and content of the project data model. This view needs to be sufficient to identify the instantiated entities on which the manager needs to operate. In chapter 5, we described how four reference perspectives (i.e., the project function perspective, the activity perspective, the time and resource perspective, and the personnel perspective) are generally employed for this purpose. Appropriate views of the relevant entities within the project data model can be achieved through employing the same four reference perspectives in the manner we described in section 6.4.1‡.

In modelling operations on project data model entities, there is no *formal* reason why we have to limit the entities on which any particular subPMAC operates to those that can be viewed within a single perspective. So, in theory. and assuming that human beings had limitless information processing capacity, we could dispense with these *selective* views of perspectives and, simply. report on the state of the structure of the *whole* project data model before and after each operation. However, as we described in section 6.4.1, this is an

† To capture the full, dynamic picture, one really needs the support of a computer-based hierarchical net modelling tool like Design/CPN (Albrecht, Jensen & Shapiro 1989, Huber, Jensen & Shapiro 1989).

‡ In chapter 10. we will describe the entity classes in the core of a generic project data model instances of which may be viewed within each of these perspectives.

unrealistic requirement either for a project manager or for a support system which has to interact with a manager in presenting him with structured views of the project data model which are not too complicated for him to comprehend. This is the reason for using perspectives to provide views which prioritise certain aspects of the project data model.

Taking a view within a perspective which seems natural to the project manager helps him apprehend the relevant aspects of the structure of the current instantiation of the project data model immediately ***before*** any particular subPMAC operation is carried out (*so* he can know what is to be done) and immediately ***after*** it has been carried out (*so* he can know what has been achieved)?.

In the refinement of the ***standard planning*** PMAC shown in figure 8.1, the input predicates required to activate subPMAC SP1 ***(organise work definition into a task hierarchy)*** are generated as output predicates from the PMAC's pre-condition test, given that this test was successful.

Figure 8.2 illustrates how subPMAC SP1 can then be carried out through building on an instantiation of entities in the project data model viewed within the ***activity*** perspective. This involves inheriting the work definition identified in the predicates which are input to place SO, and then focusing on the tasks which need to be carried out by the project work system to produce the required products. SubPMAC SP1 serves to ***organise*** the decomposition of the work definition implicit in the requirements for the project into a task hierarchy. Each leaf task in the hierarchy then implements part of the work in the work definition. Higher level tasks may be viewed as ***virtual*** tasks as they define particular sub-groupings of leaf tasks which define the actual work components of the project work system.

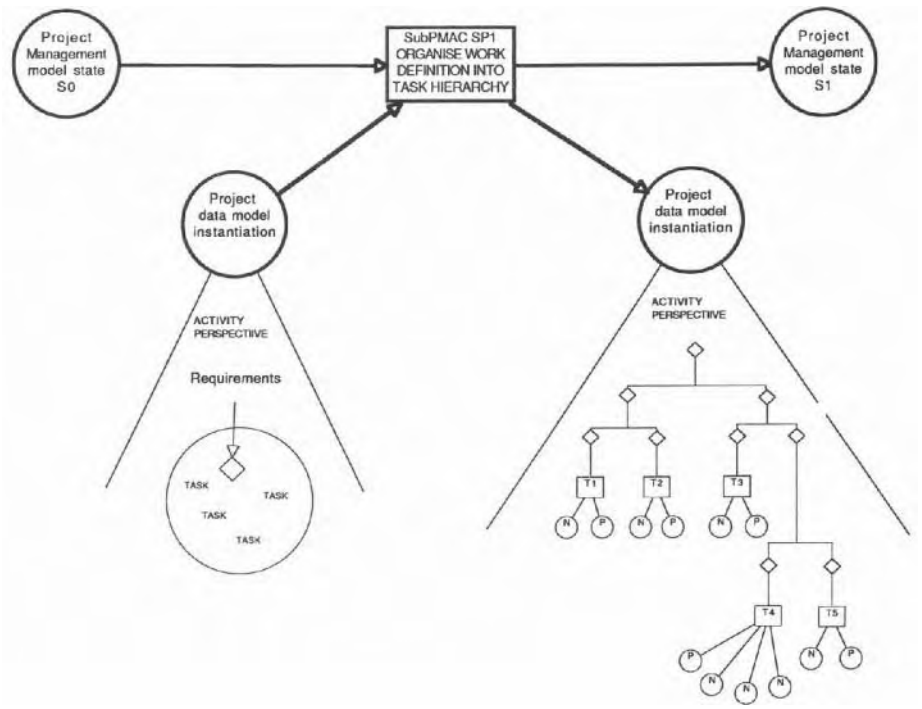
An estimation of the effort required to achieve the part of the work defined for each individual leaf task is usually made at this time, together with a specification of the skill profiles for the various resources which will be employed to provide this effort. Effort estimates for higher level task groupings, and ultimately for the whole project, can then be formed by summing the estimates for individual tasks bottom-up through the instantiated task hierarchy. In the case where a global effort estimate exists for the project ***a priori,*** this can be decomposed, top-down, through the task hierarchy to provide effort estimates for individual tasks. The latter can then be compared with the equivalent estimates generated bottom-up, with the differences having to be reconciled by the project manager as he tunes the way he has organised the work definition into a task hierarchy*.

---

† It is also often helpful to be able to view the structure *as it changes* during the operations carried out within a subPMAC, but this level of refinement is not shown in the figures presented in this chapter.

‡ The topic of estimation (of effort, duration and cast) and the methods which may be used for making such estimates are discussed in more detail in chapter 12.

Figure 8.2: SubPMAC SP1 (organise work definition into a task hierarchy)



A hierarchical decomposition of tasks, viewed on its own, can give the misleading impression that the leaf tasks are independent of each other as the hierarchy shows only the ***inheritance*** of individual components of the work definition but not any dependencies between these components. However, absence of dependencies between tasks is the exception rather than the rule in the reality of the project work system. So, the workbreakdown structure usually compensates for the incompleteness of the impression that may be gained by thinking about tasks purely in terms of inheritance of work definitions by giving equal priority to indicating the key products that each leaf task ***needs*** and ***produces.*** However, connecting and checking consistency of the various tasks' ***needs*** and ***produces*** through the various referenced products is usually left to be the focus of the next subPMAC to be activated (i.e., SP2) as it is difficult for the project manager to get a good working view on the various interrelations between leaf tasks while concentrating on what these tasks inherit within a hierarchical task decomposition.

As a result of carrying out subPMAC SP1, place S1 (as shown in figure 8.1) is reached. Now, the project data model is viewed within the activity perspective: the foreground may be structured in a way that prioritises the hierarchical property of the instantiation of task entities made in SP1, together with their individual *needs* (N) and *produces* (P) attributes. The presentation of this task structure to the manager, viewing it within the activity perspective, typically follows the conventions for displaying a *workbreakdown structure* (Tausworthe 1980).

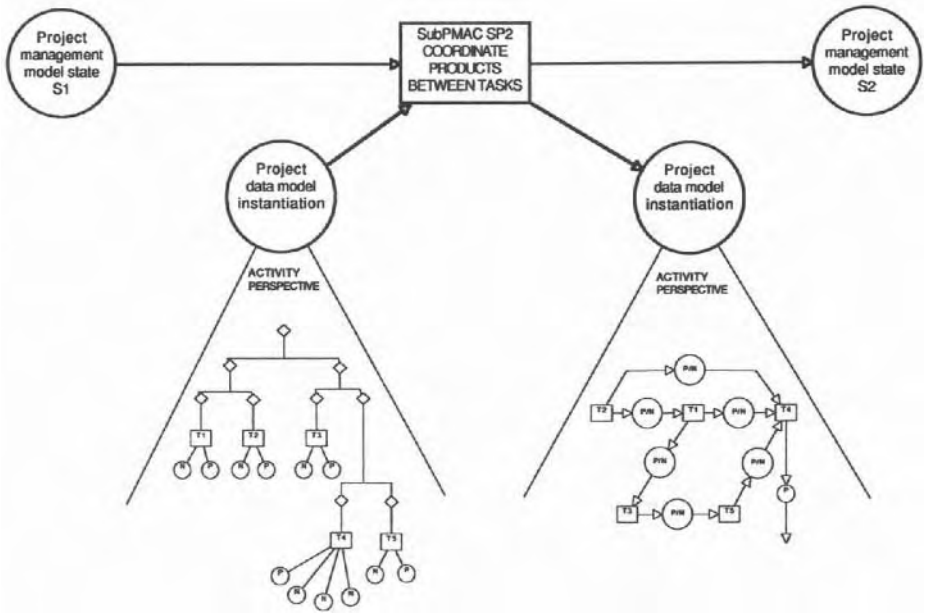
Moving on from place S1, the subPMAC which may be carried out next, SP2 (*coordinate products between tasks*), performs the coordination of the values assigned to the *needs* and *produces* attributes of the tasks in the task structure produced through the activation of SP1. The result of activating subPMAC SF2 is *to* establish *precedence relations* between tasks (i.e., a task which produces a particular product must *precede* any *task* which needs this product).

The view of this task structure, presented to the manager within the activity perspective, now needs to prioritise the precedence relations rather than the hierarchical relations between the tasks in the instantiation. The display usually shows just the leaf tasks in the hierarchy (as only these have *needs* and *produces* values instantiated), together with the duration estimates, available or made at this time, for each task. Typically, it will follow the conventions for displaying a CPA (Critical Path Analysis) or PERT (Program Evaluation and Renew Technique) diagram (Weinwym 1970, Cori 1985. Sneed 1989).

Figure 8.3 shows how the five leaf tasks (T1 through T5) shown in the "workbreakdown" partial view of the project data model instantiated through subPMAC SP1 may now be incorporated in a PERT diagram, with the precedence dations between the tasks established by linking the requisite *needs* (N) and *produces* (P) relationship attributes. Each P/N link *can* also serve to identify, and locate within the sequential task structure, the products which constitute the inter-task products referenced in the P/N linkage.

It would be incorrect to assume, though, that creating the PERT diagram somehow *transformed* the task structure instantiated in the project data model fium a workbreakdown structure to a PERT structure. On the contrary, the task structure, as seen in the workbreakdown view, should not change as a result of the operations carried out in subPMAC SP2. In fact, it is a good idea for the manager to maintain the workbreakdown view while developing a CPM or PERT diagram. This *can* prove particularly useful in tracking down the sources of missing links which prevent the construction of a *coherent* CPM or PERT representation. Quite often, this tracking process will reveal that some part of the hierarchical task decomposition is unsatisfactory. Thereupon the exit route from subPMAC SP2 is back along the link to place S1, and, from there, to the re-activation by the project manager of subPMAC SP1 in order to correct the problem, before activating SF2 again for another attempt at developing a coherent CPM or PERT diagram.

Figure 8.3: SubPMAC SP2 (coordinate products between tasks)



In the case that the post-condition test incorporated in **subPMAC SP2** indicates that its goal of **coordinating tasks** and products through ***needs*** and ***produces*** has been properly achieved, completing the activation of **subPMAC** SP2 **results** in the move to place S2 (**as** represented in **figure** 8.1), **and,** from there. to **subPMAC SP3** ***(control allocation of standard resources).***

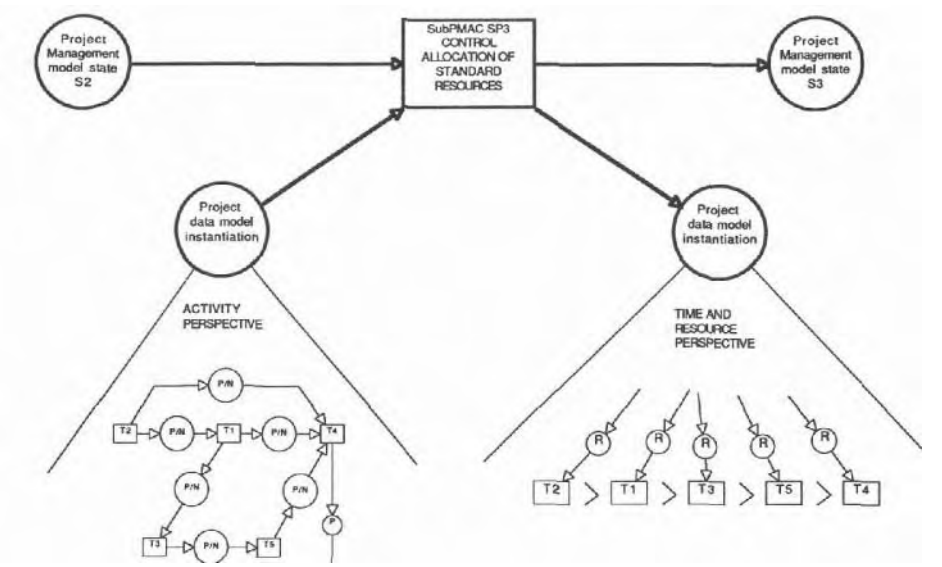
Estimation of the **resources** required to ***get*** the **work** done **can** now be made through ***controlling*** the allocation of resources which **are.** needed to implement the **tasks organised** in the task hierarchy now instantiated within the **project data** model. However, within the **activity** perspective employed in viewing the entities operated on by **subPMACs SP1** and SP2, the definition of the potentially **or** actually available individual **resources** may still be obscured **as** the emphasis is on "getting the work done" (in the foreground), **assuming** the work will, somehow, be **resourced.** Hence, estimation of the **required** resources is, at this stage, often based on the allocation of "standard" resources **(i.e.,** imagined resources with **standard** characteristics rather than actual **resources** with **given** characteristics), **as** it is not yet clear or known ***which*** resource will ***be*** available at the time it is needed, or, for that matter, exactly ***when*** it will ***be*** needed. This is because, while sequential dependencies have been established between **tasks** (through the **P/N** linkage), the tasks themselves have not been slotted into the **common time** frame.

SubPMAC SP3 has the **task** of creating this common time **frame through controlled** allocation of **standard** resources. It **uses** the **project data** model instantiation created **through** the operation of **SP2,** but, through **presenting** it for view by the **project** manager within the **time** and resource perspective in the

way indicated in figure 8.4, it can determine the sequence of tasks and allocate resources to them according to the tasks' resource requirements. In this instance, standard planning rules may be activated to inform the proportion of resources (i.e., relative effort) that will be needed by each of the various groups of tasks (e.g., designing tasks, coding tasks) throughout the development period (see, for example, Brooks 1982, Fairley 1985, Schlumberger 1986).

When the instantiated tasks have been resourced, the labour costs for the project can be estimated on the basis of cost per unit time for each type of resource allocated to a task for the duration of the allocation. Making and checking such estimates, both bottom-up and top-down (in a manner analogous to that we described for effort estimation in subPMAC SP1), usually completes the activation of subPMAC SP3. If the project manager judges the resulting pattern of resource allocations and cost estimates to be satisfactory, he can then move through place S3 (as shown in figure 8.1), and activate subPMAC SP4 in order to test the adequacy of his standard planning.

Figure 8.4: SubPMAC SP3 (control allocation of standard *resources)*



In the case, though, that the manager judges that the activation of sub-PMAC SP3 he has just completed gave an unsatisfactory result, he is likely to move along the direct link to place S1 (shown in figure 8.1) and take a hierarchical (workbreakdown) view of the task decomposition he has currently instantiated in his standard planning. This can inform the decision he has to take at this point about how to improve this result.

For example, he may decide to re-activate subPMAC SPI in order to revise some of his workbreakdown, splitting big tasks that are difficult to resource adequately, or wmbining into one large task smaller tasks which require similar resource skill profiles but which, individually, would under-utilise the resources assigned to them. Alternatively, the project manager may decide to keep the workbreakdown in its current form, and move from place S1 to re-activate subPMAC SP2 in order to change some duration estimates and, hence, hopefully, have the chance to achieve better resource levelling when he subsequently re-activates subPMAC SP3.

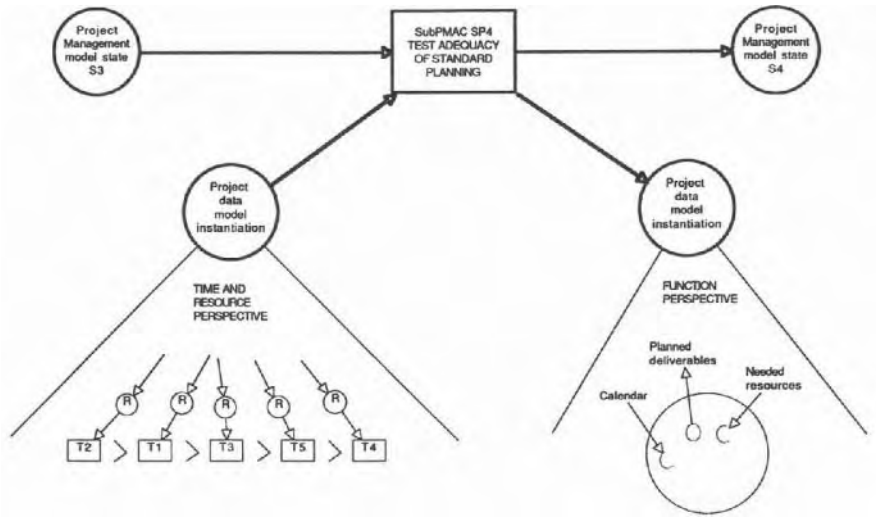
Once the manager is satisfied with the results of activating subPMAC SP3, and so moves to place S3, standard planning is complete in the sense that the project plan is structured in a way that it *could* be operated to produce the instantiated products. The adequacy of the way in which the plan has been structured may then be tested by carrying out subPMAC SP4. This subPMAC checks the standard plan for coherence and its ability to handle test scenarios (e.g., seeing how well envisaged contingencies are met). This may involve, for example, testing for conditions that the plan should meet such as those that were described in section 4.2.2.3 as potential problem areas which may be identified in a project plan.

However, the concerns of SP4 are *global* concerns with the plan as, by definition, a standard plan cannot address specific ones (e.g., those related to specific resource utilisation). A standard plan is, to a great extent, an ideal plan which will eventually be brought close to reality as a result of actual planning. Thus, as figure 8.5 indicates, subPMAC SP4 is generally carried out by taking the project function perspective on entities in the project data model which are instantiated in the current plan since the major question this subPMAC needs to answer is: "if we had this plan as our plan to run the project, could the work defined within it produce deliverables† which meet the project requirements by utilising resources in this particular way?".

If the results of the test of the adequacy of the current standard plan made through activating subPMAC SP4 is unsatisfactory in the view of the project manager, he has two alternatives. He can decide that the standard plan needs a complete rethink, starting from how best to organise the work definition into a task hierarchy. In this case, he will move to place SO where he can review the original work definition, as it was presented prior to his attempt to organise it, and then re-activate subPMAC SP1.

† A deliverable consists of a set of products delivered by the project to the client or other external stakeholders.

Figure 8.5: SubPMAC SP4 (test adequacy of standard planning)



Alternatively, he may decide that the necessary improvements can be achieved through more *local* revisions on details of the existing plan. In this case, his preferred course of action will usually be to start by reviewing what was actually achieved through his task-coordinating and resource-allocating activities which led to his current plan. This can be achieved by returning through the linkage S3 -> S2 -> S1 and examining, along the way, the views on the project data model offered through the various perspectives the use of which we have associated with these places.

If, on the other hand, the results of the adequacy test made in subPMAC SP4 is acceptable to the project manager, he can now turn from an internal assessment of his standard plan and move through place S5 to the post-condition test which examines how well the standard plan he has produced actually meets his project management goals.

As we indicated in our pre-formal description of the *standard planning* PMAC, this means examining more than the achievement of the goal that triggered the activation of this PMAC (i.e., *create and implement a project plan). In* the wider context of the acceptability of the plan to other stakeholders in the project, the project manager should also check how well his goals to *facilitate project work* and *to satisfy external stakeholders* have been achieved. For example, the cost estimates produced through the activation of subPMAC SP3 may turn out to imply a total project expenditure which exceeds the current budget constraints imposed on the project. The project manager may have decided, during the activation of subPMAC SP4, that his current plan needs a particular level of resourcing to be a viable implementation of the current work definition, even though this produces labour cost estimates which indicate that the project is likely to go over budget. From the point of view of the post-condition test of goal achievement, this risks upsetting the achievement of the

goal to *satisfy external stakeholders:* either the client will have to pay more than he intended, or the manager's own organisation will have to accept a lower profit margin on the project than what was originally anticipated or may even incur a loss.

In this case, the post-condition test may suggest that, before this standard plan can be accepted as input data for other PMACs (e.g., *actual planning),* it would *be* advisable to activate another PMAC which could help satisfy the goal the achievement of which is now risked. For example, the priority now given to *satisfy external stakeholders* could promote the activation of PMACs *negotiate resources* or *negotiate changes* immediately on completion of the *standard planning* post-condition test. The output predicates from the post-condition test would (in this particular case) carry the information that *either* the budget constraints should be re-negotiated upwards (so that the current standard plan could be retained safely) hence activating the *negotiate resources* PMAC *or* the work definition should be revised and thus activate the *negotiate changes* PMAC. *In* the latter case, it would, subsequently, be necessary to re-activate the *standard planning* PMAC with the hope of constructing a new, less expensive, standard plan on the basis of the revised work definition.

While a standard plan which passes the post-condition test of goal achievement in the *standard planning* PMAC could, in theory, provide the basis for the actual running of the project work system, it would be likely to produce very inefficient results if employed directly in this fashion. It would always require that the required amounts of the "right son" of resources were available when needed, and yet not be able to specify *a priori* where they could be needed, or when the external deliverables would be finally produced. Hence, the standard plan will need to be tailored to take into account resources which may actually be available at any particular time with the aim of utilising them in an efficient way. This is the objective addressed by the *actual planning* PMAC described in section 8.3 below.

In actual planning, tradeoffs have to be made between the relative advantages and disadvantages of particular ways of scheduling tasks in real time and assigning particular individual resources to those tasks. These disadvantages need to be understood in terms of the potential risks that may be run in regard to the project's successful progress and outcome, mitigated by how well they may be managed when starting from each alternative version of the actual plan which is being considered by the project manager for implementation. Thus, the efficiency of actual planning may be considerably improved if the appropriate risk advice is available at its outset. The provision of risk advice is the primary objective of the *analyse risks* PMAC. Hence, we will examine first how the internal structure of that PMAC may be developed so that its activation will serve to develop appropriate risk advice which will be needed for the activation of the *actual planning* PMAC (discussed in section 8.3), among others.

8.2. Developing the internal structure of the analyse risks PMAC

The pre-formal description given in chapter 7 for the ***analyse risks*** PMAC was as follows:

Analyse Risks (AR)

Function: to analyse the risks of the project.

Type: internal.

Typical input predicates: project has been initialised®; product requirements are available®; risk model is available®; coarse standard plan is available; feasibility study results are available; information on previously archived projects and case studies are available.

Typical output predicates: risk advice is available (i.e., project risk profile is available together with its implications for management).

Principal operations carried out within PMAC: organise information on project risks into project risk profile; coordinate risk profile patterns with risk management rules; test adequacy of risk analysis; coordinate information needs with information available from organisational sources.

Perspective employed: project function.

Goal addressed in ***post-condition tests:*** clarify external requirements; satisfy external stakeholders; facilitate project work.

The principal operations carried out within this PMAC define the sub-PMACs which constitute it. These are:

AR1 - organise information on project risks into project risk profile;

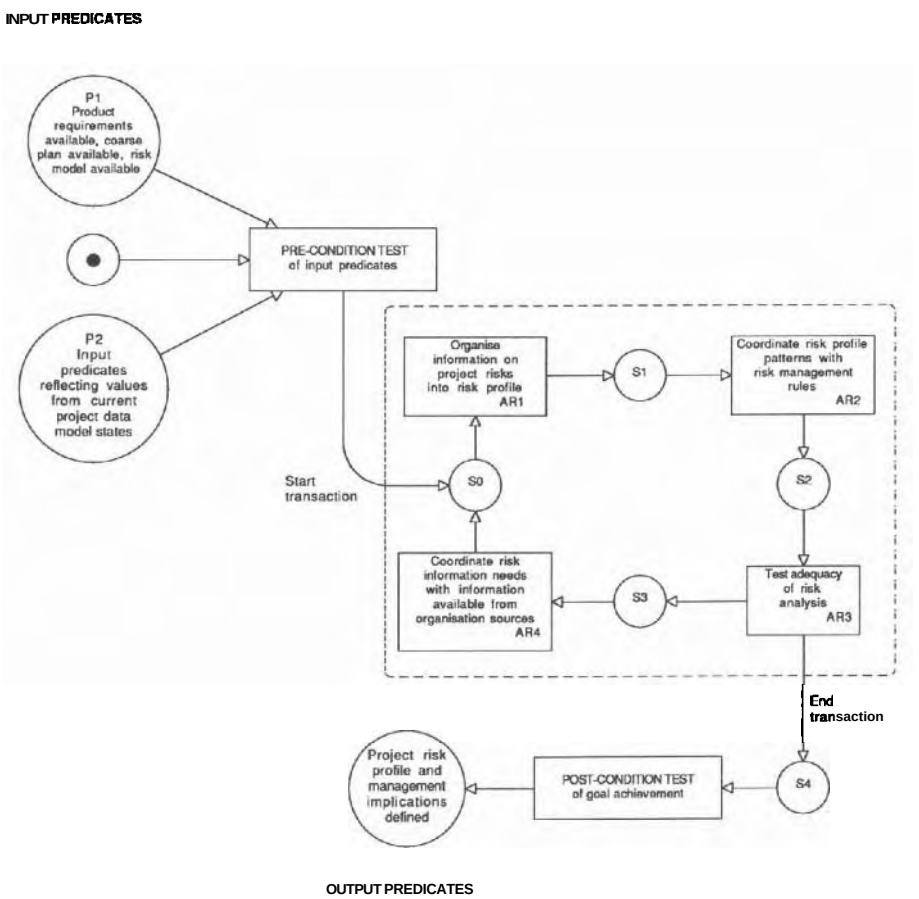
AR2 - coordinate risk profile patterns with risk management rules;

AR3 - test adequacy of risk analysis;

AR4 - coordinate information needs with information available from organisational sources.

Figure 8.6 illustrates the internal structure of PMAC AR, refined to show its four constituent sub-PMACs within a place-transition net. We assume, in the following, that the internal structure of this PMAC has been designed to incorporate the "blackbox" approach to risk analysis which we described in section 3.2, according to which, the impact of external sources of risk on the project is analysed in terms of a model based on ***risk factors.*** From the results of this analysis, advice on risk management issues is developed for use in a variety of PMACs which may be activated subsequently. For this reason, in the pre-formal description for this PMAC given above, it is stated that the input predicate ***risk model is available*** is ***required*** for the pre-condition test to succeed. However, in following the "blackbox" approach, we need not insist on the prior availability of a standard plan for activation of this PMAC (although it helps to have one). This is in contrast to the pre-conditions for a "whitebox" approach to risk analysis which would ***insist*** on the prior existence of a standard plan.

Figure 8.6: An analyse risks PMAC refined to show its internal structure

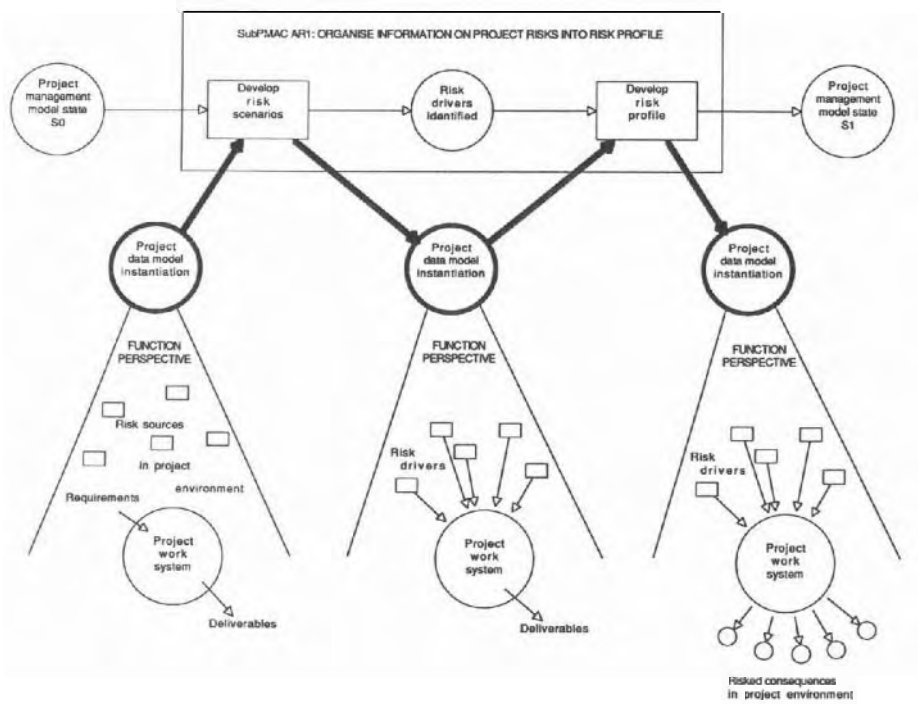


The first subPMAC to be activated as the result of a successful precondition test (AR1) has the aim of organising information on project risks into a risk profile. As illustrated in figure 8.7, it starts with an initial view on the project data model within the project function perspective, which should identify ***potential risk sources*** in the project environment. The project manager's initial task within this subPMAC, however, is to discover which of the many and diverse potential risk sources which can be viewed in this perspective actually need to be considered as ***risk drivers*** the assessment of which will provide the input to the project risk model.

Berkeley, Humphreys and Thomas (1990) identify the following two key requirements for any risk driver which will be useful in practical risk management:

(1) It must be observable (if looked for) in the context of the particular project ***before*** the risk which it "drives" may actually occur as a consequence of the prior existence of the risk driver at a particular level. If there is uncertainty about the impact of a prior observed level of a driver on a particular risked consequence, then the degree of dependency of the risk on the driver must itself be at least approximately quantifiable.

Figure 8.7: SubPMAC AR1 (organise information about project risks **into** risk profile)



(2) It must be reliably quantifiable in relation to the **risk(s)** which it drives in terms of ordered levels (even if only *low, medium,* high, or *present* versus *absent)* on a *risk factor* the description of which indicates what is to be observed *(or* to be directly **inferred** from what is observed). This quantification must also identify explicitly how the project risks which are "driven" by the risk driver *depend* on the observed level of the risk driver: that is, we must know which levels are likely to drive *up* or drive *down* the likelihood of the risked consequence occurring, compared with that of the baseline assumed in estimating key project parameters indicating degree of goal achievement **(i.e.,** relating to cost, duration, quality, morale, etc.).

In figure 8.7 we have further refined the internal structure of **subPMAC** AR1 to indicate the separate activities which relate to each of these two conditions. As we described in section **3.2.2,** "observing" a risk driver before the risk it drives **occurs** may involve exploring the worlds of the client, of the contractor **organisation,** of other stakeholders, of (potential) suppliers and subcontractors, and of the (potential) project team. These explorations are used for developing *risk scenarios,* looking into the **future** and identifying the key risk drivers which may adversely affect the project.

As illustrated in figure 8.7, the result of this activity should provide a view on the current instantiation of the project data model, within the function perspective as before, but now presenting the key risk drivers, and their potential impacts on the project, in the foreground. This contrasts with the initial, much more hazy and diverse, picture of risk sources that was available at the initiation of this **subPMAC.**

The **second** activity within **subPMAC AR1** makes use of the risk model to translate the information now available on the risk drivers into an account of the potential consequences for the project. Usually, this is presented for viewing by the project manager, in a fairly compact and global way, as a *project risk profile* (Cash. **McFarlan & McKeeney** 1983). For example, Cats-Baril, Humphrey~and **Wanless** (1988) describe how a global project risk profile may be developed, showing the overall degree of project risk in the following five domains: **(1)** size and complexity of the **project,** (2) knowledge within the **organisation** about the project environment and application area, **(3)** the technology (software and hardware) needed to **carry** out the project, (4) the characteristics of the client, and. (5) the type of **contract** covering the project. Within each of these domains, a **specific** risk proiile is also developed, displaying for the manager the degree of risk **associated** with **various** types of consequences ranging over an average of **12** domain-spec& factors.

It would be foolhardy to expect the result of activating **subPMAC** AR1 to identify the **precise** risked consequences which will actually occur in the future. This **kind** of **clairvoyance** is not likely to be possessed by a project manager and certainly not by any risk analysis technique. The project risk profile, at its best, can be informative only about the likelihoods that various types of consequences may occur, given the information about the risk drivers supplied to the risk model. However, this does not mean that a risk analysis should end at this

point. As we discussed in section 3.3, effective risk management requires that the project manager be able both to ***anticipate*** the risks to the project and to ***design*** suitable organisational structures within the project to **minimise** their negative practical impacts.

To achieve this in practice, the project manager needs to examine what sort of project risk "intelligence" he requires to work out the most effective strategies and techniques to use to manage the identified levels of risk in the context of his particular project. This can be facilitated through the prior provision of ***project risk advice,*** organised in such a way that the appropriate advice is made available (and referenced through input predicates) for the various **PMACs** which inform and determine these strategies and techniques.

This is the objective of **subPMAC** AR2, which examines what kind of risk management rules should be **triggered** to provide risk advice on the basis of patterns which can be discerned in the project risk profile. The particular advice may consist of recommendations on how to perform (or what to watch out for when performing) workbreakdown, change control, quality assurance, etc. It may indicate the kinds of user involvement, staff skills, leadership and management techniques which may need special consideration **within** PMACs which will subsequently be activated. One of the most difficult problems for the project manager is to assemble a comprehensive list of **rules** of this type and to know how to identify the patterns which should trigger them. Thus, usually, this process is accomplished in an intuitive and informal **way†.**

On completion of this activity, the project manager moves (via place S2, as shown in figure 8.6) to the point where he should test the adequacy of the risk analysis, that is, **subPMAC AR3.** Making this test would be quite easy later on, with hindsight, when it is known which of **the** risked consequences did actually materialise and how appropriate was the risk analysis that was **produced.** This sort of feedback is, of course, not available at the time that the test of the adequacy of the risk analysis has to be made within this PMAC. Instead, the test has to rely on looking at the adequacy of the information currently provided about risk drivers in terms of whether it leads to apparent anomalies, or unusual patterns, in the risk profile, or to gaps and inconsistencies in the risk advice which the manager judges should be forthcoming.

If the test of the adequacy of the risk advice fails, then the project manager can **move** (through place S3) to **subPMAC** AR4 ***(coordinate information needs with information available from organisational sources).*** The ***information needs*** will ***be*** those specifically required to resolve the anomalies which the test in **subPMAC** AR3 discovered. The needed information can then be gained by the manager from the identified organisational sources in the project environment. Given the successful activation of **subPMAC** AR4, the project

† **Nevertheless, risk analysis techniques developed within two different ESPRIT I project management support system prototypes have, with some success, been able to incorporate simple rule bases and triggering mechanisms for this purpose. In each case, the rules in the rule base were collected through interviews with experienced project managers** (Cats-Baril & **Wanless 1988,** Moynihan, McCluskey & **Verbruggen 1989).**

manager can move to place SO for the second time and, hence, to a re-activation, in sequence, of subPMACs AR1 and AR2 in the effort to generate a revised risk profile and fresh project risk advice which, hopefully, will satisfy the adequacy test in subPMAC AR3.

In the case that the project manager is satisfied with the adequacy of the risk analysis, he can then move (via place S4 in figure 8.6) to the post-condition test of goal achievement. As well as testing the goal which triggered the AR PMAC (i.e., to ***clarify external requirements),*** this test ***can*** also check whether the risk analysis meets the goal to ***satisfy external stakehoders,*** particularly those senior managers in the contractor organisation who may wish to use the results of the risk analysis for their own external assessment of the project and its potential implications for the organisation as a whole. It is also desirable to see if the risk advice may be employed to ***facilitate project work,*** particularly by sharing risk advice with team members who may be able to conmbute to effective risk management through taking this advice into account as they conduct their own activities.

8.3. Developing the internal structure of the actual planning PMAC

The ***standard planning*** PMAC described in section 8.1 is an appropriate means to the end of creating and implementing a plan for the pmject during the initial planning phase of the project. At this time, prior to the launching of the project, "standard" resources may be imagined in the planning as a basis for making estimates, identifying the need to negotiate desirable changes with external stakeholders, providing a context for quality assurance planning, and so on. However, at the time the project is launched, and subsequently, ***actual planning*** must ***be*** employed: that is, planning based on ***actually available*** ("real") resources at any particular time. Thus, actual planning must involve coordination with a time axis which will also provide a monitorable schedule for the project tasks. Its pre-formal description was given in chapter 7 as follows:

Actual planning (AP)

Function: to plan on the basis of knowledge about actually available resources to the project at any particular time.

Type: internal.

Typical input predicates: standard plan is available®; resources committed are known®; calendar is known®; risk advice is available; cost, duration and effort estimation models to be used are defined; resource skill profile is available.

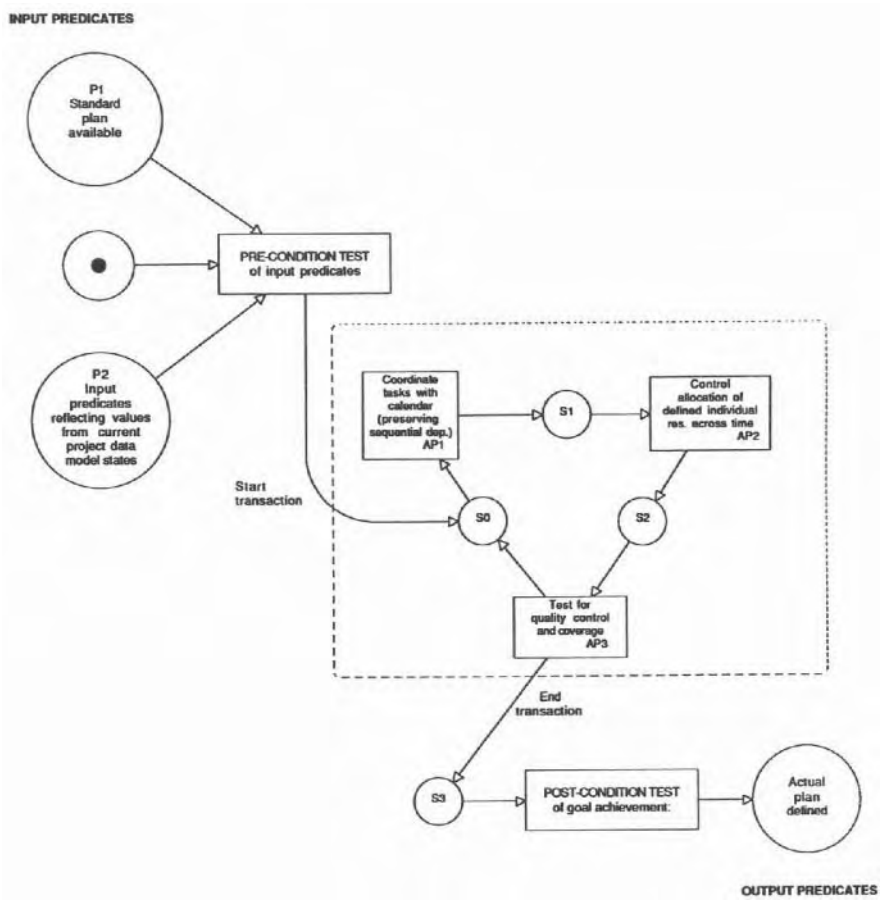
Typical output predicate: actual plan is ready.

Principai operations carried out within PMAC: coordinate tasks with calendar, control allocation of defined, individual resources across time; test for adequacy of the actual plan.

*Perspectives **employed**:* activity; time and resource; personnel.

*Goals addressed in **post-condition** tests:* create and implement a project plan; satisfy external stakeholders; maintain relationship between plan and reality; facilitate project work; advance technical expertise.

Figure 8.8: An actual planning PMAC refined to show its internal structure



The principal operations carried out within the PMAC define the sub-PMACs which constitute it. These are:

AP1 - coordinate tasks with calendar.

AP2 - control allocation of defined, individual resources across time;

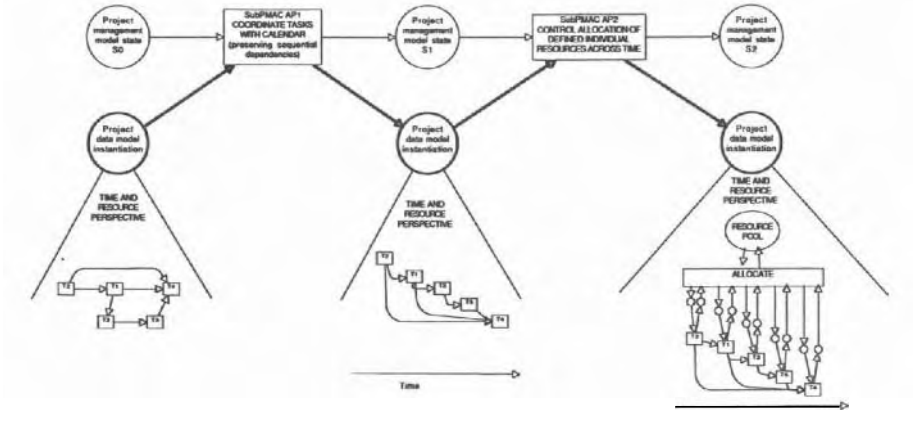
AP3 - test for adequacy of the actual plan.

Figure 8.8 illustrates the internal structure of PMAC AP refined to show its three constituent subPMACs linked within a place-transition net.

Use of the actual *planning* PMAC, in practice, requires that a standard planning PMAC has been used previously to build a coherent task hierarchy with *needs* and *produces* links properly coordinated between tasks, and that effort estimates (expressed in terms of "standard" resources) are available. These assumptions are expressed *as input predicates* to the *actual planning* PMAC and are tested in the *pre-condition test* for the activation of the PMAC's internal structure-

Figure 8.9 shows subPMACs AP1 and AP2 operating on an instantiation of project data model entities presented for viewing by the project manager within the *time and resource perspective.*

Figure 8.9: Use of subPMACs AP1 and AP2 in developing a project data model instantiation



First, subPMAC AP1 is employed to develop the instantiation to coordinate the task hierarchy existing within it (with *needs* and *produces* links in place) against a global clock (i.e., calendar) within a common time frame. Then, subPMAC AP2 can be used to extend the instantiation to define the allocation of individual resource instances (i.e., particular people) from the available resource pool to the scheduled tasks, while being able to check resource availability and redundancy at any time as resource instances are taken from the pool for use on tasks, and released from finished tasks to be returned to the pool. This will be achieved through employing the time and resource perspective to view the relevant project data model entities.

However, if only the time and resource perspective is used, in the resulting instantiation of project data model entities, individuals, viewed solely as human resources, are likely to be allocated to tasks according to their availability, skills and experience, without taking into account considerations which become prominent within the personnel perspective (e.g., job satisfaction, willingness to work with specific others, career advancement). Thus, actual planning is not likely to be complete or, eventually, successful unless the personnel perspective has been used as well as the time and resource perspective in examining the team formations effected in the project data model instantiation developed through the use of subPMAC *AP2*.

SubPMAC AP3 can then test the adequacy of the actual planning through viewing the resulting project data model instantiation, in turn, through *all* the perspectives which the project manager should also adopt in viewing the project work system and the project environment. The test carried out in subPMAC AP3 may, for example, fail on coherence grounds if the resource pool is exhausted at some point before the resource allocation process is completed. It may fail on functional grounds if there is insufficient resource levelling evident in the overall allocations made throughout the time frame. The test may also fail if the schedule resulting from the use of subPMAC AP2 does not fit into the overall time frame set by inherited project requirements constraints.

The post-condition test of goal achievement, as well as addressing the goal which triggered the PMAC (i.e., to ***create and implement a project plan),*** should also examine the degree to which the resulting actual plan would help, if it were implemented in reality, to achieve the other goals identified in the pre-formal description of the PMAC. It should check, for instance, whether the plan is likely to ***satisfy external stakeholders*** (e.g., is the plan within the required budget and duration? are the external deliverables produced at the right time?). It should determine whether the plan can contribute to the goal to ***advance technical expertise*** by providing opportunities to team members to improve their skills by working on particular tasks. The post-condition test should also check whether the goal to ***facilitate project work*** is also met by not causing difficulties for the project team such as those created by uneven working patterns for team members, or rapid switching of individuals between incompatible tasks. Finally, the plan should also be tested in terms of its ability to meet the goal to ***maintain the relationship between the plan and reality*** (e.g., does it promote the development of good monitoring procedures and the

future identification of discrepancies and exceptions?)†

If the post-condition test indicates that the proposed actual plan fails to meet these goals, the current attempt to use ***actual planning*** subPMACs in planning the project work is then "undone". Instead, the ***standard planning*** PMAC may ***be*** re-activated by the project manager in order to re-organise his standard plan (e.g., create a new task decomposition) before making a new attempt to use the ***actual planning*** PMAC starting from a revised input from the re-activated ***standard planning*** PMAC. Alternatively, another goal may now be pursued instead, leading to the selection of a different PMAC. For example, changing his immediate goal to the goal to ***clarify external requirements*** for the project may lead the project manager to select the ***negotiate changes*** PMAC, with the intention of negotiating more feasible requirements for the work that has to be planned.

8.4. Developing the internal structure of the set up monitoring procedures PMAC

Work that is to be carried out within the project work system is monitored against the final actual plan. However, in order to compare what is planned for values of entities instantiated in the project data model with what is happening in regard to the states of the objects these entities represent in the reality of the project work system, it is first necessary to get access to that reality. This requires a structured view on the instantiation of the relevant entities in the project data model set up in such a way that the information made available through direct observation or reports on objects in the project work system represented in this instantiation can be compared with the anticipated values given previously to the same instantiated entities according to the current project plan.

The process of setting up the observation and reporting structure which will provide the values for the instantiated entities is effected through a PMAC the characteristics of which were described in a pre-formal way in section 7.2 as follows:

Set up monitoring procedures (SM)

Function: to set up an observation and reporting structure in such a way that information made available through direct observation or r e p s on particular nodes in this structure can be compared with the anticipated values at equivalent nodes in the planned project work system.

Type: internal.

† In sections 8.4 and 8.5, we will examine the role played by the actual plan in this respect within the internal structure of *the set up monitoring procedures* and *identify/predict discrepancies and exceptions* PMACs, respectively.

Typical input predicates: actual plan is available@; monitoring standards are available@, project risk advice is available.

Typical output predicate: monitoring procedures *are* established (referenced through time-linked monitoring predicates); entities to be observed and reported on are identified.

Principal operations carried out within PMAC: organise observation and reporting system; coordinate reports with entities to be reported on; control allocation of reporting responsibilities; simulation test for quality control and coverage.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condition tests: maintain relationship between plan and reality; facilitate project work; satisfy external stakeholders.

The principal operations canied out within this PMAC define the sub-PMACs which constitute it. These are:

SM1 - organise observation and reporting system;

SM2 - coordinate reports with entities to be reported on;

SM3 - control allocation of reporting responsibilities;

SM4 - simulation test for quality control and coverage.

Figure 8.10 shows how an internal structure generated for this PMAC may be represented as the four subPMACs within a place-transition net†. This PMAC is activated on the successful completion of a pre-condition test which examines the actual plan for the project to identify entities in the project data model which can be monitored according to the monitoring standards identified in the input predicates to this PMAC. If no plan exists, or it *cannot* be monitored (e.g., it is not detailed enough), this PMAC cannot be activated.

The initial subPMAC activated within the internal structure, SM1, serves to organise the observation and reporting system‡. The entities instantiated in the project data model which will receive their values through the operation of this observation and reporting system in reality are then matched with the points in the project work system where such reports are used for monitoring anticipated realities and results (subPMAC SM2). SubPMAC SM3 may then be employed to allocate observation and reporting responsibilities to the project manager and to team members. A simulation test may then be made (subPMAC SM4) involving generation of reports, according to the responsibilities that have been set up, and checking them against, for example,

---

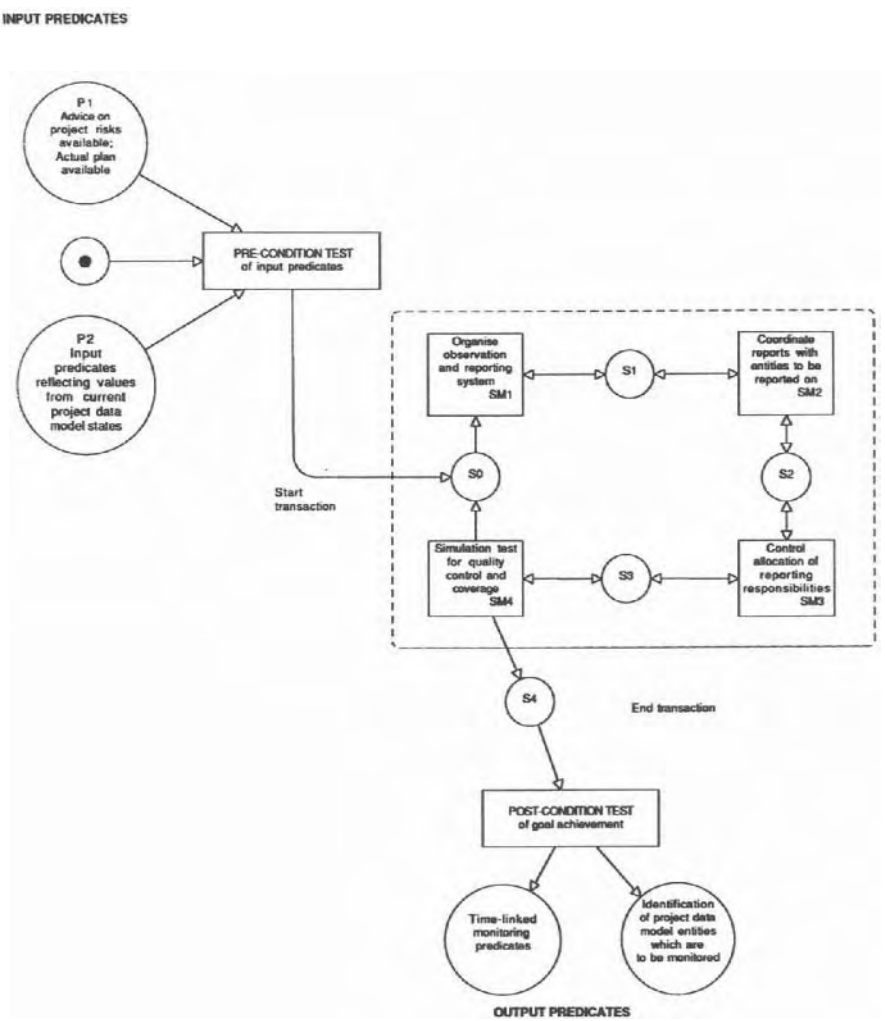
† We do not discuss here the perspective views on the project data model instantiation which precede and follow each operation, as *we.* did for the operations implementing the PMACs described m sections 8.1, *82* and 8.3, but the reader should be able to develop these for himself if needed

‡ In section 9.3, we will describe the details of the process by which this is achieved.

(i) quality control criteria, and,

(ii) coverage (at the appropriate time) of particular information delivered for comparison purposes for the *full* set of currently relevant objects in the project plan.

Figure 8.10: A set up monitoring procedures PMAC refined to show its internal structure



If the test fails, the observation and reporting system has to be re-organised (that is, subPMAC SM1 needs to be invoked again and the cycle repeated). If the test succeeds, there is still a post-condition test to be passed. It is still possible to fail to achieve the goal to ***maintain a relationship between plan and reality,*** if the project plan has not got sufficient comparison points in it (usually, milestones) to pick up on all the discrepancies with the real project work system which, if they occurred in reality, might cause trouble. For example, a project plan which contains a task which lasts for 50% of the project's time span, and is on the critical path, and has no intermediate milestones which can be used for monitoring would be likely to cause the post-condition test to fail (if it had not already been corrected in the pre-condition test by failing to match a quality control procedure) regardless of how well the monitoring system maps onto the project plan.

If the post-condition test fails, then predicates are generated indicating previous conditions which have to be met in a re-organised plan (which are not met in the present one). These predicates are then input predicates to the ***standard planning*** (SP) and ***actual planning*** (AP) PMACs next time they are activated by the project manager.

If the post-condition test succeeds, then two types of output predicates are generated. The first type of predicates identify the ***time-linked monitoring procedures to be*** employed, ***so*** that the arrival of reporting information can be checked and logged. These serve as input predicates to the ***identifylpredict discrepancies and exceptions*** PMAC. The second type of output predicates identify those entities in the current instantiation of the project data model which correspond to the objects in the project work system which may now be subject to time-linked monitoring in reality, so that the estimated or anticipated values of the relevant attributes of these entities (e.g., time, date, effort spent) can be reviewed and updated according to the monitoring results. These predicates may also serve as input predicates to subsequent activation of the ***standard planning*** PMAC, as the re-organisation of a project plan which currently contains ***monitored*** entities has additional implications over one which does not. That is, if the re-planning involves deletion of any instances of entities marked for monitoring in the current plan, the ***standard planning*** PMAC must ***be*** re-activated from subPMAC SP1 onwards as a consequence.

8.5. Developing the internal structure of the identifylpredict discrepancies and exceptions PMAC

Once the ***set*** up ***monitoring procedures*** PMAC has been used successfully, an observation and reporting structure has been set up and monitoring becomes possible. The latter may be achieved through invoking an ***identifylpredict discrepancies and exceptions*** PMAC the characteristics of which were described in a pre-formal way in section 7.2 as follows:

Identify/predict discrepancies and exceptions (IP)

Function: to identify and predict deviations from the plan given the results of monitoring and assess the severity of discrepancies.

Type: internal.

Typical input predicates: actual plan is activated@;entities to be observed and reported on are identified®; estimates of effort, cost and duration are available®; risk advice is available; monitoring procedures are established.

Typical output predicates: missing reports are identified; lists of confirmations and exceptions are available.

Principal operations carried out within PMAC: control timed reporting; coordinate planned and reported values for relevant entities in the project data model; control forward tracking as effects of identified discrepancies.

Perspectives employed: project function; activity; time and resource; personnel.

Goals addressed in post-condition tests: maintain relationship between plan and reality; facilitate project work; satisfy external stakeholders; create and implement a project plan.

The principal operations carried out within this PMAC define the sub-PMACs which constitute it. These are:

IP1 - control timed reporting;

1P2 - coordinate planned and reported values for relevant entities in the project data model;

IP3 - control forward tracking as effects of identified discrepancies.

Figure 8.11 illustrates the internal structure of an ***identify/predict discrepancies and exceptions*** PMAC, refined to show its three principal subPMACs linked within a place-transition net.

This structure provides, first, for the use of a subPMAC IP1 (employing the time and resource perspective) in controlling, observing and reporting at identified points within the project work system, as currently instantiated in the project data model. SubPMAC IP1 achieves this through allocating responsibilities in this respect to the appropriate resources, so that they observe and report at the required time. Successful completion of the activation of subPMAC IP1 enables (via place S1 in figure 8.11) the use of a subPMAC IP2 to coordinate planned and reported values for particular attributes of the relevant instantiated entities in the project data model in order to identify discrepancies†.

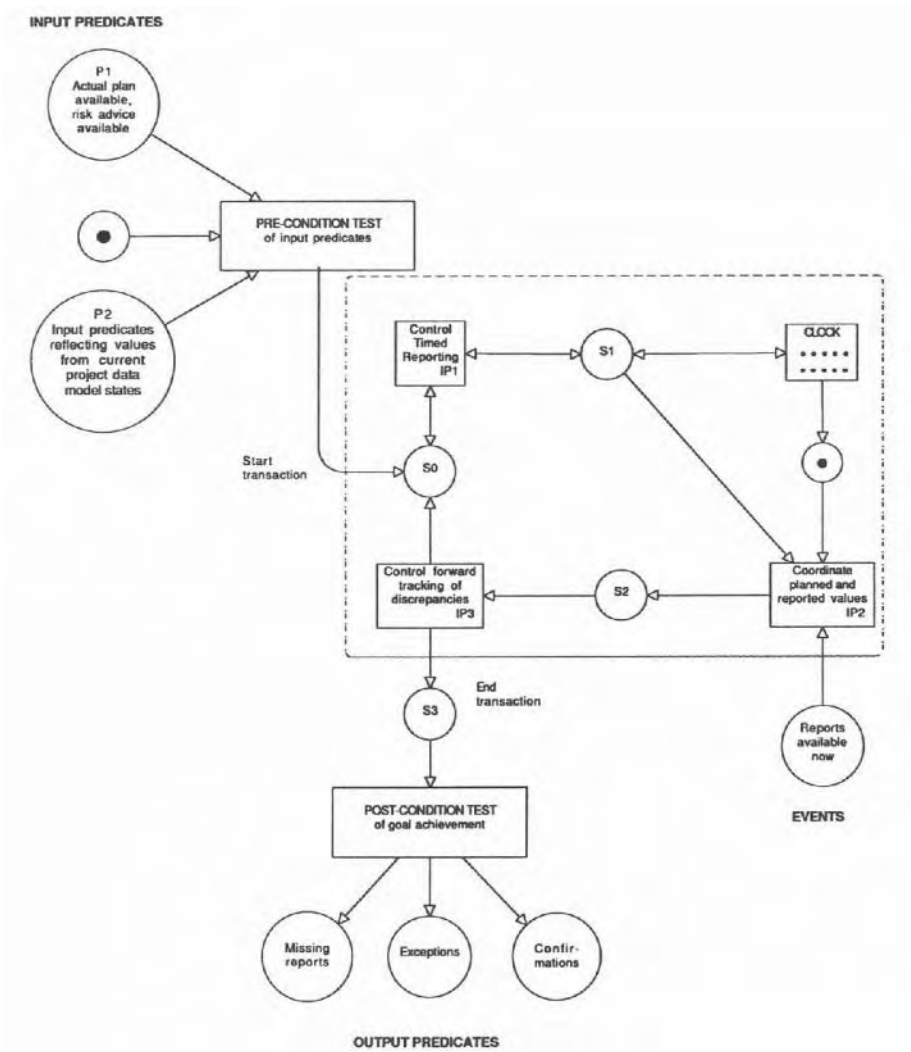
If the ***set*** up ***monitoring procedures*** PMAC has been executed properly, coordination of report items with instances of entities in the project data model, as they are received, should be quite straightforward. However, if PMAC SM was not previously used to ensure a good observation and reporting structure, coordination may involve a complex search through large parts of the total

† In section 9.4, we will discuss the details of the processes by which reports are analysed to produce confirmations and identify discrepancies.

current instantiation of the project data model, in order to coordinate instances of the relevant entities within it (when found during the search) with reported material. Another complex search of this type would also then be necessary in order to examine discrepancies on reported values. An ***identify/predict discrepancies and exceptions*** PMAC would, in this ***case,*** need to be generated with the **specification for** such searches given within **subPMAC** IP3.

However, no refinement of this kind is shown explicitly in the **structure** represented in figure 8.11 which presumes that activation of PMAC IP will rely on a precondition test of the prior existence of an **adequate** observation and reporting **structure.**

Figure 8.11: An identify/predict discrepancies and **exceptions** PMAC **refined to show** its internal **structure**



In figure 8.11, subPMAC IP2 takes as its inputs

(a) an initiating predicate from the clock, saying that it is time to process *reports* due at a particular date, and,

(b) the reports actually available at that time.

The confirmations and discrepancies identified in subPMAC IP2 are the subject of the forward tracking in subPMAC IP3 which investigates whether possible effects of an identified discrepancy indicate that it should be treated as an *exception* to what can be handled within the current actual plan for the project If subPMAC IP3 is unable to resolve this issue, the project manager may wish to move (via place SO) back to subPMAC IP1, where he can adjust his control of timed reporting in a way that will, hopefully, yield more definitive information on this issue in the next (clock-timed) set of reports which will be processed by subPMAC IP2.

If, on the other hand, the manager is satisfied with the way in which the forward tracking of discrepancies in subPMAC SP4 identified the exceptions (if any), he can then move (via place S3) to the post-condition test of goal achievement The output predicates generated through activating this post-condition test may address the following three types of conditions:

(1) Missing reports: Predicates in this set identify the points in the project work system from which information is required in order to review and update attributes of relevant entities in the current instantiation of the project data model, but which have not yet reported this information (i.e., the attribute values have not yet been updated). The values of these predicates may consequently guide the project manager in the re-activation of subPMAC IP1 to allocate "report-chasing-up".

(2) *Confirmations:* Predicates in this set identify key planned events (e.g., milestones) which have been achieved since the last time this PMAC was invoked. In the absence of such predicates, the only type of automatic confirmation reporting would be of the type "no news is good news", indicating that "everything is going according to plan". However, affirmative reporting (e.g., deliverables ready for collection) is often required for Liaison with external stakeholders.

(3) *Exceptions:* Predicates in this set, in cases where there is a mismatch between the plan-expected and the reported values, identify, in each case, the attributes of the relevant instantiated entities in the project data model the values of which indicate exceptions For each exception so identified, predicates in this set may usefully be set to indicate the point in the project work system which generated the exceptional information and the nature of the exception (usually, a difference in parameter values relating to a particular object in the project work system).

Resolving exceptions requires a revision of the actual plan for the project, and, thus, re-use of the actual *planning* PMAC, but it may be possible to resolve exceptions successfully without prior re-use of the *standard planning* PMAC. However, if the post-condition test in the *actual planning* PMAC indicates that resolution of exceptions is not successful, then, the

standard planning PMAC must be invoked since major re-planning has become inevitable. This strategy of using the ***standard planning*** PMAC only after the ***actual planning*** PMAC post-condition test has failed follows from adopting the objective "try to maintain the relationship between plan and reality with minimum disturbance due to re-planning"?.

As well as examining the achievement of the triggering goal for the IP PMAC (that is, to ***maintain the relationship between plan and reality),*** the post-condition test of goal achievement may indicate, for example, how the list of confirmations may promote the goal to ***satisfy external stakeholders. Or,*** if there are many missing reports, the post-condition test may indicate that higher priority should be given to the goal to ***facilitate project work,*** perhaps through subsequent activation of the ***handle team problems*** PMAC. Finally, if there are many exceptions, the post-condition test may indicate that attention should, once again, be. directed to the goal to ***create and implement a project plan,*** as the need for re-planning may be inevitable.

8.6. Linking of PMACs through predicates in instantiating the project management model

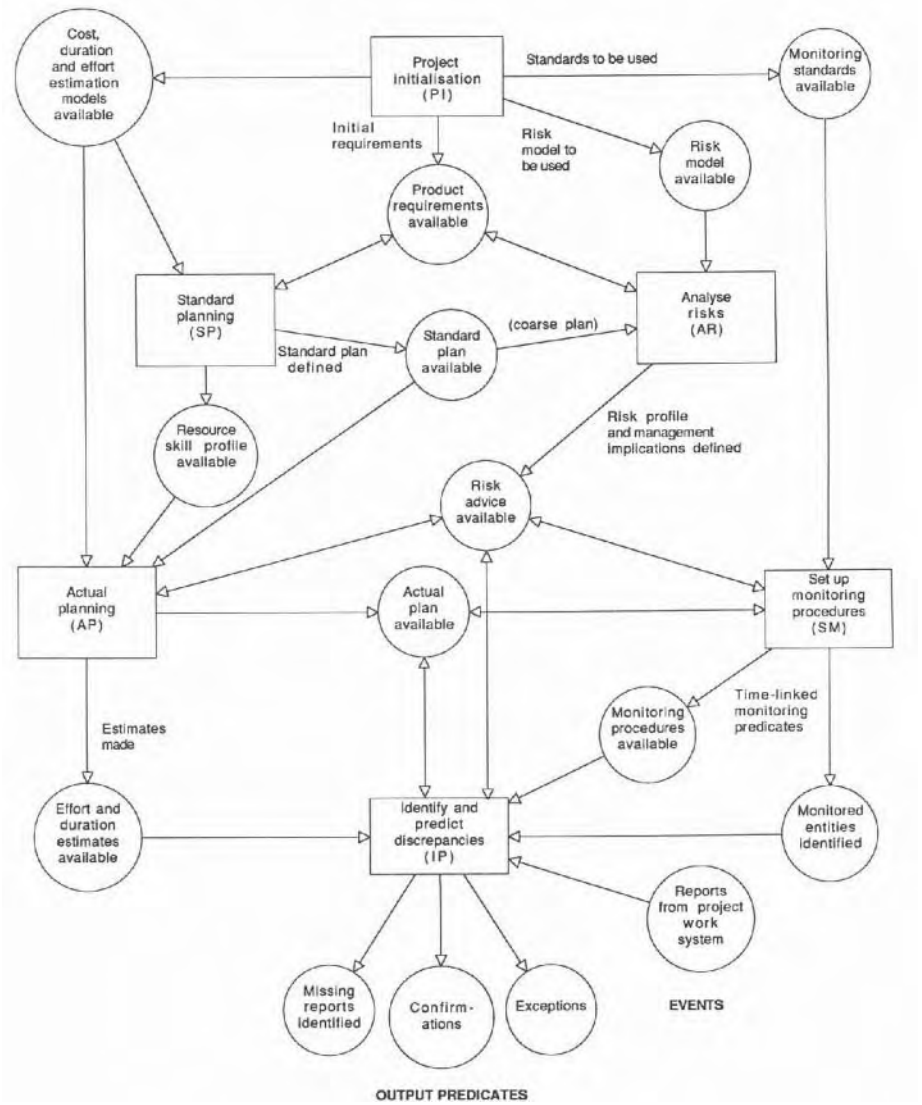
The discussion of the five PMACs considered in the previous sections of this chapter gave several examples where the output predicates from one PMAC served as input predicates to another. Figure 8.12 shows, in overview, some of these linkages between the output and input predicates of the particular PMACs. The same place-transition formalism that was used in sections 8.1 through 8.5 to express linkages between subPMACs is employed in this section to express linkages between the PMACs themselves. So, figure 8.12 simply summarises and arranges representations of parts of the project management model as generated and displayed in figures 8.1, 8.6, 8.8, 8.10 and 8.11 into a representation which might be viewed as the result of generating a ***composite instantiation*** of PMACs within the project management model. Within this representation, we have included also parts of the linkage with the ***project initialisation*** PMAC (PI) (which, however, has not been described in this chapter), as this' PMAC is the source of some of the key input predicates required by the pre-condition tests of most of the other PMACs shown in the picture.

For clarity of exposition, we have added the name of some of the predicates which are output and input along the linkages shown in figure 8.12, but this should not be taken to imply that these are the ***only*** predicates which are used to exchange information between the PMACs shown. Neither should it be taken to imply that the only use of the predicates named is to ***carry*** information along the links shown in this figure.

† **For a justification of this objective, see the discussion on "change control" in section 5.1.2.4.**

In summary, the representation shown in figure 8.12 should not be interpreted as either a global or a static model of the project management system. It is simply a wider (and coarser) view on part of an instantiation of the project management model *generated* in terms of a composite structure of PMACs.

Figure 8.12: A composite instantiation of PMACs



The output predicates shown at the bottom of figure 8.12 *(missing reports identified, confirmations,* and *exceptions), are* those which are of particular importance to the project manager for the purpose of project progress control. There are, of course, many other output predicates from the PMACs identified in figure 8.12 which are of interest to the manager. Here, though, we have explicitly identified just these three, and we have shown their position as *links* within the net illustrated in figure 8.12, in order to give an idea of the complexity of the *source network* of prior management activities which is required to ensure that the project manager's subsequent attempt to control the progress of his project may be efficient and effective, thus meeting his goal to *maintain* the *relationship between his plan and reality.*

In the next chapter, we examine some of this complexity on a local, rather than a general, level, concentrating on building a *local process model* for a particular subPMAC within each of two of the PMACs shown in figure 8.12 (that is, the *set* up *monitoring procedures* and the *identify/predict discrepancies and exceptions* PMACs). These subPMACs focus on the inferences the project manager has to make in (i) setting up and organising an observation and reporting system, and, (ii) coordinating the planned and reported values for relevant entities in the project data model. Success in these inferencing activities is of particular importance for the project manager's goal to maintain the relationship between his plan and reality.

Chapter 9

Making inferences about the project

Deciding on the degree of refinement of the internal structure of a PMAC is a rather arbitrary decision on the part of the modeller. It depends mainly on what he wants to attain through the modelling activity. In chapter 8, the granularity of the description of PMACs was rather coarse as this was sufficient for describing the generative and high level aspects of project management activities. In this chapter, we show how a further refinement of certain parts of the internal structure of a PMAC ties in more or less naturally with knowledge modelling. We will be concerned in particular with PMACs which relate to progress control activities of a project manager where detailed knowledge modelling is possible and rules which may reflect this knowledge are relatively easily obtained. These were described in chapter 7 as the ***set up monitoring procedures*** PMAC (SM) and the ***identify/predict discrepancies and exceptions*** PMAC (IP). However, in principle, this kind of refinement could be carried out for every PMAC that was described in a pre-formal way in chapter 7 and represents a continuation of the process of refinement in modelling that was described for certain PMACs (including SM and IP) in chapter 8.

The two subPMACs, constituent parts of PMACs SM and IP, that we refine further in this chapter are those identified in chapter 8 as ***organising observation and reporting system (SM1)*** and ***coordinating planned and reported values for relevant entities in the project data model (IP2).*** These were shown in context in figures 8.10 and 8.11, respectively. We will refine each of these subPMACs to the level at which they show how inferences can be made about the state of progress in the project.

9.1. Reference concepts for measuring progress

As we described in chapter 4, a project can be viewed in terms of the tasks that comprise it which aim at meeting one or more project objectives, and which are performed under certain resource constraints, especially money and time. From the point of view of monitoring the progress of the project and, hence, of

monitoring the progress of each task that comprises it, the ***outcome*** of a task is an important concept This can be constructed from:

- the ***amount*** and ***quality*** of the output(s) (e.g., products, deliverables) it produces; and,
- the ***actual amount*** of resources it consumes (e.g., hours worked, computer time used, total time spent).

The value of the outcome of a certain task varies depending on the productivity achieved on the task. As such, the outcome of a task is a ***variable*** the value of which can be ***predicted*** before a task starts, and which can be given its ***real*** value after the task has finished. Any combination of characteristics of the output produced with characteristics of resources consumed can be seen as an outcome of a task (e.g., the amount of work finished against the money spent, the quality of the work achieved against the time spent).

While a task is executed, it is executed under certain (environmental) conditions which are assumed to be known. As such, these (assumed) states of such environmental conditions are ***parameters***† as their value is assumed to be constant or to be the outcome of a function the shape of which is supposed to be constant (e.g., a learning curve associated with every human resource working on a task). In the latter case, the value of a parameter can change during a time interval, but its value is ***known*** at every moment in time (e.g., determined by its position on the learning curve). The value of a parameter is postulated to influence the productivity of a task and, through it, the outcome of the task. Thus, as long as the values of the parameters are known, reasonable estimations can be made about the outcome of a task. In other words, parameters set the scene for performing a task

It must be emphasised that a certain quantity can not be exclusively classified as a variable or a parameter at all possible instances. Both these terms are used to specify the ***role*** a quantity can play during the monitoring process rather than represent an inherent property of the quantity itself. For example, a quantity such as hardware performance is considered as a parameter (i.e., assumed to have a constant value) for the purposes of the SM1 subPMAC in organising the observation and reporting system. However, it becomes a variable (i.e., its value may not be constant) for the purposes of the IP2 subPMAC in coordinating the planned and reported values for relevant entities in the project data model.

---

† The notion of a parameter as used in this chapter is closely related to the notion of *parameter properties* of entities in the project data model which we will describe in chapter 10. The main difference being that, in this chapter, a parameter is interpreted as something that does not change at all or, if it changes, changes as predicted. By contrast, in chapter 10. parameter properties will be described as properties that cannot be changed by the actions of a manager. In general, the use of the concept of a parameter in this chapter is wider than that in chapter 10 because it also relates to entities that are not (yet) represented in the *generic core* of the project data model.

9.2. Types of knowledge involved in making inferences about progress

In general, two types of knowledge are needed for making inferences (or reasoning) about the progress of the project or a task. The first type is ***declarative*** knowledge which provides the basis for description of the objects in the domain and the relations between them This knowledge corresponds to the information represented as passive (state) information within the project management model or the project data model. We have already discussed some of the declarative knowledge related to the project management model in chapters 7 and 8, and we shall discuss the one related to the project data model in chapter 10, below. As such, we shall not consider this type of knowledge further in this chapter.

Declarative knowledge, however, does not say anything about ***how to use*** the objects and relations it represents to solve a problem or make inferences. This is the task of the second type of knowledge needed for making inferences: ***reasoning*** knowledge. Modelling this kind of knowledge is achieved through instantiating a ***local process model*** which comprises an ***inference structure*** component and a ***reasoning process control*** component. We call this a local process model as it describes the process by which this knowledge is employed locally, that is, within a particular subPMAC. In the following two sections we outline the general characteristics of its components. Then, in sections 9.3 and 9.4, we describe how these components may be modelled within the local contexts of the SM1 and IP2 subPMACs, respectively.

9.2.1. The inference structure component

The ***inference structure*** consists of a net linking ***Reasoning Activities*** (RACs) and ***Reasoning Entities*** (REs). ***Reasoning Activities (RACs)*** describe operators in a reasoning process about the project and are, thus, modelled as functions which effect transitions within subPMACs in the project management model. ***Reasoning Entities (REs),*** on the other hand, describe the operands in a reasoning process. These may be of either of two kinds:

(1) ***Predicate Reasoning Entities (PREs),*** which are represented by predicates in the project management model and which reference the requirements for, and results of, reasoning within the local process model. There are three types of PREs:

   (a) ***input*** PREs, which are needed as input information by a reasoning activity;

   (b) ***intermediary state*** PREs, which are both needed as input for a reasoning activity and produced by another reasoning activity; and,

   (c) ***output*** PREs, which carry output information resulting from the (whole) reasoning process performed in the local process model.

(2) ***Virtual Reasoning Entities (VREs),*** which refer to attributes of entities within the project data model. VREs do not exist as entities within the

project data model itself but their structure is ***created*** out of selected (relevant) **attributes** of one or more project data model entities and instantiated with the values of these attributes as and when required?. Thus, VREs are ***virtual entities as*** far ***as*** the project data model is concerned. As we described in section **6.4.1,** a **virtual** entity of this kind is like an image of an object in a mirror or on a projection screen: it does not exist independently of the "real" project data model entities from which it is built.

Via the PREs and VREs, predicates in the project management model and attributes of entities in the project data model play particular roles in the reasoning process. These roles are determined by the various reasoning activities (RACs) which employ these PREs and VREs. For example, a concept such as *"**too late**" **can*** act ***as*** a ***hypothesis,*** interpreted by the value carried by a **PRE** or as an ***observable,*** referenced by a VRE. A PRE such as "60% ***delay is too much***" can ***be*** used in RACs that establish the seriousness of a delay occurring in a task

In summary, RACs identify inferences that can be made, while PREs and VREs identify the entities which are needed for the inference to be made. PREs, but not VREs, may be produced as a result of the reasoning activity (RAC). The input and output PREs from the (whole) local **process** model are the links between a reasoning process and its **project** management context. That is, they may be produced by another previously carried out **subPMAC** or be needed by another **subPMAC.** Input PREs can comprise PMAC input predicates in set P2, as described in section 6.3.2: they can describe complex states of the project environment or the project work system as represented in the project **data** model.

9.2.2. The reasoning process control component

The ***inference structure*** of the local process model is defined by the connections between the RACs and can be interpreted as a general structural representation of the reasoning knowledge involved. However, the connections between the RACs established **through** intermediary state PREs only represent movements of tokens carrying predicates (knowledge elements) which **are** ***possible,*** under appropriate pre-conditions, in the local process model. As we will illustrate in section 9.3.2, the place-transition net generated through linking RACs and PREs usually exhibits a high degree of concurrency (in **theory,** at least).

This does not mean that, by constructing such nets, we **are** trying to model the project manager's reasoning activities as if they were performed mentally with a high degree of **concurrency‡.** Instead, we interpret the place-transition

† **We discuss how this is achieved in detail in section 9.3.1, below.**

‡ **In fact, human reasoning processes tend to be sequential and goal-directed, rather than concurrent (Johnson-Laird 1983, Beach & Mitchell 1987).**

net as offering dynamic simulation capabilities concerning the sequences of RAC activation that *could,* logically. be carried out in making inferences about the project within a local process model. Therefore, to complete the picture of the actual reasoning procedures involved during any particular activation of the local process model, we need to model also a ***reasoning process control*** component in a way that will describe in what sequence or under what conditions the RACs in the inference structure should be activated to reach a certain goal. In other words, this component should dictate the ***reasoning procedure*** to be employed.

From an external perspective, the objective for any reasoning process control component is analogous to that we described on a more general level for a project management methodology in chapters 5 and 6. The general aim for a management methodology is to place additional constraints on the activation of management activities in the interest of ensuring effective management practices, given the particular organisational and project requirements, context and objectives, and the immediate history of management activities. At a micro level, this aim applies to the constraints on the activation of RACs within a reasoning procedure with the aim of ensuring an effective reasoning strategy. Thus, the reasoning procedures which are actually instantiated in any particular application of an inference structure should, ideally, interpret a reasoning strategy which is defined as a ***micro-component of the management methodology*** which is currently specified for the project.

In sections 9.3 and 9.4, the concepts introduced above will be used for describing in detail two subPMACs (SM1, ***organise observation and reporting system,*** and IP2, ***coordinate planned and reported values for relevant entities in the project data model).*** In section 9.3, we also enhance, through examples, the concepts introduced in this section.

9.3. A refinement of the SM1 subPMAC

The aim of the SM1 subPMAC is to organise the observation and reporting system which will be used during the running of the project to collect information about the progress of the project. In the refinement of the SM1 subPMAC, we examine first, in section 9.3.1, its inference structure component and then, in section 9.3.2, its reasoning process control component. In section 9.3.3, we provide some examples of the reasoning knowledge that can be used by the RACs modelled within the SM1 inference structure.

9.3.1. The inference structure for the SM1 subPMAC

Figure 9.1 gives a refined, but still rather approximate, place-transition net representation of the kind of inference structure which may be generated for the SM1 subPMAC. In figure 9.1, the various RACs comprising the inference structure are shown linked through the intermediary state PREs. Links from

input PREs to output PREs and with VREs are also shown. The place-transition representation formalism is similar to that used in figure **8.12** in chapter **8.** That is, each place on the link connecting a pair of RACs is inscribed with the name of the intermediary state PRE which is both output predicate generated by the RAC which links ***to*** the place and input predicate consumed by the RAC which links ***from*** the place. Where two places are shown linking into one RAC, the output predicates of the RACs which link ***to*** each of these places comprise, collectively, the input predicates to the RAC which links ***from*** both these places. For example, PREs ***cruciality of outcomes*** and ***certainty about outcome values*** are both input predicates to RAC ***Transform-3.***

However, in **figure 9.1,** we have also employed an extension of the representation formalism that we have used in previous net diagrams. The purpose of this extension is to indicate the special use of VREs within the net. VREs are shown inscribed in places denoted by ellipses with thick borders, whereas PREs are shown inscribed in places denoted by ellipses with thin borders. The predicates which comprise PREs are carried on tokens within the net in the usual way. That is, an input predicate arriving at a particular RAC will be ***consumed*** by that RAC. An output predicate arriving at a place, from a particular **RAC,** will have been ***produced*** by that RAC. If the particular output predicate constitutes an ***intermediary state*** PRE, it will, of course, then become an input predicate to be consumed by the RAC which links from the same place. It follows that any intermediary state PRE will have only a local, ephemeral and transitory existence during a dynamic simulation within an instantiation of a local process model.

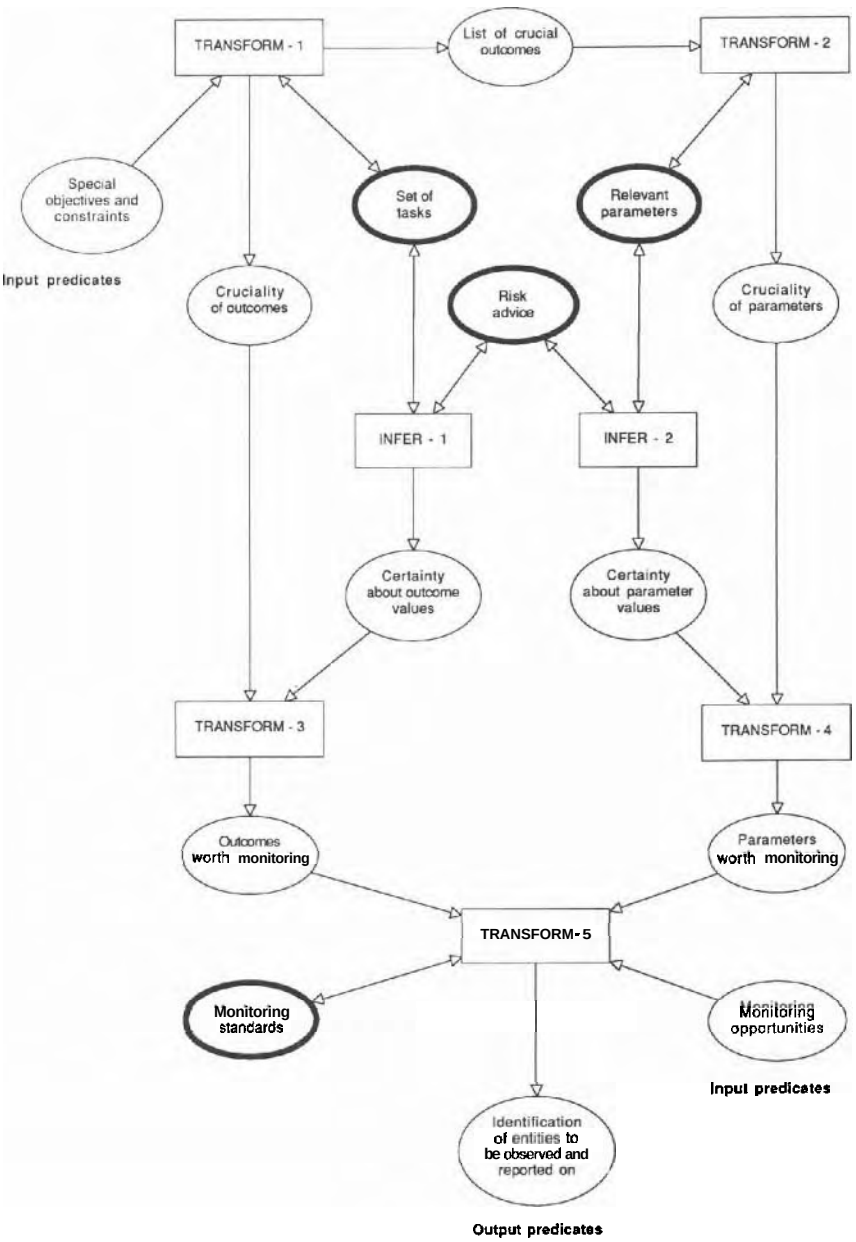
Any VRE will always be inscribed in a place connected by a *two-way* link (double arrow) to a RAC. The two-way link operates as follows: whenever the particular **VRE** is needed by the RAC, the RAC outputs a token along the link to the place where the VRE is inscribed. The values currently instantiated in the project data model for all the entity attributes currently "mirrored" in the VRE are copied into a complex predicate. This forms their image as the current instance of the VRE†. This **VRE** instance is carried back (on the token) along the same link to the RAC which requested the VRE.

Hence, any instance of a **VRE** is generated (as a complex predicate) and then immediately consumed, whenever required. The contents of the VRE are always up to date, as they reflect the current status of the project data model at the moment the VRE is instantiated. However, the VRE itself is "invisible" within the project data model: its creation and subsequent consumption has no effect whatsoever on any entity within that model.

---

† **In this way, attributes of entities described in the project data model can be used in the inferencing process by being composed into VREs. For this purpose, *virtual properties* may be constructed from basic properties of entities in the project data model. An example of such a virtual property is the notion of *delay* which is, in fact. the** result **of the comparison of the basic properties *planned end date* and *actual end dote* of any instance of the Task entity class.**

Figure 9.1: Inference structure for the SM1 subPMAC



This description of the creation and consumption of VREs *on* demand by a RAC may help to explain why more than one arrow may be shown linking outwards from a place inscribed with a VRE, within the place-transition net representation shown in figure 9.1. Normally, in a place-transition net diagram, *more than one* link out of a place would indicate indeterminacy, as there is no indication of which way a token leaving the place would go. This implies that there would then be no way of knowing, on the basis of the information in the net, *which* of the RACs linking *from* the place would be expected to consume the token and thus receive, as input, the predicates that it carries.

Wit n the formalism used to construct figure 9.1, this is indeed the case for any place carrying PREs (i.e., represented by an ellipse with a thin border). It is not the case, however, for places carrying VREs, because of the additional rule we have defined for each place of this type. That is, any token arriving on a place defined by an ellipse with a thick border must return along the same (bidirectional) link by which it arrived. Hence, there is no indeterminacy in a dynamic simulation of the instantiated net. This same rule also explains why there is no direct link between any pair of RACs linked *to* and *from* (but never *through)* the same place represented by an ellipse with a thick border. Therefore, RACs are always linked through PREs, never through VREs.

The following PREs and VREs provide input information to the inference structure of SM1 subPMAC (i.e., *organise observation and reporting system)* illustrated in figure 9.1.

Information from input PREs:

Special objectives and constraints which relate to indications about the importance of realising certain outcomes. For example, the output of a task may be stated to be very important, because it has to be shown to the client on Tuesday afternoon, or resources used may be stated to be an important quantity to monitor in the case that they are very scarce.

Monitoring opportunities relating to characteristics of the running project that can give information about the feasibility of monitoring certain quantities.

Information from VREs:

Set of tasks described in terms of their *expected outcomes* and the *&pendencies between tasks.* The expected outcomes are described in terms of resources to be used and outputs to produce, for every task in the plan currently instantiated in the project data model. The dependencies between the tasks are also subsumed in the project plan, and inferred from the workbreakdown structure or the schedule.

Relevant parameters relating to a list of parameters relevant for performing tasks included in the plan.

Risk advice pertaining to the specific risk areas in a project. This information may be instantiated in the project data model through risk analysis as discussed in chapter 3 and detailed in section 8.2.

Monitoring standards, that is, organisational standards for monitoring, providing information about which quantities have to be monitored and the recommended frequency of monitoring them.

The following RACs are part of the SM1 inference structure:

Transform-1, which yields a set of outcomes that are crucial for the project. According to the input PRE ***special objectives and constraints,*** certain outcomes of tasks are selected as crucial to the success of the project. Outcomes of tasks can occur either individually (i.e., as created by the tasks themselves) or by one of the parent nodes in the workbreakdown structure. For each outcome, the degree of cruciality is given. For example, "deliverable A of task X is extremely important", or "deliverable B of task Y can wait". The intermediary state PRE ***list of crucial outcomes*** contains only the names of the crucial outcomes without any information on the degree of their cruciality. The latter is included in the intermediary state PRE ***cruciality of outcomes*** which is an output predicate of *Transform-1*.

Transform-2, which yields a set of parameters with indications about their cruciality according to the set of crucial outcomes (intermediary state PRE ***cruciality of parameters).*** The cruciality of a parameter depends on the cruciality of those outcomes whose occurrence depends on the correct value of the parameter.

Infer-1, which yields the degree of certainty that a particular outcome value will occur (intermediary state PRE ***certainty about outcome values).*** This is accomplished by using the appropriate risk advice.

Infer-2, which yields certainty associated with the values of parameters, which are either already known or will be known in the near future (by definition or by prediction), and about the reliability of assumptions made about these values (intermediary state PRE ***certainty about parameter values).*** This RAC also uses information gained from the ***risk advice*** VRE.

Transform-3, which yields a list of outcomes worth monitoring. This RAC produces the intermediary state PRE ***outcomes worth monitoring*** (i.e., those considered crucial for determining the progress of the task or a set of tasks).

Transform-4, which yields a set of parameters which are worth monitoring (intermediary state PRE ***parameters worth monitoring).*** These parameters are relevant, but their values are not yet known with sufficient certainty at this point within the inference structure.

Transform-5, which yields information pertaining to the definition of monitoring procedures. The output PRE ***identification of entities to be observed and reported on*** consists of a set of observables†, related to the

† Observables are quantities that can be measured in the running project and stored on values of entities instantiated in the project data model.

outcomes and parameters to be monitored. Also, the points in time in which monitoring will take place, the level of detail and methods used to monitor are inferred in this RAC and are included in this output PRE.

9.3.2. Reasoning process control for SM1 subPMAC

The way the inference structure component is modelled within a local process model will place certain constraints on the reasoning procedures that may actually be followed by the project manager when activating that local process model within a subPMAC. On the basis of the information in the net shown in figure 9.1, the constraints on the reasoning procedures which may be employed within subPMAC SM1 can be described as follows:

In ***Transform-1*** and ***Tramform-2,*** the crucial outcomes and parameters are identified. ***Infer-1*** and ***Infer-2*** determine the certainties of outcome and parameter values, respectively. By means of ***Transform-3*** and ***Transform-4,*** the outcomes and parameters worth monitoring are established. Through the activation of ***Transform-5,*** the final monitoring procedures are produced.

However, these constraints, on their own, do not specify any strict sequence of activations of RACs (i.e., a reasoning procedure). As can be seen from inspection of the net in figure 9.1 (or, formally, through an analysis of the input and output predicates of the RACs in the inference structure, as expressed in the PREs) there is no fixed order of activation for the first four RACs. Moreover, dynamic simulation of the net is not possible, without additional constraints, as the RACs ***Infer-I*** and ***Infer-2*** cannot receive any token before having to produce one, so the ***initial marking*** of the net at the start of the dynamic simulation cannot be defined.

However, starting from the process modelling information represented in the net, it is possible to induce an order of activation for the RACs in a dynamic simulation by means of ***additional conditional statements*** in the development of the reasoning procedures. An example of an additional conditional statement might be: "If the number of crucial outcomes is expected to be small, then do ***Transform-1*** before ***Infer-I".*** The rationale for this condition could be that, whenever there are only a few crucial outcome values involved, it will be inefficient to establish the certainties of a fairly large set of outcome values of which the majority will be rejected. Apart from inducing an order, one may also wish to prescribe repetition, under particular conditions, in the activation of (groups of) RACs. An example of such a repetition is repeating ***Transform-2*** until a sufficiently small large set of crucial parameters remains.

Adding conditional and repetition statements of this kind implies incorporating new elements in the reasoning process control component beyond those expressed in the place-transition net. This is because the net expresses only the ***necessary*** constraints on any reasoning procedure according to the instantiated inference structure. In order to incorporate these new elements, we must first describe two extensions to our predicate-transition approach to project

management model development. These extensions are made because:

(a) the RACs need to be interpreted as ***functions,*** rather than sets of expressions, effecting transitions, and,

(b) a ***reasoning strategy interpreter*** needs to be specified.

In the following, we will consider briefly what would be involved in making these extensions.

In net modelling, two representation forms may be distinguished. The ***graph form,*** which is the form we have employed so far in this book, is normally applied for system description and informal explanation while the ***matrix form*** is applied for formal analysis (Jensen 1986). Classical predicate-transition nets (Genrich 1986) use ***expressions*** in both the graph and the matrix form. However, the "new version" ***coloured Petri nets,*** as proposed by Jensen (1986), use ***expressions*** in the graph form but ***functions*** (rather than expressions) in the matrix form (Albrecht, Jensen & Shapiro 1989)†.

Up till now, it has not been important for us to distinguish between ***predicate-transition net*** modelling and ***coloured Petri net*** modelling because, in the graph form, they are, to all intents and purposes, the same thing (at least, at the level of precision of modelling attempted in this book). However, interpreting local process models in terms of "new version" coloured Petri nets provides us with the opportunity of being able to describe a RAC in terms of a ***function call*** in which the PREs and VREs to be employed are expressed as ***arguments*** in the call statement‡.

The following example shows how each of the RACs in the local process model for the SM1 subPMAC (as illustrated in figure 9.1) could be preformally described in this way. In the example, we employ a kind of pseudo-code where the function called is indicated in bold type, with its argument list given in parentheses. In the argument list for each RAC function, the output PREs come first. Then (separated by a semi-colon) come the input PREs and, finally (separated by another semi-colon), the VREs.

Transform-1 (cruciality of outcomes, list of crucial outcomes; special objectives & constraints; set of tasks)

Transform3 (cruciality of parameters; list of crucial outcomes; relevant parameters)

---

† **The "old version" coloured Petri nets used functions in both the matrix and the graph form.**

‡ **The word "argument" is employed here by analogy with the naming conventions in certain programming languages, where arguments define parameters of a function or subroutine which are used for passing references to data and/or control information between the function or subroutine and the next higher level of control (i.e., the program which "calls" it). We have chosen to use the word "argument" to avoid the confusion with other meanings already assigned to the word "parameter" in our modelling approach. Note also that the notion of a "function call" is equivalent to the idea of a "procedure call" in certain programming languages.**

Infer-1 (certainty about outcome values; ; risk advice, set of tasks)

Infer-2 (certainty about parameter values; ; risk advice, relevant parameters)

Transform-3 (outcomes worth monitoring; certainty about outcome values, cruciality of outcomes;)

Transform-4 (parameters worth monitoring; certainty about parameter values, cruciality of parameters;)

Transform-5 (identification of entities to be observed and reported on; monitoring opportunities, outcomes worth monitoring, parameters worth monitoring; monitoring standards)

Our re-casting of RAC definitions in terms of function calls means that it is possible to specify ***when*** a particular RAC should be activated (i.e., when a particular RAC function call should be executed) within a higher level control structure†. Modelling this higher level control structure would involve, in effect, generating a ***reasoning strategy interpreter*** which can make selections between different components in the inference structure that may be available for incorporation at a particular point in a reasoning procedure (i.e., establishing the truth value of the "if' part of a condition specified within a reasoning strategy) or being able to establish the terminating condition for a loop within a reasoning procedure. Obviously, how this is modelled will be a matter of interpreting a ***prescribed reasoning strategy.*** However, the idea of inheriting a prescribed reasoning strategy as a micro-component of a management methodology is outside the usual experience of present-day software development project managers. Thus, while the modelling and evaluation of prescribed reasoning strategies is an interesting topic for research, it is outside the scope of this book

9.3.3. Examples of knowledge that can be used in performing SM1 RACs

Knowledge that can be used in performing SM1 RACs can be expressed in terms of rules held in complex passive (state) components of the project management model accessed by the RAC functions (described in section 9.3.2), themselves active (process) components of the project management model. Examples of rules which may typically be used are given below.

***Knowledge used in* Transform-1.**

The following rule can be used for determining the cruciality of an outcome:

If the output of a task is used in several other tasks

† The activation constraints modelled in the inference structure are still present, as they are a d through the input arguments, and form the basis for a pre-condition test for RAC function activation.

or the **task** is on the critical path

or the output is crucial from a functional point of view

or the resources needed are scarce and are also needed in future tasks

then the outcome of the task is crucial.

***Knowledge used in* Transform-2.**

The following rule can be used for determining the cruciality of a parameter:

If a parameter **is** important for **realising** a certain outcome of a task

and the outcome of the task is crucial

then the parameter is crucial?.

***Knowledge used in* Infer-1** and **Infer-2.**

The knowledge used in these two **RACs** is not easy to ascertain. If an explicit risk assessment has been carried out on the project, this could be used as the main source of information. Otherwise, the project manager will probably need to rely on intuition and rules of thumb, based on his own or colleagues' previous experience.

***Knowledge used in* Transform-3.**

In deciding on the selection of the outcomes that **are worth** monitoring, the following rule can be used :

If the certainty of an expected outcome value is low

or the cruciality of the outcome is high

then the outcome is worth monitoring.

The degree of certainty with which a particular outcome is expected to materialise is usually considered to be a more important factor in this decision than the cruciality of the **particular** outcome. For example, a less crucial task, carried out by **person(s)** whose capabilities are not known or **are** uncertain, will be monitored more closely than a **more** crucial task carried out by **person(s)** whose capabilities **are** trusted by the manager.

***Knowledge used in* Transform-4.**

In deciding on the selection of the parameters which are **worth** monitoring, the same rule is involved as the one used for the selection of outcomes worth monitoring. That is,

If the certainty of the value of a parameter is low

---

† Naturally, if a parameter (e.g., hardware) is crucial for several tasks, then its cruciality increases. Although the extent to which a parameter is important for realising outcomes differs from task to task, there are some general rules for dealing with such cases.

or the cruciality of a parameter is high

then the parameter is worth monitoring.

Similarly, the certainty about the value of a parameter is usually considered to be a more important factor for this decision than the cruciality of the corresponding parameter.

Knowledge used in Transform-5.

The knowledge which may usefully be employed in this RAC is not easy to establish because it is highly dependent on the project environment and the constraints it imposes on the manager's monitoring practice. In most cases, the particular contractor organisation or the client organisation prescribe the monitoring standards to use in the project. These standards normally prescribe

- monitoring the percentage of work finished;
- monitoring hours spent by man, by task;
- monitoring the quality of finished work.

In general, ***percentage of work finished*** is considered the most important quantity to monitor. However, the amount of work finished, in itself, cannot be measured directly. Instead, data from chunks of work that are finished may be collected and added up to constitute the overall percentage of work finished. Asking people about the amount of work that remains to be done is the usual means of ascertaining the amount of work that has finished.

Another quantity which can be easily monitored is ***money spent,*** measured as hours worked by the team members multiplied by their tariffs. When, for example, the percentage of work done meets a predicted level at a certain point in time, but more money is spent on the particular task than originally planned, this can mean trouble (**i.e.,** indicating that the project may go over its budget).

Although there are exceptions, managers monitor hours spent and percentage of work finished on a weekly basis (in larger projects, up to 20 years of effort, a two-weekly basis is preferred). The length of time intervals for monitoring also depends on the level of detail into which they have broken down the work of the project. The level of granularity of the plan and the level of detailedness with which it is described determine, for a large part, the possibility of following the progress of the project in a detailed manner. Thus, the project plan is a very important source of information for deciding about the time intervals of monitoring.

The points in time at which the project manager monitors parameters are dependent on the particular organisation within which he works. There are no general rules about how to decide on them but, in general, the way the manager checks the values of parameters does not seem very complicated. For example, one way of checking the productivity of a team member is by giving him small pieces of work (**e.g.,** a task estimated to take one day), and then monitoring the results of this person's work. Project managers also tend to use more general observables to gain an impression of the value of some parameters (**e.g.,** the stability of the client organisation).

It is a general project management dictum that it is very important to have ***explicit*** standards for establishing whether a task is finished or not. However, in order to be able to classify a piece of work as finished one also needs to determine whether the associated deliverable of the work has met the ***quality criteria*** that it was required to meet. This is a difficult problem as quality is not a ***single, measurable***, variable; it is composed of a number of observables which, collectively, could be used to indicate quality. The decision about which are these observables and which particular ones apply to which deliverable and at which phase of software development depends on the particular manager, the project requirements and the management methodology being followed, particularly in relation to quality standards.

Quality of a deliverable can be monitored in several ways. For example, through ***personal observations*** whereby the project manager looks at certain characteristics of the work produced depending on the nature of the work being considered. During the specification phase, for example, he may check for the transparency and completeness of the functional specifications produced; during implementation, he may check the format of the code, and so on. Sometimes, managers see quality as equivalent to the ***satisfaction of*** the ***client.*** When the project provides the client with results of tasks at regular intervals, the approval of the results by the client is taken to indicate that the results of the tasks were of sufficient quality. Other possibilities for monitoring quality include trusting the members of the team, or asking advice from other, external, experts (e.g., quality assurance team, as we discussed in section 4.2.4.1).

9.4. A refinement of the IP2 subPMAC

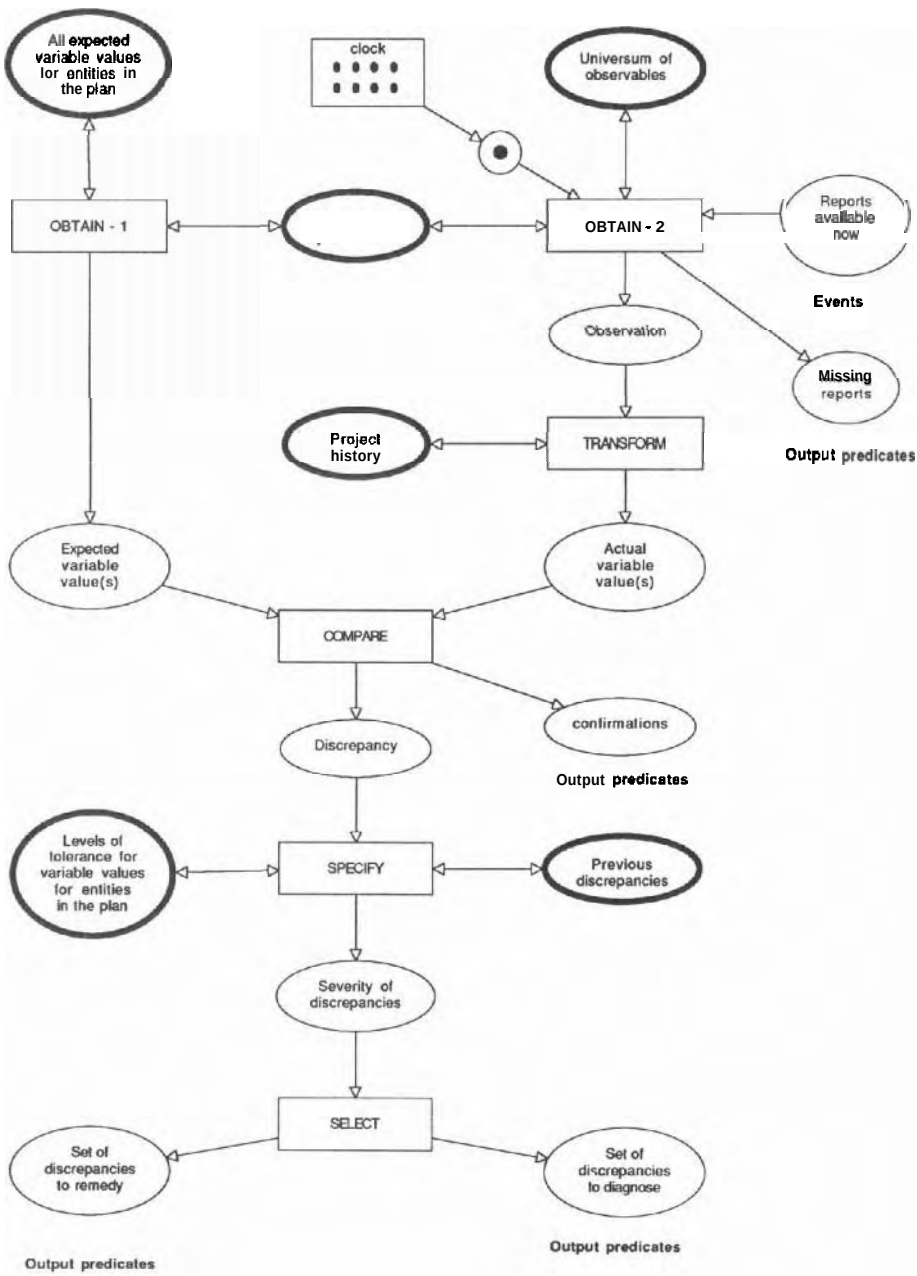
The aim of the IP2 subPMAC is to coordinate planned and reported values for relevant entities in the project data model, and, as such, its result is to determine actual progress of the project work. It can not commence unless which entities need to be monitored has been decided (i.e., SM1 needs to have been carried out) or the project work has not started. In section 9.4.1, we describe its inference structure and, in section 9.4.2, we describe the reasoning process control component for subPMAC IP2. In section 9.4.3, we provide some examples of the kind of knowledge that can be used in the RACs in the inference structure.

9.4.1. The inference structure for the IP2 subPMAC.

Figure 9.2 gives a refined, but still rather approximate, place-transition net representation of the kind of inference structure which may be generated for the IP2 subPMAC ***(coordinate planned and reported values for relevant entities in the project data model).***

The following PREs and VREs provide input information to the IP2 inference structure illustrated **in figure *9.2.***

Figure 9.2: Inference structure for the IP2 subPMAC



Information from input PREs:

Reports available now, the arrival of which is described in figure 9.2 in terms of ***events***†.

Information from VREs:

All expected variable values for entities in the plan, which refer to a list of values established by the project manager during planning. This list consists of ***assumed parameter values*** and ***expected outcomes.***

Universum of **observables,** which relates to the set of observable quantities in the running project. The universum of **observables** consists of the phenomena one can observe **empirically.**

Monitoring procedures, which have been set up as a result of activating the ***ser*** up ***monitoring procedures PMAC*** (as described in chapter 8).

Project history, which refers to data about how things in the project did go in the past.

Levels of tolerance, which provides indications about levels of tolerance for deviations of actual values from expected values, resulting from uncertainties in the plan, previous stages of the task, or other historical data.

Previous discrepancies remembered from previous stages of the project.

The following **RACs** are part of the IP2 inference structure:

Obtain-1, which yields ***expected*** values of variables at the point in time when monitoring takes place.

Obtain-2, which yields observation(s) from work being carried out within the project work system, reflecting ***actual*** values of variables at the point in time when monitoring takes place. It operates primarily on the information obtainable from currently available reports, and also identifies missing reports (those expected in the current reporting interval, according to the instantiated monitoring procedures, but for which the events signalling their arrival have not yet occurred)

Transform, which yields a transformation of the **observation(s)** into a value that can be compared with the expected value of the corresponding quantity produced by ***Obtain-I.*** For example, if the percentage of work finished is the selected quantity to use for monitoring, it is this percentage of work finished that must be measured when the task is monitored. Thus, the observations collected through ***Obtain-2*** must be transformed into a statement about the percentage of work finished.

Compare, which performs a comparison between the **expected** values and the actual values (observed **and/or** transformed) for a particular entity. The comparison results in determining the occurrence of a discrepancy or a confirmation.

† In chapter 10, we define events as changes in states which are not (directly) covered by the project manager's actions.

Specify, which determines the severity of a discrepancy. The degree of the severity of a discrepancy is used to indicate whether the inferred discrepancy has to be taken seriously. Important inputs to this RAC are those related to the levels of tolerance for variable values for entities in the plan and those of previous discrepancies in the project or in the particular task.

Select, which selects the discrepancies which should be remedied without further search for their causes, and those discrepancies which merit a more detailed consideration in order to establish their (probable) cause(s) (i.e., diagnose them).

9.4.2. Reasoning process control for the IP2 subPMAC

On the basis of the information in the net shown in figure 9.2, the constraints on the reasoning procedures which may be employed within subPMAC IP2 can be described as follows:

In ***Obtain-I,*** the expected variable values (instantiated as a result of prior activation of the ***actual planning*** PMAC) are collected and, in ***Obtain-2,*** data on the running of the project are collected and produced as observations. However, observations cannot always be directly mapped onto a variable as described in the project plan. Hence, ***Transform*** effects a ***transformation*** of the observations produced by ***Obtain-2*** into the form necessary for comparing its value with the anticipated value in the plan. ***Compare*** then compares the expected values with the actual values of the variables. As a result of the comparison between the predicted value of the criterion variable and the inferred one, discrepancies may arise or confirmations (that all goes as planned) may be found. ***Specify*** classifies the discrepancies according to whether they are worth paying particular attention or can be ignored (for the moment). ***Select,*** finally, distinguishes between those severe discrepancies which the manager needs to remedy immediately without looking further into their causes and those discrepancies the causes of which he needs to diagnose before doing something about them.

As can be seen from inspection of the net in figure 9.2 (or formally through an analysis of the input and output predicates of the RACs in the inference structure, as expressed in the PREs), ***Obtain-2*** is activated whenever (and only when) the clock "ticks", that is, it releases a boolean token indicating that the current reporting interval has elapsed. Nevertheless, dynamic simulation of the net is not possible, without additional constraints, as the RAC ***Obtain-1*** cannot receive any token before having to produce one, so the ***initial marking*** of the net at the start of the dynamic simulation cannot be defined.

However, starting from the process modelling information represented in the net, it is possible to induce an order of activation on ***Obtain-I*** and ***Obtain-2*** in a dynamic simulation of the IP2 local process model by means of ***additional conditional statements*** in the development of the reasoning procedures

described above. An example of a conditional statement that influences the order in which ***Obtain-1*** and ***Obtain-2*** are executed is: "If it is relatively unknown what to look for in advance, then perform ***Obtain-2*** first". The rationale behind this condition is that, in the absence of a "strong" model, the best thing to do is to gather as much data as possible. This addition to the reasoning process control component represents a strategy that can be labelled "data-driven monitoring". Similarly, repetition statements can be added. The most simple example of such a statement is "repeat IP2 until task(s) ready".

The following example shows how each of the RACs in the local process model for the IP2 subPMAC (as illustrated in figure 9.2) could be pre-formally described in the same way we described for the RACs in the SM1 local process model in section 9.3.2. As before, we employ a kind of pseudo-code where the function called is indicated in bold type, with its argument list given in parenthesis. In the argument list, for each RAC function, the output PREs come first. Then (separated by a semi-colon) come the input PREs and, finally (separated by another semi-colon), the VREs.

Obtain-1 (expected variable value(s); all expected variable values; monitoring procedures)

Obtain-2 (Observation; clock-tick [boolean], reports available now; universum of observables, monitoring procedures)

Transform (actual variable value(s); observation; project history)

Compare (discrepancy, confirmations; expected variable value(s), actual variable value(s);)

Specify (severity of discrepancies; discrepancy; levels of tolerance for variable values for entities in the project plan, previous discrepancies)

Select (set of discrepancies to remedy, set of discrepancies to diagnose; severity of discrepancy;)

The remarks about the need to generate a reasoning strategy interpreter to complete the modelling of the process control component which we made in section 9.3.2 in regard to the local process model for subPMAC SP1 hold here as well.

9.4.3. Examples of knowledge that can be used in performing IP2 RACs

Like the knowledge described in section 9.3.3 for the SM1 RACs, this knowledge can be expressed in terms of rules held in complex passive (state) components of the project management model accessed by the RAC functions described in section 9.4.2. Examples of rules which may be typically used are described below.

Knowledge used in Obtain-1.

If the project manager has made his expected values explicit in advance (not often the case), this operation is quite straightforward since values can be

derived directly from the project plan. The obtained value can be interpreted as the prediction of the *future* actual value of a quantity at a certain point in time.

Knowledge used in Obtain-2.

The value of an observable is selected from the running project, partly on the basis of the monitoring strategy chosen by the manager. Also, quite often, the project manager obtains observations which have not been planned to be obtained at all as, for example, when team members volunteer information about how things are going in the project.

***Knowledge used in* Transform.**

What happens here can either be (very) difficult or (very) easy depending on what has been observed. For example, transformation of observables to obtain a value for the quantity "money spent", is easy and more or less accurate. That is, when team members keep timetables of the hours they have spent on the project, the amount of money spent is simply calculated by multiplying the number of hours they have spent by their individual tariffs.

Transforming observables to gain a measure of percentage of work finished is more difficult because it is *derived* from two other types of data: the amount of work remaining and the total amount of work. In some cases, variable values may be the same as the observables while, in other cases, quite complex computations or combinations from the original observables may be necessary.

Transforming what is observed to a measure of quality is not easily done as most of the time subjective judgement is needed (as we discussed in section 9.3.3). This is especially true when monitoring is done at points in time other than at milestones, because quality criteria are less explicitly stated in such cases.

Knowledge used in Compare.

The way this comparison takes place depends on the type of quantity involved. Most of the time, this comparison is fairly trivial involving a simple subtraction in the case of numerical quantities. The yielded discrepancy can take the form of a number (e.g., the difference between the amount of money spent and the amount of money planned to have been spent), or can be in verbal or even graphical form. The *discrepancies* may be noticed after comparing the values of the variables or there may be no discrepancies found if everything runs smoothly. In other instances, one may find out that the value of a variable, as measured in the process, does not meet the value which was expected (determined in advance) by the manager. Often, managers prefer to keep track of work completed and work scheduled by means of a time-related graph whereby the optical distance between the two lines can serve as a discrepancy indicator.

***Knowledge used in* Specify.**

Two kinds of factors are relevant for making inferences in this RAC. First, the project manager wants to be certain that there really exists a discrepancy. Secondly, he wants to have enough time to repair *or* prevent serious damage in case of the existence of a discrepancy.

What is often observed in practice is that, during the first 30% of the time allocated to a certain task, a discrepancy is not immediately seen as severe; a learning effect is assumed to eventually smoothen this initial delay. Only very large discrepancies (e.g., spending twice as much time as expected) are regarded as serious at this point. However, when more then 30% of the work has been done, discrepancies of magnitude as low as 10% tend to be taken seriously.

Furthermore, the classification of discrepancies as more *or* less severe depends strongly on the history of the task and/or the resources involved. It makes a big difference if a discrepancy is noted between the amount of work planned and the amount of work completed, in the case where an ***improvement*** is involved when compared with the results of the previously executed monitoring activity. In terms of a graph this would mean that the lines representing the amount of work planned and the amount of work completed either are converging or are diverging even more. This can also be written in rule form as

if a task shows a constant *or* growing delay

then the discrepancy is (very) severe.

***Knowledge used in* Select.**

The objective of this RAC is to establish whether a certain discrepancy must be scrutinised further by means of a diagnostic activity and thus move onto another PMAC (i.e., diagnose and remedy exceptions), or whether corrective measures can be taken without any further enquiry into the cause of the discrepancy. The most characteristic example of a corrective action of this kind is the manager immediately adding manpower to a delayed task. Though this behaviour is often ill-advised (e.g., Brooks 1982), in practice, project managers tend to be under a great deal of time pressure ***to*** do ***something*** to find a solution to a problem rather than invest time in finding out what really caused it. In such cases, project managers often discover too late that the cost of not knowing has exceeded the cost of finding out.

However, on some occasions, investing time in diagnosing the causes of a discrepancy is advisable, and, indeed, very wise. This is particularly the case when there is a probability that the same delay will occur in a task which still has to start. This means that, if the manager suspects, expects or knows for certain that there will be a future task which is in some way similar to the current, delayed, one, he will be willing to diagnose the existing discrepancy for the current task since this will help him eventually to prevent delay in the future task. This is especially true in cases when human resources involved in the current task that has caused problems will be also working on the future, related, task.

9.5. Acting on the basis of inferences

Making inferences about the progress of the project is a means to an end for the manager; the end being to ensure that progress is being made and that it will continue to be made on the project. In fact, making inferences about the project progress can be seen as a continuous activity, instigated by the need to know how the project is doing.

Although, when developing the local process model for subPMAC IP2 in section 9.4, we discussed making inferences about project progress as a primarily time-dependent activity (i.e., performed at pre-specified repomng intervals, timed by a clock), this is only its formal aspect, necessitated by the need of the manager to receive reports regularly from his team in order to report to his superiors and plan what to do next. Its informal aspect is manifested much more frequently, however, as, for example, during group meetings and informal discussions with the members of his team. Here, the information on which inferences are based may be much more diverse than that available in the regular reports (which is why we refer to the "universum of observables" in figure 9.2). On such occasions, of course, the results of the inferences are more of the nature of a "gut feeling" about existing or forthcoming problems rather than a systematic analysis of discrepancies, their causes and how to handle them. These "gut feelings" often lead to actions which are not visible to other stakeholders (e.g., the manager's superiors). For example, if the manager senses uneasiness in the team about how to go about a task although team members are working on it and can effectively report progress (e.g., a certain number of lines of code produced), he may call a special meeting to address the issue explicitly even if no team member has actually articulated the existence of a problem with the task.

However, although "gut feelings" and the responses of the manager to them constitute the greatest part of the everyday reality of management, they cannot be adequately modelled (if at all) nor can one be trained in them or provided help with through some computer-based support system. This is why our discussion has been restrained to modelling the more formal and regular aspects of progress monitoring and control.

Making inferences about the progress of the project, whether formally or informally, leads to action on the basis of their results. Even deciding not to do anything about a discrepancy discovered or, if no discrepancy is found, saying that work should continue as it is planned represents an action on the part of the manager. Effectively, the inferences made and the actions to which they lead affect the project work system in addition to any changes that have already affected it as a result of the passage of time (e.g., amount of work having been carried out); for example, problems encountered with the progress of the project may be handled by the manager by re-organising the project work system. Similarly, making inferences and the actions to which they lead also affect the project environment as in the case when a resulting action involves transactions with it (e.g., when the manager negotiates with the client about increased availability of users to try out the system or part of the system being developed).

Thus, as a result of monitoring progress, the manager becomes aware of the current states of the project work system and project environment. As we discussed in chapter 6, these states are represented in the project data model which we shall describe in the next chapter.

Chapter 10

The project data model

As we described in earlier chapters, project management activities are carried out through operations on objects located in the project environment and in the project work system. Even project management activities which do not involve actual operations of this sort (e.g., a planning PMAC), still need to involve views of the relevant object system so that their results can be fruitful. However, as we have argued earlier, from the point of view of modelling the project management process (i.e., the concern of this book), any process aspects of these systems are of no concern here. For example, a manager is not concerned with the process by which a programmer will write a piece of code; his concern is that he does write it well, that is, he is concerned with the ***state*** of the task to which the programmer has been assigned (e.g., if it is completed or still in progress, the quality of its output). Similarly, the manager is not concerned with how his superiors will actually process the information which he provides to them; only that they have processed it and with the results of the process (e.g., they are content or unsatisfied with the results of the project).

With these issues in view, we have described the need to model those parts of these two relevant object systems (i.e., the project environment and the project work system) only to the extent that they are of concern to the manager (and, by extension, to this book). That is, we need to model their state rather than process parts. Figure 6.1 has indicated the role of the ***project*** data ***model*** in the project management process through containing information about the project environment and the project work system. In chapter 6, we argued that the project data model, as instantiated at any moment in time, is basically a ***static*** representation of the project environment and the project work system. because, here, the emphasis is on ***project management*** and not on a description of how the project work system or the project environment changes over time. The project data model must be able to represent the ***effect(s)*** of these changes, not the ***rules of change*** themselves. The modelling technique employed in defining and instantiating the project data model (entity-relationship modelling) reflects this difference in emphasis between modelling the project management system (for which it is not appropriate) and modelling the project environment

and the project work system (for which it is well suited).

Moreover, in building the project data model, we do not differentiate between entities within it which relate specifically to objects that are located in the project environment versus those objects that *are* located in the project work system. From the manager's point of view, the entities in the project data model simply refer to those objects which *are* relevant in carrying out his activities. However, we do not propose here that, in each new project context, the project data model user (i.e., the project manager) should generate the project data model *from scratch*. Rather, we provide, in this chapter, pre-structured specifications of the entities which can reasonably be expected to constitute the ***generic core*** of the project data model, regardless of the particular context in which it will be employed. The project manager can then expand, detail, or tailor the generic core entities to fit in with the particular project he is responsible for at the same time he tailors the activities and methods he will employ in managing it

In section 10.1, we describe the selection criteria we used to decide on which elements in the project work system and in the project environment should be taken as relevant entities constituting the generic core of the project data model. In section 10.2, we distinguish between different types of attributes of these entities. In section 10.3, we outline the generic core of the project data model and, in section 10.4, we provide an example of the relationship between a PMAC and the project data model (i.e., how a subPMAC uses the project data model).

10.1. Selection criteria for entities in the project data model

The main problem encountered in developing the generic core of the project data model is deciding ***what to include*** and ***what to leave out*** of the model. That is, deciding which entities constitute a generic core and which can be seen as not relevant to this generic core.

In a general sense, ***everything*** related to either the project environment or to the project work system is or could be relevant as an entity of the project data model. For example, a stock market crash can influence the position of the client organisation and, as a result, effect a change in the environmental conditions under which the project runs. By the same token, marital problems of a team member may affect the mood of that person and, hence, his productivity, thus causing a drop in the overall productivity of the project work system. One could think of a great number of such examples which could lead to an argument for the inclusion for particular entities in the generic core of the project data model (e.g., stock market state in the first example, personal problems of team members in the latter). This kind of strategy for model development could engender a generic core of the project data model which would be of quite unmanageable size, containing all the entities which could, in some foreseeable circumstance, become relevant. This is certainly not practical.

Thus, we have had to use some particular selection criteria in identifying which elements in the project environment and in the project work system can be used as entities to define the generic core of the model and which of their particular attributes can be used to characterise them and be defined as their properties. The initial way to construct this set is by identifying the union of all that can be seen in the perspectives employed by the manager to everything that could be relevant for the project. This has been called, elsewhere, the ***universe of discourse*** (International Standards Organisation 1982) on which data modelling is founded. It is, of course, impossible to enumerate everything that will be ***left out*** of this universe as a result of choosing a particular set of perspectives. Moreover, one should realise that managers ***do*** differ in the way they manage projects and, as a consequence, differ in the set of entities and properties they tend to use for representing the state of the project work system and the project environment.

Accordingly, the first criterion for selecting relevant entities we use is considering the elements in the object systems the construction and modification of which is under the ***discretion*** of the manager from the point of view of model building. Elements which lie within the discretion of a manager are represented by those attributes of entities whose values he is ***supposed to*** be ***able to*** or ***permitted to*** change and which do change during the lifetime of the project (Jaques 1976). For example, the quality of all human resources available within the project environment is mostly beyond the project manager's discretion: it is very difficult for him to change it and, if he can change it, it seems very unlikely that this change will affect his project (e.g., through training his resources on the project, he can affect the quality of these resources for his organisation as a whole but for the purposes of a next project). On the other hand, the quality of the human resources working in his project is within his discretion: he can change the quality by changing the particular personnel in the team (if possible) by claiming more skilled persons for his project and this change will immediately have an effect on the project as it would, by definition, increase the quality of the project's results. However, the use of the discretion criterion does not imply that there can be no changes in those elements of the project environment or the project work system which are not selected for inclusion in the generic core of the project data model. It only reflects that the ***manager*** is not able or allowed to effect these changes.

The second selection criterion we have used is that related to the ***constraints*** under which the manager has to operate in managing his project. These constraints normally fall outside the discretion of the manager to affect but he needs to know about them and carry out his activities within their range. For example, the manager can not alter the requirements set by the project environment on his own initiative but he needs to know about them so that he can plan his project and monitor its progress against them. Moreover, changes in such requirements occur very frequently during the lifetime of the project and can have significant consequences for the project. Thus, knowledge of the ***current*** requirements by the manager (as well as by his team members) is an imperative.

The two selection criteria we have just described can help us decide on the relevant entities that need to be selected to constitute the generic core of the project data model. However, there is still the problem of selecting the ***attributes*** of these elements which are relevant from the point of view of the project manager (rather than including ***all*** their potential attributes in the model). For example, a natural attribute of a human resource is whether the person is male or female (i.e., his or her sex). One can argue that the sex of a person can be a relevant attribute of the entity human resource in some cases (e.g., when the manager may fear its influence on team interpersonal problems). But making sex a defining attribute of the entity human resource in the generic core of the project data model is based on the assumption that such problems are the norm rather than the exception.

In deciding on which of their attributes should be selected to use to characterise the entities in the generic core of the project data model, we have used the notion of ***perspective*** that was introduced in section 5.3. Perspectives describe what can be "seen" (and, hence, included in the project data model) through taking viewpoints that focus on certain aspects of the reality of the project environment and the project work system. These perspectives can help both with identifying entities which may have been missed out through the use of the other two selection criteria mentioned above ***and*** with discovering the relevant ***attributes*** of entities. Relevant attributes are those attributes of an entity which are salient in one or more of the perspectives (i.e., function, activity, time and resource, and personnel perspectives) taken on the entity. Thus, the four perspectives we described can assist the modeller in describing those aspects of entities that are worth modelling. They serve as a kind of magnifying glass, enlarging salient attributes of identified entities.

However, changes in the state of entities in the project data model are not only a result of project management activities. They can be a result of the effect of the occurrence of ***exogenous events.*** Events ***are*** things which happen unexpectedly and effect state changes whose actual moments of occurrence are unanticipated (within the model) whether they have happened on their own or in conjunction with some action by the manager. ***Exogenous*** events are those events which are not covered by modelling the project management process. To that extent, they should be the subject of an extension of the project data model beyond the part predicated on states of it that are manipulated through PMAC operations. However, modelling "how" changes in states of the project environment and the project work system occur in cases where they are not emanating from the actions of the manager is not the purpose of this book. Here, we will simply indicate, for all attributes of entities considered, whether they will (probably) change due to project management activities or due to other factors emanating from the project environment or the project work system. This will, at least, indicate where changes can and will occur and leads to a meaningful discussion of the dynamic aspects of the project data model.

Finally, one could argue that the project manager himself is a part of the project work system just like his team members, and, as such, he himself should be considered as a human resource assigned to the project, and his

activities be described as tasks which consume the effort available from him as a resource. These tasks are, of course, also part of the project work system and should, as such, be represented as entities within the project data model (i.e., project management tasks being seen as special instances of tasks). In one sense, this can be handled by limiting the view the project manager has of himself at the level of the project data model to a resource to be assigned to identified tasks. In another, it could lead to a recursion which is difficult to model at the process level (i.e., where the project manager's activity is to model the process of performing the activities which constitute the process of managing the project). But such a modelling problem would only have to be faced in building a ***project process model*** and, thus, need not be considered in project data model development. Instead, here, we have not differentiated between project management tasks and other tasks, treating the manager as any other resource.

10.2. Attributes of entities

In chapter 6, a distinction was made between different types of attributes of entities: ***property*** attributes that describe intrinsic characteristics of the entity class, and attributes describing ***relationships*** which may exist between entities. For simplicity's sake, in this chapter, we refer to property attributes of entities as ***properties*** (and describe the properties of each entity in section 10.3.1) while we refer to relationship attributes of entities as ***relations*** (and describe them in section 10.3.2).

In describing changes in the states of the project data model, we will identify the likely ***variant*** (as distinct from ***parameter***) properties of entities in the project data model. ***Variant properties*** are those properties of instances of the entity class in the project data model that are expected to change over subsequent time slices. This variation can be caused either by direct action by the manager or by the fact that work has been going on in the project. ***Parameter properties,*** on the other hand, are those properties of instances of the entity class that are generally assumed not to change over different time slices†. However. changes may occur in practice and these are caused by exogenous events.

The difference between a variant property and a parameter property can be illustrated by an example. If a person is working on a certain task in the project it can be said that part of his time is "consumed" or, in other words, he is conmbuting effort to the task. This means that the value of the property "effort spent" for the particular task entity is changed. This change is expected and should, in fact, occur. If we assume another property called "availability of resources" in the entity class human ***resource,*** one can say that, at the

† Variation in the values assigned to these properties can, of course, occur over different instances in an entity class.

beginning of a certain task, the value of this property is planned and should stay the same during the period of working on the task (that is, working on the task does not change the value of the "availability" for the particular human resource entity). However, in practice, this value may change because, for example, as a result of a decision of someone higher in the hierarchy of the contractor organisation, resources may be shifted between projects while they are working without their shifting away from the particular project having being planned for by the manager. If such a thing happens, the value of the "availability of resource property" is changed in instances in the resource class affected by this higher level decision. However, this change results neither from work being carried out on the task, nor from an action of the manager, but from an exogenous event outside the discretion of the manager (i.e., someone else's decision). The distinction between variant properties and parameter properties in the project data model and the specification of the former, will provide the handles for the project management process to effect changes in the states of the project data model.

10.3. The generic core of the project data model

In creating the generic core of the project data model we proceeded in the following way:

(1) we identified the entities in the generic core of the project data model using the selection criteria described in the previous section;

(2) we identified the salient properties of these entities by applying the four perspectives we have described (i.e., function, activity, time and resource, personnel);

(3) we identified the key relations between the entities, that is, we determined the Sinks between the entities leading to the creation of a ***core structure*** in the project data model;

(4) we established, for each of the salient properties of the entities, and for each of the relations, whether it is meant to change over subsequent project states (i.e., to be a ***variant,*** rather than a ***parameter,*** property or relation);

(5) we established, for every variant property or relation, whether the change will be induced by the manager or by the work going on in the project; and

(6) we established, for each parameter property and relation, how sensitive it is likely to be to the occurrence of certain types of events that frequently happen during the lifetime of a project.

We describe the results of this process in the following two sections. Section 10.3.1 describes the core entities in the project data model and their property attributes. Section 10.3.2 then describes their relationship attributes.

10.3.1. The core entities in the project data model and their properties

In this section, we describe the entities which relate to objects in the project environment and the project work system which have been selected to constitute the core entities of the project data model. We first describe those entities which relate to the project environment (by using the selection criterion that they should represent ***constraints*** on the project). Then, we describe a particular entity which can be seen as an entity related to both the project environment and the project work system **(i.e., deliverable)** and, finally, we describe the entities which ***are*** constituted of objects related to the project work system.

Entities related to the project environment chosen for inclusion in the project data model as of particular importance to the manager are the **requirements,** the **contract, standards,** the **resource pool** and the **calendar.**

Entity: **Requirements**

Salient properties within the function perspective:

basic goals and objectives of the system to be developed

functional and performance capabilities of the system to be developed

Salient properties within the activity perspective:

how to measure and evaluate the performance of the system to be developed

Comments on properties: All properties of the requirements entity are parameter properties. The parameter properties most susceptible to changes due to ***events are*** the ***functional and performance capabilities of the system.*** It is well known that requirements from the project environment tend to change quite frequently during the lifetime of a project, simply because the client organisation evolves through time. These changes may be especially acute when a software development project has been commissioned, involving the induction of new computerised systems (running the software delivered from the project) within the client organisation. Thinking about computerised systems changes the way one views the organisation and this change generates new requirements for the system.

Entity: **Contract**

Salient properties within the function perspective:

client

Salient properties within the time and resource perspective:

work budget (estimated, spent)

staff budget (estimated, spent)

travel budget (estimated, spent)

other budget (estimated, spent)

start date (estimated, actual)

end date (estimated, actual)

Comments on properties: The ***client*** is a parameter property, as only details of the client (contacts, etc.) should change without invalidating the contract. All ***estimated*** properties are parameter properties while all ***spent*** properties ***are*** variant properties. The primary source of the variation is the work actually being carried out on the project. The same holds for the ***estimated*** and ***actual start date*** and ***end date*** properties. The parameter properties most likely to change are the ***estimated budget*** properties and the ***estimated end date*** property, which can get different values due to a variety of ***events*** in the client organisation. Though the budget is, in principle, fixed, in practice, a great deal of re-negotiation often occurs concerning the value of this property. In some cases, the project manager is the main negotiator for the contractor organisation and is, as such, one of the main agents effecting a change in these properties.

Entity: Standards

The standards to which the manager must adhere may originate in the client organisation or the contractor organisation. Standards can address different aspects of the project as, for example, documentation, monitoring, reporting, project management, quality, etc. As the scope and nature of these standards tend to differ greatly depending on the particular organisation, we do not attempt here to describe this entity in great detail.

Salient properties within the activity perspective:

topic(s)/areas subject to standards

ways to control and adhere to standards

Salient properties within the personnel perspective:

person(s) responsible for enforcing standards

Comments on properties: In general, all properties of standards are parameter properties: the very nature of a "standard" makes their change unlikely or undesirable. Nonetheless, some standards give the manager some leeway to tailor them to the actual situation encountered.

Entity: Resource pool

This entity is a troublesome one to model as it is not the "pool" itself that is contributed to the project but the elements in the pool who are actually allocated to it (i.e., actual resources). Were it the case that at the moment of the launching of the project all needed resources had been allocated to it, this entity could be left out from the project data model. Unfortunately, this is never the case. Negotiating over, and obtaining, resources from the contractor organisation's resource pool is one of the major tasks of a manager. This negotiation is carried out on the basis of information about what resources the contractor organisation has actually available (or could make available). For example, the manager will ask for someone in

particular who is available or could be available on the basis of knowing that this person has skills required in a specific area relevant to his project Thus, in a sense, the resource pool is a representation of what the manager "knows" about the resources in the contractor organisation.

Moreover, "resource" covers a number of different things that the project environment can contribute to the project It includes human and non-human resources (machines, office space), but *can* also include other kinds of resources such as models and techniques that *can* assist the manager in the work he has to do (e.g., for estimation, risk assessment). A computer-based project management support system can also be seen as a resource made available by the contractor organisation to the project manager. All these different types of resources are subclasses of the general resource entity class, which will be described below.

Salient properties within the time and resource perspective:

set of known resources or resource types

obtainability of known resource(s) types

Comments on properties: Both these properties are parameter properties. They *can* change (e.g., people may leave the contractor organisation) but this change is not effected through the action of the manager. The manager *can* effect this change only in connection with *another* project That is, claiming a particular resource for his project, the manager changes the obtainability of that particular resource for *another* manager and *his* project

Entity: Calendar.

This fairly simple entity is needed for the planning of the project work as it provides information on the working hours and days relevant for the project and acts as a time frame for the project

Salient properties within the time and resource perspective:

contractor organisation working days

client organisation working days

Comments on properties: Basically both these properties are parameter properties, outside the control of the project manager.

The entities described so far *can* be interpreted as a (partial) representation of the "inputs" from the environment to the project. During the execution of the project, there are also "outputs" from the project provided to the project environment. These outputs can be different types of things (for example, verbal and written reports, manuals, software) delivered to the contractor organisation or the client organisation. These outputs are represented by the entity deliverable.

Entity: Deliverable

Salient properties† within the function perspective:

destination of deliverable

type of deliverable

status of deliverable

Salient properties within the time and resource perspective:

projected dare of delivery of deliverable

actual delivery date of deliverable

Salient properties within the personnel perspective:

responsibility for deliverable

Variant properties due to work: ***actual delivery date, status of deliverable.***

Variant properties due to manager: ***responsibility for deliverable.***

Parameter properties: ***destination of deliverable, type of deliverable, projected dote of delivery.***

Comments on properties: The relatively large number of parameter properties reflects the fact that the deliverables of a project are subject to fairly hard constraints. Changes can occur when, for example, user requirements are adjusted, or an adjustment is made to the projected delivery date when there turns out to be slippage in the project and this slippage is irreversible and/or accepted.

The entities described above are those which are salient in modelling the "inputs" and "outputs" of a project The entities that we describe in the remaining of this section are those entities which are, for the larger part, related to the state of the project work system.

Entity: Resource

As has been mentioned already, this entity class contains a number of subclasses that must be described. The relation between the entity class and its entity subclasses follows the modelling conventions described in chapter 6. Thus, we will first present the properties of the general resource class, which are inherited by each subclass. This is followed by a description of properties specific to each resource entity subclass.

---

† Note that there is no *effort* property for deliverables because deliverables consist of products which are produced by tasks. In other words, the effort is represented through the task entity. Also note that. in general, most of the following entities described here lack many properties in the activity perspective. This is due to the fact that the activity perspective strongly emphasises *relationship attributes* between entities. These relationship attributes are described in section 10.3.2.

Salient properties within the function perspective:

name of resource

type of resource

suitability of resource

organisation resource belongs to

Salient properties within the time and resource perspective:

cost of the resource

amountleffort required from resource

amount/effort spent by/from resource

start of work resource

end of work resource

availability of resource

Salient properties within the activity perspective:

nature of work allocated to resource

Variant properties due to work: *effortlamount spent by/from resource.*

Variant properties due to manager: *effortlamount required from resource, start of work resource, end of work resource, nature of work allocated to resource.*

Parameter properties: *suitability of resource, cost of the resource, availability of resource, organisation resource belongs to, type of resource, name of resource.*

Comments on properties: The number of variant properties due to the manager indicates that, in this area, the manager has some control over what happens in the project. *Events* that can change parameter properties will occur most frequently for the property *availability of resource. A* closer look at the variant properties due to the manager shows, however, that, in principle, the properties *start of work* and *end of work* are under the control of the manager, but, in practice, this is the point at which exogenous events quite often overtake the manager. For example, when a resource is unexpectedly transferred to another project this changes the *availability of resource* property and, through it, the *end of work* property.

Entity: **Human resource**

This subordinate entity class has all the properties of the superordinate **resource** entity class described above *and* the additional specific properties described below.

Salient properties within the function perspective:

qualification of human resource

skills of the human resource

previous experience of human resource

Salient properties within the personnel perspective:

motivation of the human resource

capacity for teamwork of human resource

Variant properties due to work: *skills of human resource, previous experience of human resource.*

Variant properties due to manager: *motivation of resource.*

Parameter properties: *qualification of human resource, capacity for teamwork of human resource.*

Comments on properties: Though *qualification* and *capacity for teamwork are,* in principle, "givens" for the manager, quite often, the way the project develops tends to influence these properties. **For** example, the *capacity for teamwork* may decline as a result of unforeseen tensions between the members of the project team.

Entity: **Non-human resource**

This subordinate entity class can be mainly instantiated by **things** like machines, office space, etc. It inherits the properties of the **resource** class and has the following additional properties.

Salient properties within the function perspective:

characteristics of non-human resource

Salient properties within the time and resource perspective:

***reliability** of **non-human** resource*

Comments on properties: Both these additional properties are **parameter** properties. In practice, however, this is not always true. Quite often, a machine turns out to be slower than expected, or office space is less spacious than promised. Another problem that occurs frequently is a machine that goes "down" much more often than expected effectively reducing the time people can work with it.

Entity: **Task**

This entity refers to a more or less self-contained unit of work defined by the project manager. Most of the work in a project is subsumed under the concept of a task.

Salient properties within the function perspective:

location in system development life cycle

***criticality** from a functional point of view*

Salient properties within the time and resource perspective:

*required **non-human** resources*

required human skills

required human experience

required human resources
critical resources for the task
estimated effort
effort spent
remaining effort
estimated other costs
spent other costs
remaining other costs
required start date
planned start dare
actual start date
required end dare
planned end dare
actual end date
criticality in terms of time

Salient properties within the activity perspective:

work description
standards for carrying out work described

Salient properties within the personnel perspective:

responsibility for the task

Variant properties due to work: *effort spent, remaining effort, spent other costs, remaining other costs.*

Variant properties due to manager: *actual start date, actual end date, criticality from a functional point of view, planned start date, planned end date, responsibility for the task.*

Parameter properties: *location in systems development life cycle, required non-human resources, human skills, human experience, human resources, critical resources for the task, estimated effort, estimated other costs, required start date, required end date, criticality in term of time, work description, standards for carrying out the task.*

Comments on properties: Two kinds of *events* influence the parameter properties: (a) those events that are due to a wrong estimation of the task in terms of required resources and required effort and time needed, and, (b) those events that can happen to resources like illness, demotivation, etc.

Entity: Product

A product is something that is produced by a task and may be needed by another task. Some products are part of a deliverable.

Salient properties within the function perspective:

quality criteria for the product

product description

Salient properties within the time and resource perspective:

date of acceptance of product

date product is finished

Variant properties due to work: *date of acceptance of product, date product is finished.*

Variant properties due to manager: none.

Parameter properties: *quality criteria for product, product description.*

Comments on properties: Some properties of a product are parameter properties. A product seems reasonably secure from *events* other than delays that emerge from the work on the product. A fairly general practice related to products is changing the quality criteria used when the product is "finished" but does not yet satisfy the previously used quality criteria.

Entity: **Team**

The team is the group of human resources working on the project.

Salient properties within the function perspective:

composition of team

skills of team

organisational structure of team

productivity of team

Salient properties within the personnel perspective:

stability of team

balance of team

motivation of team

coheswn of team

Variant properties due to work: *cohesion of team, skills of team.*

Variant properties due to manager: *motivation of team, composition of team, organisational structure of ream, productivity of ream.*

Parameter properties: *stability of* team, *balance of team.*

Comments on properties: *Events* mainly influence the *stability of ream* (e.g., unexpected withdrawals of personnel) and the *balance of team* (e.g., disturbances due to frequent personnel changes).

10.3.2. Relationship attributes of entities in the project data model

The structure of the generic core of the project data model is defined by the relationship attributes which describe the relationships which may exist between instances in the core entity classes described in section 10.3.1. These relationship attributes could have been included in the description of the entities given there alongside their property attributes. However, for the sake of clarity, we deal with the relationship attributes separately here, indicating in each case the ***pair*** of entities which relate to each other. Thus, from the description of each relationship attribute, we can easily infer the description of the reciprocal relationship attribute. In each case, the relation is the same, only the ***direction*** of the relationship is reversed. So, this should highlight the structural aspect of the generic core of the project data model in a more explicit way. In order to keep terms as simple as possible, we refer to relationship attributes described in this way as ***relations.***

These relations indicate ***global cases,*** that is, those relations that may be present within the total membership of the sets of instances of entity classes in the project data model, but which are not necessarily present within any particular instantiation. Thus, they relate to instances in entity classes that ***can be*** modelled, rather than what is ***always*** modelled.

Each relation has a ***source entity*** and a ***destination entity***†. Furthermore, as described in section 6.4.2, a relation has a ***cardinality,*** that is, the number (one; more than one) of instances within the source and destination entity classes that can be linked by the relation. Some relations have a ***reciprocal,*** which means that if A is related to B, B will be related to A by the reciprocal of the relation between A and B.

The most important relations and their reciprocal relations are (the indications between brackets after the relations refer to the cardinality of the relation and the reciprocal) :

produce (be produced by) many (many)
be allocated to (use) many (many)
consist of (belong to) many (many)
need (be needed by) many (many)
work on (be worked on by) many (one)
constrain (be constrained by) many (many)
define (be defined by) many (many)
guide (be guided by) many (many)
limits (be limited by) many (one)

Through these relations (and their reciprocal relations), the entities of the project data model can be connected, as is shown below (source entities first,

† In the visual representation of relationship attributes employed in chapters 6 and 12, a relation is shown as a segment (arrow) directed *from* the source entity *to* the destination entity.

destination entities second) :

tasks *produce* products
products *are produced by* tasks
tasks *need* products
products *are needed by* tasks
resources may *be allocated* to tasks
tasks ***use*** resources
requirements *define* deliverables
deliverables *are defined* by requirements
deliverables *consist of* products
products *belong to* deliverables
the resource pool *constrains* the resources
the resources *are constrained* by the resource pool
standards ***define*** products
products *m e defined by* standards
standards *guide* tasks
tasks *are guided by* standards
the calendar *limits* the availability of resources
the availability of resources *is limited by* the calendar
tasks may *be worked on by* the team
the team *works on* tasks

The following relations, and their reciprocals, are instantiated by the project manager:

produce and *produced by*
need and *needed by*
consist of and *belong* ***to***
work on and *be worked on by*

The *be allocated to* relation is also instantiated by the manager but not its reciprocal. These are the *variant* relations in the sense that they can be changed by the manager, for example, by executing certain PMACs. The other relations can be considered as *parameter* relations as being outside the manager's discretion. They can change, however, due to events.

A relationship attribute does not necessarily have to define a relation between instances in *different* entity classes. There are occasions where it is useful to define relations between instances in the same entity class, although it is desirable to avoid defining relations of instances with themselves (the latter should be defined as properties). In particular, the relation *have as father* (with cardinality many to one) and its reciprocal, *have* ***as*** *sons* (with cardinality one to many), enable the instantiation of a hierarchy within a class.

Within the generic core of the project data model, there are two entity classes where it is definitely useful to instantiate hierarchies. These comprise tasks and **products,** respectively. A number of tasks can (collectively) *have as father* another (single) task, and a particular task can *have as sons* a number of other tasks. A **task** instance which has sons but no fathers is known as a "root task" and defines the top node in a task hierarchy?. Conversely, a task instance which has a father by no sons is known as a "leaf **task".** The same applies for **product** instances in a product hierarchy.

Care must be taken, when instantiating a hierarchy through linking task (or **product)** instances through have *as father* and *have as sons* relationship attributes, that no instance is ever linked to itself through a series of *have as father* or *have as sons* relations, regardless of whether other instances are involved as intermediaries in the linkage. Otherwise, what will have been defined is some kind of inconsistent network of tasks (or products) rather than a true hierarchy.

10.4. An illustration of a PMAC operating on tbe project data model

In chapter 6, we described how an instantiation of part of the project data model could be built by creating instances of entities and assigning values to their attributes. In the following, we try to show through the use of an example how, by means of operations on the project data model performed by sub-PMACs, these instances are created and modified. Our example is over-simplified and incomplete in that we have restricted ourselves to a discussion of the instantiation of entities defined in the generic core of the project data model, and the setting of some of their salient attributes. In any practical application, the project data model would have been tailored previously to introduce additional entity classes and attributes relevant to that application, and a full account would discuss how those were instantiated and set as well. However, the example will probably suffice for our intention to illustrate the types of operations involved in a typical sequence of subPMAC activations, rather than to specify anything prescriptive.

In the example, we will consider the four subPMACs which constitute the internal structure of the *standard planning* PMAC, as modelled in section 8.1 (and illustrated in figure 8.1). We will concentrate here on the operations performed on the project data model within each subPMAC in turn during an activation of this PMAC.

We start by assuming that the pre-condition test of input predicates has indicated that there are no instances of task already present in the project data model*. Hence, subPMAC SP1 (*organise work definition into a task*

† There can be more than one hierarchy instantiated in the task entity class, but each hierarchy has only one root task, which can *serve* m reference the whole hierarchy.

‡ Note that this pre-condition test would be likely to produce a different result if the *actual planning* PMAC had been previously activated, as in the case of re-planning.

hierarchy) is activated first: the operations it performs on project data model entities produce a task hierarchy as illustrated in figure **8.2.** This means that a set of task instances that constitute the task hierarchy are created. For each of these instances, its ***have*** as ***father*** and ***have us sons*** attributes are set to reference the task instances which are their destination entities, hence fixing the location of the instance in the task hierarchy. The ***work description*** property attribute of each of these task instances is assigned a description (set as its "value") of the work to be done within the particular task, and its ***estimated effort*** property is set to the preliminary value estimated at this time. The property attributes ***location in system development life cycle, criticality from a functional point of view, standards for carrying out work described, needed human resources*** and ***needed human experience*** may also be set.

Instances of the product entity class will be created (if they do not already exist), each with its ***product description*** attribute set to describe a product which may be produced or needed by a particular task or tasks. In order to indicate the precise relations between the various task and product instances, the ***produce*** and ***need*** relationship attributes of the task instances are set to reference the relevant product instances and, reciprocally at the same time, the be ***produced*** by and ***be needed*** by relationship attributes of the product instances are set to reference the relevant task instances.

Next, **subPMAC SP2** ***(coordinate products between tasks)*** is started. This takes as its input the task hierarchy that was instantiated through activating **subPMAC** SP1, and produces a network structure between the task instances in the hierarchy, as illustrated in figure **8.3.** This is done by linking each task instance which produces a particular product instance with any task which needs that product instance. This is done for all the leaf tasks in the hierarchy and the resulting inter-task l i i g e is displayed for perusal by the project manager who may wish to estimate some task durations at this point, particularly for those task instances which currently appear to have high ***criticality from a functional point*** of ***view*** and ***criticality in terms of time*** (the latter being another property which may be assessed and set at this point). This results in the setting of ***required start date*** and ***required end date*** attributes for particular task instances.

Also, at this time, additional ***be produced*** by relations may be set for product instances which are ***needed*** by one or more tasks but which are not, as yet, ***produced*** by any task instance. It may be that the relevant task instance already exists, in which case the additional relations with it can be instantiated immediately. However, it may the case that, in building the task hierarchy through the previous activation of **subPMAC SP1,** the project manager was unaware of the need to include the relevant task. If this is so, it may be necessary to suspend the activation of **subPMAC SP2** and re-activate **SP1** to amend the task hierarchy appropriately.

On successful completion of the activation of **subPMAC** SP2, the next step is to employ **subPMAC** SP3 ***(control allocation of standard resources)*** to estimate the resources required for each task to get the work done. This means that the time and resource perspective is taken on the project data model as

shown in figure 8.4, and that all ***required resources*** and ***estimated*** properties of instances of the **task** entity in the task hierarchy are now set. Some previously set estimates, particularly of effort, may be revised in the case that these are not independent of task duration and the nature of the standard resources provisionally allocated.

Note that ***required resources*** is defined ***as*** a property, rather than a relationship, attribute as it refers to hypothetical "standard resources which would be needed to achieve the work described for the task, and not to actual resource entities which may be available to the project. Subsequently, during the activation of the ***actual planning*** PMAC, subPMAC ***AP2 (control allocation of defined individual resources across time)*** will need to be employed to instantiate the ***use*** relationship between each individual **task** instance and the **resource** instances actually assigned (at that time) to that task†. During standard planning, however, the requirements for this instantiation of ***use*** and the reciprocal be ***allocated to*** relationships ***are*** established, but not the instantiation itself.

On successful completion of the activation of subPMAC SP3, subPMAC SP4 ***(rest adequacy of standard planning),*** as shown in figure 8.5, may be activated to check whether all various properties of the instantiated **task** and **product** entities that are relevant from the activated perspectives are set appropriately in the manner we described in section 8.1. In making tests, **task** and **product** instances are retrieved and the current values of their relation and property values are examined. However, subPMAC SP4 focuses on making tests, rather than remedying test failures (the latter is achieved through directing the re-activation of other subPMACs). Hence, it does not create new instances or reset existing attribute values of instances in these entity classes.

If the tests made in subPMAC SP4 are passed successfully, the ensuing post-condition test of goal achievement concerns whether this initial plan, from a high level functional point of view, is viable at all. If the test is positive, output predicates will be produced which may allow other PMACs which use them as pre-conditions to be activated. For example, such a PMAC might be the ***actual planning*** PMAC, which will do the planning given the available resources (after these have been negotiated through activating the ***negotiate resources*** PMAC). As is indicated in the description of the ***actual planning*** PMAC, this will lead to a time and resource perspective being taken on the project data model and result in the instantiation of the be ***allocated to*** relation between **task** instances and the **resource** instances.

† SubPMAC AP2 was described in section 8.3, and its location within the internal structure of the actual *planning* PMAC was illustrated in figure 8.8.

Chapter 11

Supporting the project management process

The discussion in chapters 2 through 5 addressed what is involved in the project management process as it is carried out by a project manager without any reference to tools and techniques which managers use in their work. Chapters 6 through 10 discussed what is involved in modelling this process and described how this may be achieved and they presented the results of the modelling we have attempted. As we have argued in section 6.1, project management modelling is a goal-constrained activity, the orientation of which is determined by the intention of the modeller concerning what to use the results of this activity for. Our explicitly stated intention was to provide both a better understanding of the management process through this modelling enterprise and a starting basis for the design of project management support systems (PMSSs).

However, before one even starts to think about designing such a PMSS on the basis of the models we presented so far, one has to *consider* what *is entailed in supporting a* project manager in carrying out his responsibilities, and how this may be achieved in practice through the operation of a system which can work hand in hand with him. This is the focal point of this chapter. In discussing relevant issues, we draw upon our exposition so far in the previous chapters and we utilise information we have collected through interviews with experienced project managers.

11.1. Providing computer-based support to project management

Providing support in a process or an activity is open to different interpretations by different persons involved in developing systems purporting to providing such support. At one extreme, expert systems may view support as providing to their users some knowledge encapsulated within the system. At the other extreme, techniques and tools may provide, to the user, simply a structure for entering his data and some algorithms to carry out any necessary calculations automatically. Somewhere in between these two extremes, interactive techniques may provide support through helping the user structure his particular

problem by providing guidance about how to go about it, and any necessary knowledge he may need on the way.

In all cases, support is offered to the user, but it refers to different types of support. The question that always needs to be answered is whether the type of support that is offered is the one called for by the nature of the particular activity and actually required by the particular type of user. Often, systems built on the basis of the *designer's* notion of what support is needed which does not correspond to the *user's* notion of required support tend to be under-used (if used at all) negating the reason for their existence (Paprika & Kiss 1985). A common design error here is to assume that the user (in this case, the project manager) may be *aided,* rather than replaced, merely through capturing relevant knowledge, formalising it in a "project management support system" which then attempts to apply this knowledge in a similar way to how the unaided project manager would (Humphreys 1989a).

Essentially, discovering the answer to the question of what kind of support is required in the particular case and by the particular type of user should begin with an investigation of the project manager's view of the process that the system is aimed to support, and of his roles in that process. Chapters 2 through 5 in this book addressed issues of precisely this type. However, this is only a *starting point* in developing user requirements, a means to an end. The next stage involves modelling this process in such a way that indications about the global design of the system can mirror the way the user views the project management process, and how he would like the activities he performs in this process to be supported. Only then can we concentrate on discovering what information the user needs from the support system at each stage, how he may be provided with it, and so on. This was the brief that chapters 6 through 10 attempted to meet.

Building models of a process and implementing them in a system design faces a host of potential problems if unjustified assumptions are made, or if relevant issues are left untouched during the knowledge elicitation and modelling stages in the development process: that is, when the designer *assumes* he *knows* how to proceed, from the point he stopped enquiring about the process onwards. Computer-based systems incorporating simplified or unrealistic assumptions can imprison managers in a constant conflict of their own perceptions and needs with those demanded by the system (Argyris 1977).

For example, consider the case of a modeller who wishes to elicit knowledge about the process of creating a workbreakdown (i.e., organising tasks into a task hierarchy) for a software development project. He may stop interviewing the manager who has the requisite knowledge once he discovers that the process involves (in the modeller's understanding of it) decomposing tasks into smaller tasks where the father tasks are phases in the software development process (e.g., specifications, design, coding, testing). Then, the modeller may assume that a tool that can help the manager should provide him with a pre-structured task hierarchy with the development phases as *fixed* top entries in the hierarchy. The result may be the provision of a structure which, from the user's point of view, is too inflexible and does not correspond to his

usual practice. Thus, jumping to conclusions about what kind of support is needed rather than discovering what the user would actually find useful, often results in unsatisfactory support tools.

By the same token, one could argue that the project managers who are the potential users may themselves not know what kind of support techniques they could profit from or what are their actual needs and problems in their work that can be met by a support technique (Mumford 1983b). Naturally, project managers' assertions about the kind of support they would appreciate in their work are likely to be heavily influenced by what they already know to be available in the market and their evaluation of any such support tools they have experienced.

In our investigation of characteristics of tools that project managers claimed they would welcome for use in their practice, we discovered the following typical requirements:

- the tool should be able to carry out simulations,
- it should keep historical data,
- it should respond fast,
- it should be controlled by the project manager (rather than have a mind of its own),
- it should show likely problems,
- it should deal with task interdependencies,
- it should be not expensive.
- it should carry out critical path analysis,
- it should be a planning aid,
- it should be flexible,
- it should not use too much time to enter data,
- it should have possibilities for different kinds of reports.

As can be seen from these desired characteristics of project management tools, desired elements of them range from concrete characteristics (such as expense and speed) to abstract characteristics (such as planning aid, show likely problems). The former types of characteristics are barely related to support while the latter are almost entirely support-related.

Still, this list of desired support could be extended by taking into account not only these elements these managers have discussed, but also extrapolating such elements from our previous discussions on the process of project management and the problems that one encounters while carrying it out. Focusing on the process, as well as its problems, can lead the designer to think about the ways in which his proposed PMSS can provide support.

As the earlier chapters of this book have amply documented, project management is a complex process where most of the problems encountered are often ill-structured. In such situations, built on traditional rule-based lines, expert systems are of little use. They can provide support for some aspects of this process where, for example, rules and knowledge can be relatively easily

determined (e.g., resource allocation, problem diagnosis, scheduling; Blanning 1984). However, they have the drawback that they provide their user with little (if any) flexibility as they "clone" knowledge (Ford 1985). Flexibility, though, is often voiced by project managers as a key requirement for computer-based support (e.g., Wynne 1983).

On the other hand, decision support systems (DSSs) seem to fit well with the demands of unstructured situations (Vari & Vecsenyi 1984). Their underlying ***adaptive design methodology*** (Keen & Gambino 1983) stresses the need to find out, quickly, what is important to the user in such situations. This allows the user to confront his problems, direct and control the system to a great extent (Ford 1985).

However, in practice, the majority of DSSs seem to be employed in essentially structured situations (Landry, Pascot & Briolat 1985). This arises from a tendency found in some such developments to look at the problems they attempt to support as ***objective realities.*** That is, it is assumed that there are ***facts*** that can be discovered concerning the particular problem to be handled rather than that the problem is ***perceived*** by its owner in a special way (i.e., it is a ***mental construction).*** As a result, support systems tend to be designed according to the designer's perception of the problem rather than the problem owner's perception of it. Thus, they are, by definition, used in structured situations where the possibility of an acceptable match between the designer's view of the problem and the problem owner's view of the problem is relatively high (e.g., concerning routine problems such as report writing).

Stabell (1983) discusses that the introduction of DSSs in management was preceded by an implicit assumption that automation was the long-term goal. DSSs introduced, instead, the need for systems that support rather than replace the manager. Still, Stabell argues, "decision support systems" is often a misnomer:

> "If a computer-based system is interactive, user-controlled, and easy to use, and if the manager is assumed to use judgement in arriving at a decision, then we have decision support. There is little explicit specification of the ***kind*** of use necessary for support of decision making. It is as if ***any*** use is decision support..." (Stabell 1983, p. 223).

In project management, a large number of computer-based tools exist under the rubric of ***project management systems***†. They tend to combine, in varying proportions, characteristics of expert systems, DSSs, and information systems. In their latter function, they provide information to the manager, although suggesting or prescribing ***what*** information is relevant to use, out of the abundance of information available, or ***how*** to use it, is not always supported (Ackoff 1967).

However, in most such project management systems, an underlying model of the whole project management process is found lacking. As a result, each comprises merely a cluster of tools mirroring particular activities the manager

† We discuss some of these in chapter 13.

has to do (e.g., workbreakdown, scheduling) together with facilities by which he may implement them. The lack of an underlying management process model also has the effect that the user's perception of the project management system is fragmented also. Often, such systems are used solely to perform certain management functions (such as initial plan preparation) and, for the rest, the manager retreats to his habitual pen-and-paper method of managing as this still involves less effort and time than struggling with tools within the system the use of which, in practice, involves more trouble than they are worth. An underlying, integrated, process model offers the possibility of avoiding this kind of trouble, and can be the basis of an integrated PMSS which has a good chance of being viewed as an indispensable mate by the manager. However, for the promise of such an integrated PMSS to be realised, a great deal of consideration needs to be given to how it can fit in with the everyday reality of project management and how it can help in the whole range of activities that are involved in it.

Last, but not least, issues relating to how work can be divided between the system and the project manager need to be carefully considered. Otherwise, a PMSS which does not clearly demarcate the boundary of its responsibilities and the manager's responsibilities, or a PMSS which takes all the manager's responsibility to be its own, will not find a satisfied user.

11.2. Division of labour between the manager and the support system

Moving from considering a single human being managing the project to considering how a project is managed by a hybrid being, comprising a project manager and a project management support system (PMSS), involves an appreciation of their relative capabilities and shortcomings as well as their relative capabilities for acting on what they have jointly decided. They, of course, both share responsibility for arriving at a "mutually satisfying" solution (Bennett 1983).

In relation to data that needs to be collected for interpretation or usage, the PMSS is likely to have a more reliable memory than the project manager: it should rarely "forget". On the other hand, the project manager has a wider appreciation of what is relevant to be remembered than the system does. Collecting data from the project work system and the project environment and working out whether to store it in his own memory, or that of the PMSS, is primarily the responsibility of the project manager?. Moreover, this hybrid being needs to act directly on the project in implementing decisions related to it; here again, it is the project manager who, usually, has the sole responsibility for this as, he, not the PMSS, will be held responsible for the results of his project

† It may not be entirely his responsibility in the case where the PMSS is able to collect data directly from other stakeholders such as the client, or members of the project team, but the project manager will still be responsible for monitoring the quality of this information and the timeliness of its arrival.

management activities, regardless of how they were initiated.

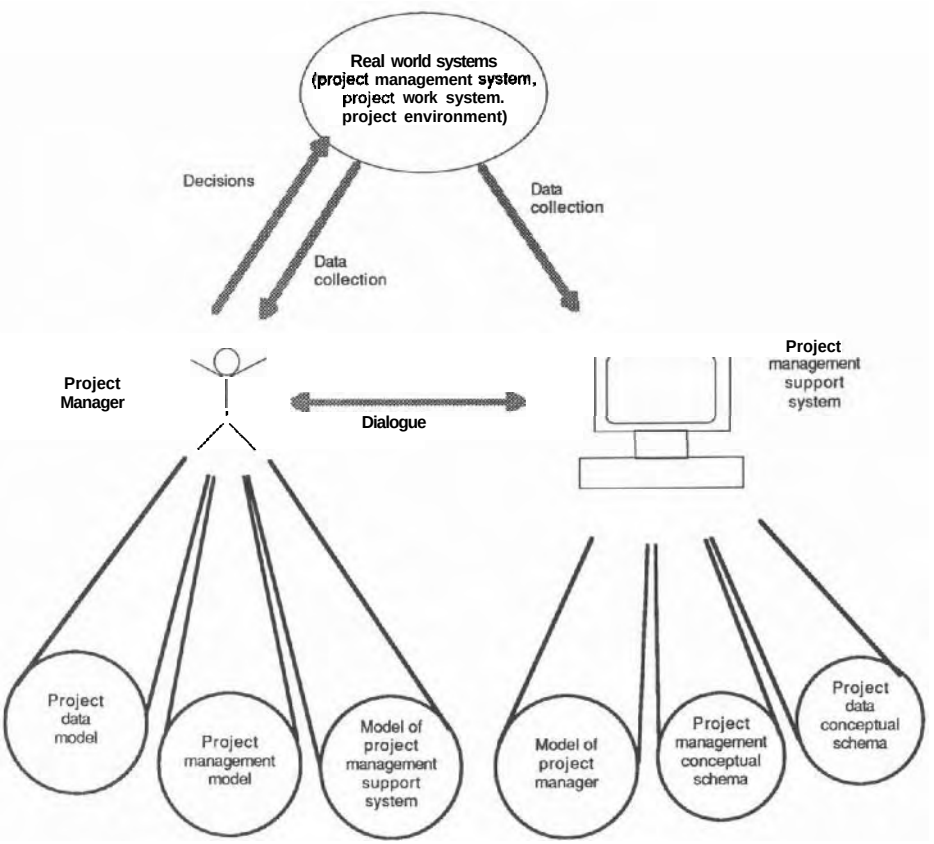
However, although the project manager appears to be the provider of information to his counterpart within the hybrid being and the implementer of any decisions taken, the PMSS is not a simple repository of "knowledge" about the project The notion of a hybrid being implies a fully interactive mode of operation of the PMSS, a direct dialogue between the project manager and the PMSS so that they can ***jointly*** perform project management activities. Since the project manager is the point of contact of the hybrid being with the real world, it is essential that he knows ***what*** are the types of information about the project work system and the project environment which are stored within the PMSS so that the responsibilities for ***what has to be remembered*** about the project can also be divided efficiently between the manager and the PMSS. For this he needs to have, in mind, a ***model of the project management support system*** so that he can be aware of its capabilities and be able to exploit the facilities it offers to him in an efficient and effective way.

In this respect, an ***intelligent*** PMSS can help through maintaining a ***model of the needs of the project manager.*** It can then provide him with useful information about the scope of what it "knows" about project planning and execution, and about how it may be able to work with the project manager in organising relevant information from the current instantiation of the project data model (as held in its memory and in the memory of the project manager) in constructing reports for various purposes. The PMSS may also be able to prompt the project manager at appropriate points in the dialogue to think about what he probably knows concerning some aspect of project management which is beyond the scope of the PMSS itself (Berkeley, Fernstrom & Humphreys 1987).

Figure 11.1 illustrates the situation where a project management support system (PMSS) works in interaction with a project manager in the manner we have outlined above. This may be considered in contrast with the situation shown in figure 6.1, which assumed that the project manager was performing his management activities on his own, without support. The roles of the project manager in relation to the project work system, the project environment and the project management system, and their instantiations in the project management and project data models, are still the same as that shown in figure 6.1. However, there are additional models and conceptual schemata to be considered and maintained.

As we indicated above, in order for the dialogue between the project manager and the PMSS to be effective in enabling the best use to be made of the PMSS's facilities, the manager needs to develop and maintain a ***model of the PMSS*** and, conversely, the PMSS needs to develop and maintain a ***model of the needs of the project manager.*** Also, the PMSS is shown as maintaining a ***project management conceptual schema*** (PMCS) and a ***project data conceptual schema*** (PDCS).

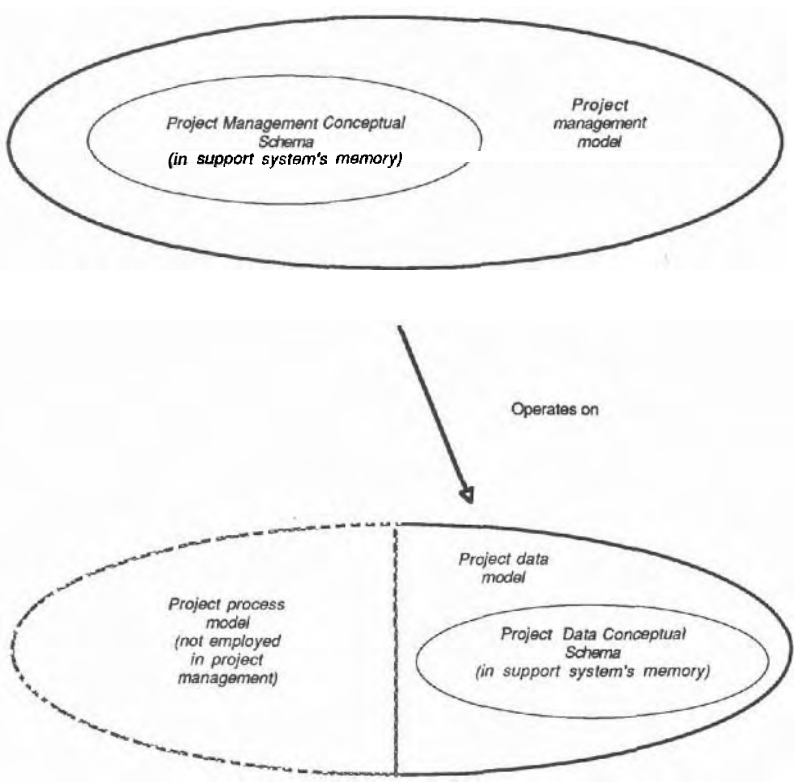
Figure 11.1: Division of labour between project manager and PMSS



The PDCS operationalises, within the PMSS, part of the full project data model (i.e., the one employed by the project manager and the PMSS, working in interaction). Although developed via the concepts of the project data model, it is specified to have a bounded and coherent structure, matching the requirements of the operations which may be made on it by the PMSS's techniques. Therefore, the PDCS consists of an enumeration of the classes of entities that the PMSS deals with, the relationships among these entities and the constraints on their instantiation (Tsichritzis & Klug 1978, Brodie 1984, Durchholz 1984, Ullman 1985). We will give a detailed example of a PDCS enumerated in this way in section 12.5.

Analogously, the **PMCS operationalises,** within the **PMSS,** part of the full project management model. It consists of an enumeration of the techniques within the **PMSS** which, collectively, constitute its capabilities for management support In chapter 12, we will discuss a range of these techniques, located at the levels in the **PMCS** which support project management model generation and tailoring, management methodology interpretation, **PMAC** execution, and provision of structured views in perspective on the **PDCS.** The relationship between the project management model, the project data model and their respective conceptual schemata is outlined in figure **11.2.**

Figure 11.2: Models and conceptual schemata



While the PMSS can take the responsibility for remembering information supplied to it about entities in the PDCS, the project manager should understand that it is ***his*** responsibility for remembering information about entities represented in the project data model but not in the PDCS. For instance, information about problems experienced with personnel in his team may exist in the instantiation of the project data model which the project manager carries in his head, but may never find its way into the PDCS, as this is material which the project manager wishes to keep confidential to himself†. Consequently, the project manager should not expect the PMSS to be able to support him by pointing out, for example, pre-existing team problems during interactive resource allocations.

Similarly, rules of thumb that a project manager uses himself when assessing a project plan, like those which we described in section **4.2.2.3,** should be considered to be part of the project management model but may or may not be instantiated in the PMCS. Whether they are instantiated will depend on the precise way ***plan reviewers*** and ***plan critics*** (discussed in section **12.2)** operate as a technique within the ***standard planning*** support environment which we will describe in section 12.3.

This is one example of why information about the ***differences*** between the contents of the PMCS and the project manager's own version of the project management model forms an important part of his model of the PMSS. If the manager knows that a particular rule of thumb is included within the PMCS, he may expect the PMSS to employ it appropriately, and report on the results if necessary. If he knows it is not instantiated within the PMCS, he should also realise that it his responsibility to decide when and how to use the rule and evaluate the significance of the results. Danger arises when the manager thinks that the PMSS will use a rule which is not actually in the PMCS. In the absence of any report from the PMSS on the issue addressed by the rule, the manager may think that all is well, when the rule, if employed, would actually have suggested otherwise.

For successful division of labour between manager and support system, it is important that their respective interpretations of entities in the project management and project data models, while not identical, be consistent and include the same notions in order to allow a productive dialogue. Moreover, in this dialogue, the PMSS should be responsive to the manager's needs for support in different situations: it must make the correct interpretation of what the manager wants to obtain, on the basis of the model of the needs of the project manager that it employs. It should also provide feedback to the manager which will enable him to establish whether ***his*** model of the PMSS itself is correct (i.e., the project manager should be able to confirm that the PMSS is "with him").

---

† We discuss further the issue of project data model entities which are not defined in a PDCS in section 12.5.4.

‡ We will describe the kind of support which can be offered by the PMSS during resource allocation when we discuss the *actual planning* support environment in section 12.3.

In summary, successful division of managerial responsibilities between the project manager and the **PMSS** requires that, in practice,

(1) the manager and the **PMSS** trace the same project data model entities over the full range of their interactions;

(2) the responsibilities for developing, storing and retrieving information are properly divided between the project manager and the **PMSS;** and,

(3) in building and using the project data model for management purposes, the dialogue is appropriately balanced between the project manager and the **PMSS.**

Considering division of labour within the project management process in this way lays the foundation for establishing which project management activities are best carried out by the project manager acting on his own, and which can profitably be shared between him and the **PMSS.** We conclude this chapter by discussing how to establish just how profitable a proposed division of labour might be for any particular management activity, through examining the associated costs and benefits to the project manager. Then, in chapter 12, we discuss *ways* in which project management activities may be shared between the **PMSS** and the project manager in both his ***management expert*** role and his ***project management*** role.

11.3. Deciding on the scope of a PMSS

In planning the global design of a **PMSS** it is necessary to consider the scope of the system that will be implemented, that is, the actual content of such a system (e.g., which techniques to implement, at what level of detail, what kind of learning support for the user to include).

For a start, the scope of any particular **PMSS** will be constrained by the effort available for the development and implementation of the system in any practical application (that is, the effort needed on the part of the developer to elicit user requirements, design, implement and test the developed system). The amount of the required effort, however, may not be easy to establish a ***priori.*** There is always the danger that the amount of required effort for implementing a particular support technique or the system as a whole may be exaggerated by the person who actually has to carry out the implementation if he is not directly responsible for the marketing of the developed **PMSS.** Furthermore, implementation effort may be wasted if the implementer does not fully understand the practical significance of required support technique; in such a case, a **PMSS** may be developed which performs limited sets of segmented support techniques without, for example, a model of the overall process within which these techniques are embedded. In any case, it is important that the available implementation effort is not misspent; the degree of user satisfaction resulting from a given implementation effort should be as great as possible.

A successful support system is a system that is, and continues to be, used once it has been delivered to the client. Ensuring that it does is not an easy task

as it entails that the system both satisfies its user requirements and helps the manager carry out his work while, at the same time, does not necessitate that the user invests a great deal of his time entering necessary data into the system for a perceived little return. In other words, it is important that the system be attractive to its users in terms of the perceived ***benefits*** they obtain by using it (rather than using another system or no system at all). For success, these benefits must be perceived by the user ***as*** outweighing the ***costs*** incurred in using the system.

In a **PMSS** where, when in use, the system's model of the needs of the user and the user's model of the system are well in tune with each other, the costs incurred by the user due to the frustration of finding that he cannot work with the system in the way he anticipated can be neglected, as such situations will rarely arise. Another source of cost to the user is the financial wsts of buying the system. Prices of such systems vary widely just as their capabilities do.

Our primary concern, however, is with the cost to the user which is primarily related to the time it takes him to enter the necessary data into the system for it to perform some required support function. This may be considered proportional to the amount of key strokes, mouse clicks, etc., weighed by the inconvenience of making each type of operation as perceived by the user.

The ***data entry effort*** for each **PMSS** support technique of interest may be considered to be a function of the amount of data the project manager is required to enter into the **PDCS** when using the technique where neither can the system provide support without this data, nor has the manager unlimited time to respond to excessive demand for data entry by the system. Kelly (1988) describes the high level of data entry effort as a main worry about existing project management software which, in general, can take up so much time that "its use could easily become a substitute for the actual management of a project" (p. 88).

However, while it is fairly straightforward to assess data entry effort for a **PMSS** as a whole, it is more difficult to apportion that effort amongst individual support techniques of the **PMSS.** This is particularly true for any integrated **PMSS** which employs a single unified **PDCS** and comprises a **PMCS** which dismbutes its support functions across a wide range of techniques supporting project management activities. In chapter 12, we describe how, within this kind of **PMSS,** it is possible to identify individual support techniques as modular components of the **PMCS** which can be dynamically clustered into support environments at **PMAC** level, as required. This raises the possibility of examining the expected data entry effort associated with any support technique proposed for inclusion in the **PMSS** as an input to the decision about whether it should be implemented in practice.

Nevertheless, it is important to note, in assessing this type of cost to the user for any particular technique, that whenever it is activated within a support environment, much of the data on which it operates will not have to be entered by the user during its activation. This data will already be available within the current instantiation of the **PDCS.** It will have got there through some previous

operation canied out through an earlier activation of another support technique. This raises the difficult problem of working out how the assessed data entry effort is to be apportioned between the support technique which originally aided the project manager in getting the relevant data into the PDCS and all the other techniques which may subsequently make use of it

The benefits to the user for each technique in the PMSS which the manager may use can, in theory, be defined in terms of its ***gross functionality,*** that is, the perceived usefulness of the technique in all its various applications (i.e., in the context of the whole set of support environments into which it may be clustered) in supporting the work of the project manager.

The gross functionality of the outputs of a technique can be ascertained by interviewing potential users and asking them how useful they would consider the results which can be gained through its use. However, individual project managers may have their own particular ideas about which of the techniques on offer in a support environment they will be willing to use more than others, depending on their own knowledge, abilities and experience gained through previous use of other PMSSs and project management tools. Furthermore, especially in cases of radically new suggestions, the potential user may not be able to speculate on the power of these suggestions or on their usefulness in practice as he will be bound by what he already knows or what he can envisage. Thus, such evaluations are better made on the basis of a prototype which can function as a test bed for the introduction of the techniques to be evaluated.

However, it would be a mistake to attempt to decide on the scope of a PMSS simply by measuring separately each of the techniques which might be potentially included against the types of costs and benefits discussed above, and, then, selecting those techniques which score best on the cost-benefit balance sheet. Adding or removing a particular technique can affect the ***interaction*** between user costs and benefits within the ***overall*** PMSS, when viewed in its entirety. For example, adding a technique may increase the demand for entering data while, at the same time, providing some data to the system which will be required by another technique. Furthermore, adding a technique can only be measured against the already existing PMSS techniques and the support functions they collectively provide.

In general, estimating the values of these costs and benefits to the users is not an easy or contention-free task. For example, user satisfaction with the system can not be properly ascertained until the system or, at least, a prototype of the system has already been developed, leaving it to developers to speculate on user satisfaction may lead to the development of a system which is ideal, in the developers' view, on the criteria they themselves would weigh heavily in anticipating user satisfaction, but which leaves much to be desired on the criteria which are eventually important to the actual end-users. Divorcing the user's view of the PMSS from the developers' requirements analysis in this way has been shown to be a major cause of users' rejection of support systems in practice (McCosh 1984, Paprika & Kiss 1985).

Chapter 12

Support system design issues

The previous chapter addressed some of the issues one needs to consider before embarking on the design of an integrated project management support system (PMSS), examined the division of labour between the manager and the PMSS and discussed the basis for deciding on the support techniques to incorporate. The present chapter focuses on interpreting what has been discussed so far in this book from the point of view of developing functional specifications for an integrated PMSS.

In our description of project management activities in earlier chapters, their diversity and their differing scope and complexity were commented upon in some detail. We described how they can be modelled, and linked, in a unified way, and how this can inform good management practice. We also identified two separate project management roles in this practice:

(a) the ***management expert*** role, primarily concerned with management model development and management methodology interpretation, taking into account the needs and context of the project to be managed, and,

(b) the ***project manager*** role, where activities aimed at achieving project management goals may benefit from management process guidance developed in the execution of the management expert role.

In the following, section **12.1** discusses what is involved in supporting a manager in the management expert role and how he can be helped in generating and/or tailoring a project management conceptual schema for his own use. The rest of the chapter concentrates on PMSS design issues in regard to provision of support for the project manager role. Section 12.2 describes how division of labour within a PMAC can be carried out, and section **12.3** shows how this division of labour provides the basis for the definition of a support environment and the techniques within it that can help a project manager carry out a particular PMAC. Section 12.4 describes how the local process conceptual schema for a particular support technique can be determined using the ***estimation*** technique as an example. Section **12.5** addresses the issues involved in designing the project data conceptual schema by the use of an example from

the PIMS project which has implemented such a schema in designing a project management support system. Finally, in section 12.6, we discuss how watch-*dogs* can be set to watch entities within a project data conceptual schema in a way that will safeguard the achievement of project management goals.

12.1. Support for the management expert role

In chapters 6 through 9, we have discussed a unified set of techniques, based on predicate-transition net modelling which, in theory, can provide the basis for generation, tailoring, refining, precising and simulating a project management model. and we illustrated how this can enhance the understanding and execution of a full range of project management activities. Yet, in practice, project managers rarely seek out computer-based management modelling support in order to generate and explore the structure of the project management model, despite the fact that it describes the system which comprises their own activities.

Instead, as we indicated in section 6.3.4, they tend to rely on intuition (seat-of-the-pants expertise), constrained by management methodologies. The latter are usually treated as imposed external requirements, setting constraints on the manager's intuitive view on which management activity should be carried when, how, on what, and in what manner. Set in handbooks, these methodologies are mainly pre-structured, focusing, rather inflexibly, on just those aspects of the manager's activities where the constraints which result from such inflexibility are likely to promote good rather than poor management practice. As such, they should not be taken as offering comprehensive support for project management model development, or selection, and utilisation, as that is not their real purpose.

This does not imply that project managers constitute some sort of species, possessing an inherent dislike of any sort of model development and simulation except when conducted entirely intuitively and without support. Indeed, most project managers are familiar with techniques for modelling partial aspects of the project work system (e.g., techniques for task decomposition, for defining task-product relations) and seek support for their use within their project planning activities†. Rather, we consider that the problem, up until now, has lain with the lack of appropriate modelling support available through computer-based modelling packages (at least, those which are widely commercially available).

In chapter 6, we described how the process of model development and, by implication, support for this process, is quite different in the case of modelling the project management system, compared with modelling the project work system. We indicated, in section 6.3, how an interactive *coloured Petri net* model

† We will discuss the provision of such techniques within project planning support environments in sections 12.2 and 12.3.

development and simulation system like Design/CPN (Albrecht, Jensen & Shapiro 1989) has the necessary capabilities to support the approach to project management model generation we have described in this book However, this system, while commercially available, is still new and unfamiliar to project managers. It is also likely to *remain* unfamiliar to them as it is targetted at process modelling experts in general, and, therefore, does not take account of the operational language and organising concepts developed within the context of project management.

Our own solution to this problem was to re-develop the open architecture underlying Design/CPN to provide a system which offers the required modelling capabilities but provides support in a form that is anchored in substantive concepts, and views on what is being developed, which are familiar to the project manager. The support capabilities are bolstered by context-sensitive help which is sensitive to both the context of the model, as currently under development, and to the context of the application being modelled†.

Within a fully comprehensive management modelling support system, support can be offered for the following three facets of the management expert role:

(1) generating and exploring the project management model;

(2) tailoring the project management model for use in a particular project; and,

(3) interpreting a management methodology in terms of constraints to be set at various levels within the project management model, as tailored.

In sections 12.1.1 through 12.1.3, we outliie some possibilities for the provision of support for each of these three facets. In our discussion, we will not be concerned with whether the actual execution of a PMAC will be carried out by the project manager (acting within the project manager role) on his own or in interaction with support techniques offered by the PMSS. This is because the general issues and principles in provision of support for management process guidance at the level of generating, tailoring, selecting and launching PMACs are the same in each case. The differences occur when we consider the provision of support for the project manager within *PMACs*, and the latter will be the focus of sections 12.2 and 12.3.

† Only a small part of these capabilities were built into the PMSS developed within the PIMS project, which was not able to support many of the management model development activities we describe in this book. A system with the full range of support capabilities is currently being developed by Berkeley and Humphreys at the London School of Economics within the context of the IGOFOR (Interactive *Generative* Organisational Frame *of* Reference) project which is part of the UK joint research councils' *cognitive science/HCI initiative* (1990-1992). Versions of this management modelling system will be incorporated in the PMSSs which are successors to PIMS.

12.1.1. Support for project management model generation and exploration

Support for this facet of the management expert role focuses on ***variable precision*** management modelling, helping the project manager to initiate modelling by precising the distinction between the active and passive elements contained in the informal scripts which describe the relevant management activities, in the manner we described in section 6.3. This permits model development at the level of approximate net modelling. PMACs may be defined and linked, and their internal structure refined in the way we discussed in chapter 8. Facilities should be offered such that the passive and active elements in the nets (places and transitions) may initially be described by legends in the natural language of the project manager and, subsequently, be precised into inscriptions, as and when required. Similarly, PMAC pre-conditions and post-conditions may initially be described in natural language which, later, may be precised, with the aid of the modelling system, into expressions.

Sometimes, as we described in section 6.3.3 (and gave examples in chapter 9 in the case of inferencing about observed or anticipated problems in the running of the project), it is desirable to increase the precision of modelling of the relevant aspects of the project data model within a ***local process model, so*** that the dynamic as well as the static aspects of the object system can be coherently modelled. This increase in precision can be facilitated if support is provided for

(a) precising the rules of change which denote the precise nature of the transformations made on the links (in terms of tokens consumed and produced); and,

(b) precising the logical content of the inscriptions into predicates which can be delivered as tokens and thus be tested, generated and revised during transitions through the local process model.

A successful achievement of the above offers the possibility for the project manager to come to understand the properties of the local process model through asking what-if questions: the system then provides support by examining how the local process model simulates the dynamic implications of such questions. In this way, ***management strategy decision support*** can be provided through exploration, allowing alternative scenarios to be tested, side-effects to be investigated, and contingency planning to be validated.

12.1.2. Support for project management model tailoring

The full set of possibilities offered by the management expert role in regard to project management model generation from scratch is usually not exploited by a project manager when playing this role. Instead, he will expect, and be expected to, inherit much of the project management model which he will instantiate and use to inform and guide his management activities on his project. The inheritance will come in the form of ***pre-structured model***

components, established ***a priori*** within the contractor organisation (or elsewhere) as forming a potential basis for good management practice. The project manager, acting within the management expert role, will then wish only to refine, tailor and link the inherited components to meet his interpretation of the management requirements for the project and, in the case of management activities left to his discretion, to conform to his own management style and preferences.

There is a recursive flavour to all this: it implies that one of the first management activities a project manager should carry out on taking over a project is to tailor the management process which will then prescribe and guide his own (subsequent) activities. This is why ***tailor management process*** was modelled pre-formally as a PMAC itself in chapter 7. Support for the activation of this PMAC is facilitated by providing, within the management modelling support system, a library of ***pre-structured management model components,*** built up through previous use of the management model generation facilities we described in section 12.1.1. These components can be formed at a number of levels of refinement according to whether they constitute templates for linking complete PMACs, subPMACs or local process models.

The use of such a library in project management model tailoring is greatly facilitated by the development of a modelling system which can handle ***hierarchical*** coloured Petri nets in the manner we described in section 6.3.1. Each pre-structured model component is then catalogued as a ***subpage*** at the appropriate level in the subpage refinement hierarchy. Storage and retrieval of the appropriate pre-structured component for use in any particular modelling context can then be supported very effectively through recording each component as an entity within an object-oriented modelling component catalogue. The property information about each entity (component-object) can then describe not only its level within the refinement hierarchy, but also its origin, function, application domain, level of precision, input and output predicates, and so on. In other words, the full set of information about what the component is, and ***where*** and how it can be used in model development and tailoring can now be made available to the project manager, as and when required.

The model tailoring process now involves selecting and linking of pre-structured components from the library, coupled with editing and refining of their content when required, together with the generation of just those parts of the project management model where the manager wishes to take an original, novel or idiosyncratic approach to the management of his project. The facilities required to support this process are much the same as those we described for supporting the process of management model generation and exploration in the previous section, and so a single management modelling support system may offer support for model generation and/or tailoring, as required.

12.13. Support for methodology interpretation and management process guidance

The project management modelling facilities described above can also provide support for project management methodology interpretation, viewed as a special form of method tailoring. In section 6.3.4, we described how a management methodology, externally prescribed for a project, may be viewed as a set of *partial constraints* to be imposed, at various levels, on the project management model instantiation which may be made through activating the *tailor management process* PMAC. These constraints then serve to regulate the conditions under which particular PMACs, or subPMACs, may be executed (according to the instantiation), thus shaping the project management process, where and as necessary, to conform to the prescribed methodology. Similarly, in local process model instantiation, the methodology can prescribe the strategy to be interpreted by the process control component in the way we described in section 9.2, where we gave the example of a *prescribed reasoning strategy* guiding the control of a reasoning procedure for making inferences about the progress of the project

Once the tailoring and methodology interpretation process is complete, in the eyes of the project manager, the resulting instantiation of the project management model may be checked for coherence and consistency by the support system, thus identifying any necessary revisions or extensions. The PMSS can then employ this instantiation to provide guidance to the project manager on PMAC selection and activation (conducting their pre-condition tests), coupled with monitoring the success of activated PMACs (conducting their post-condition tests).

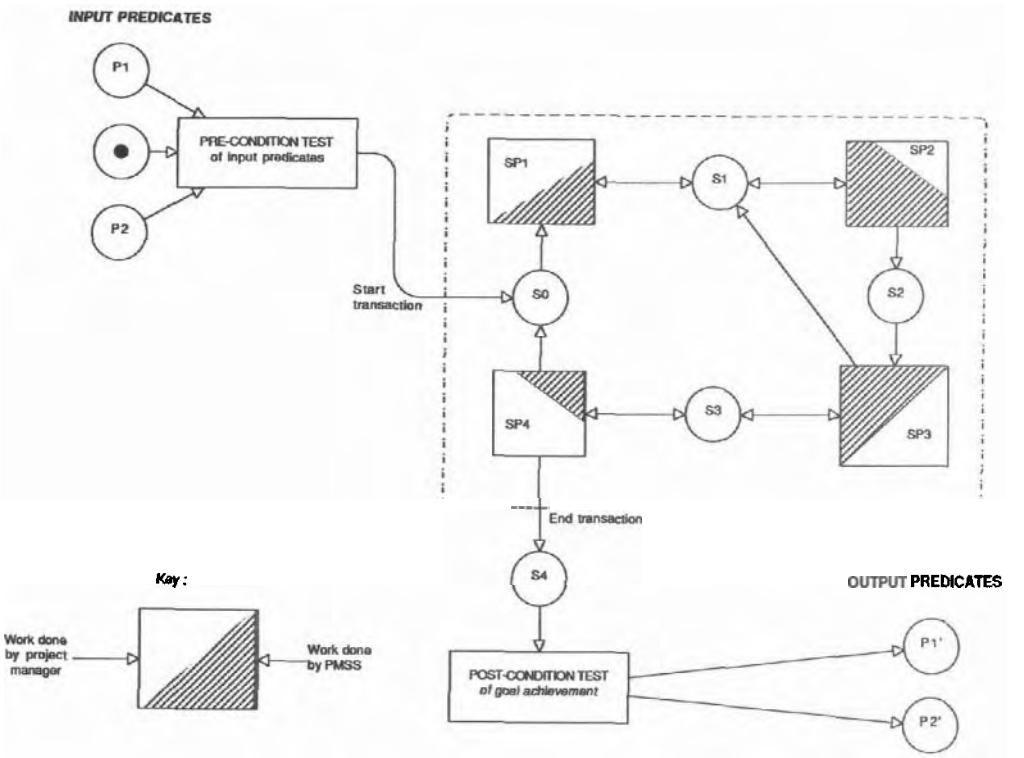
12.2. Division of labour within a PMAC

While some of the project management activities (PMACs) identified in chapter 7 may be carried out by the project manager with no support from a PMSS, the converse is unlikely to be true: no PMAC can be carried out solely by a project management support system without *any* of the operations involved being carried out by the project manager. Thus, we need to examine how labour may be divided between manager and PMSS *within* a PMAC in order to address which part of it is carried out solely by the project manager and which part of it can be carried out by the PMSS. This examination can be made at any desired degree of refinement within the project management model by partitioning the subPMACs represented as elements within the internal structure of a PMAC at that degree of refinement. Figure 12.1 gives an example of partitioning the *standard planning* (SP) PMAC, which was shown in figure 8.1, in this way.

In figure 12.1, the extent of the shaded portion of each subPMAC gives a rough indication of the proportion of the operations within that subPMAC which are carried out by the PMSS rather than the project manager when they manage the project in interaction. In order to indicate exactly which these

operations are, it would be necessary to make a full refinement of the contents of each subPMAC down to the level where the individual operations are shown as discrete elements in a local process model†. For the sake of brevity, we will not attempt this degree of refinement here. Instead, we will simply indicate, in figure 12.2, the conventional names for the support techniques which could offer the kind of support desirable from a PMSS to the project manager in carrying out operations involved in the activation of each subPMAC of the *standard planning* PMAC.

Figure 12.1: Division of labour within the standard planning PMAC



† Local process models were discussed in chapter 9.

In the case of subPMAC SP1 ***(organise work definition into a task hierarchy),*** support can be provided by the PMSS through techniques which aid the process of creating a coherent workbreakdown structure and providing a view on the task hierarchy being instantiated (as indicated in figure 8.1). Assistance in estimating the effort required by each instantiated task, and in composing these estimates into an overall estimate for the project would also be desirable.

In the case of subPMAC SP2 ***(coordinate products between tasks),*** assistance can be offered in terms of ***interactive*** task scheduling, building and displaying critical path and PERT diagrams, and providing interactive facilities for making duration estimates and defining precedence relations between tasks, resolving "bottlenecks" involving tasks on the critical path, and so on.

In the case of subPMAC SP3 ***(control allocation of standard resources),*** assistance can be offered in semi-automatic allocation of "standard" resources within a common time frame (aided by the use of some automated standard rules) enabling the manager to explore more easily the possibilities of resource levelling in terms of effort and balance of skills required at any particular time. This, in *turn,* facilitates team building as the demands made on the human resources in the team become more consistent with what can be offered without frequent changes in team membership and commitment to the project. Once resource allocations have been made, cost estimation support can be provided, taking into account estimated effort, characteristics of "standard" resources supplying this effort and other costs estimated to be incurred by tasks.

Finally, support can be provided in subPMAC SP4 ***(test adequacy of standard planning) in*** assessing the quality of the manager's standard plan. As we discussed in section 4.2.2.3, errors in creating a project plan may be due to oversight or foresight on the part of the manager. ***Plan critics*** can ***be*** used to determine errors of oversight concerning, for example, overallocation of a resource, or needed products by a task not being produced by another task. On the other hand, ***plan reviewers*** can review the plan to determine any errors in foresight such as the existence of too many dependencies between tasks, if tasks of long duration have intermediate products, and so on†. Plan critics and reviewers can take advantage of the fact that all techniques operate on, and build, a ***single,*** comprehensive, instantiation within the project data conceptual schema (PDCS), and, so, review the results of the activation of the other subPMACs, taken as a whole, rather than within the perspectives used in activating any particular subPMAC, or on the basis of the results of using a particular technique, considered in isolation.

It would be dangerous to expect, however, that plan critics and reviewers could be specified in terms of a set of automated rules, operating on a PDCS instantiation. The PDCS contains only that part of the results of the manager's standard planning activities which it is the responsibility of the PMSS to hold in its memory. Much more of what the project manager will now know or can

† Some rules which can be used to that effect have been discussed in section 4.2.23.

imagine about the standard plan for the project will be held in ***his*** memory rather than the PMSS's memory. Hence, a technique implementing plan critics and reviewers should aim to ***make suggestions*** to the project manager about how he should think about and review his planning (in all its aspects) as well as making ***ex—nations*** of what can be accessed within the PDCS.

12.3. Building the support environment for activation of a PMAC

In the previous section, we discussed provision of support techniques in the activation of individual subPMACs, but much of the power of an integrated approach to PMSS design would be lost if we considered only the serial activation of individual support techniques and their associated displays (in perspective) of aspects of the current instantiation of the project data model. For instance, in section 8.1, we described how it would be a good idea to maintain the workbreakdown view on the task structure while developing a PERT or CPM diagram. If the workbreakdown view is tied solely to the workbreakdown technique, and this technique is tied to subPMAC SP1, then, any changes in instantiations made in subPMAC SP2 will not be reflected in the (now deactivated) workbreakdown representation, even if it is left on display for the benefit of the project manager. Conversely, support for estimation activities is required (in various forms) during the activation of the subPMACs; so why not have central estimation support facilities available throughout?

Considerations like these suggest that, rather than defining specific support techniques to be activated at subPMAC level, a better strategy is to define a ***support environment*** at PMAC level. In this approach to PMSS global design, the techniques which support the activation of any of the subPMACs are clustered into the PMAC support environment and define it. Any of the clustered techniques is available for use at any time, at the discretion of the project manager during PMAC activation. Even more important, the views on the PDCS offered through the activation of any of these techniques can, at the discretion of the project manager, remain on display, and be dynamically updated as the PDCS entities it has in view change through creation or deletion of instances or (re)setting of their attributes, even though these changes may be effected through the use of a different support technique than that which was activated when the view or display was originally generated.

Figure 12.2 shows how such a support environment may be derived for the ***standard planning*** PMAC. In this figure, techniques for workbreakdown and effort estimation are shown supporting the activation of subPMAC SP1 ***(organise work definition into a task hierarchy);*** techniques for scheduling and duration estimation are shown supporting subPMAC SP2 ***(coordinate products between tasks);*** techniques for resource allocation and effort estimation are shown supporting subPMAC SP3 ***(control allocation of standard resources),*** and techniques for plan critics and plan reviewers are shown supporting subPMAC SP4 ***(test adequacy of standard planning).***

These support techniques can, collectively, be activated and clustered to form the standard *planning* support environment. This clustering brings together three variants of the estimation technique, where the main difference concerns the *aspect* that is being estimated (i.e., effort, cost, or duration). This offers the possibility of providing unified support for the process of estimation and offering this support directed at the particular nature of the aspect being estimated at any time. In section 12.4, we will examine in detail the user requirements for estimation support and, then, we will show how the local process conceptual schema for an estimation support technique of this kind can be derived from them.

Figure 12.2: Specification of support techniques in the standard planning environment through abstraction from the PMAC's internal structure

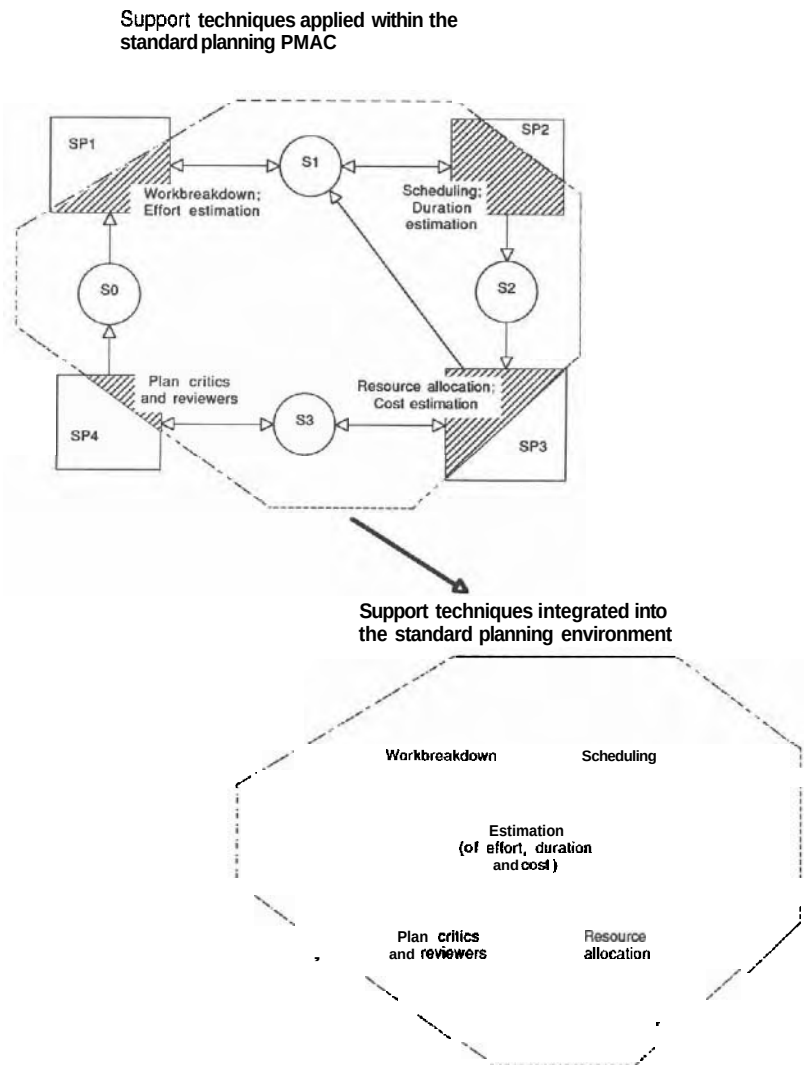


Figure 12.3 shows how a support environment may similarly be derived for the ***actual planning*** PMAC (as modelled in section 8.3) based on an internal **structure** comprising the three **subPMACs** shown in figure 8.8. It indicates that certain of the techniques which provided support within the standard ***planning*** environment may also be employed to provide support in the ***actual planning*** environment. These are the techniques for scheduling, resource **allocation,** estimation and plan critics and reviewers.

Figure 123: Specification of support techniques in the actual planning environment through abstraction from the PMAC's internal structure

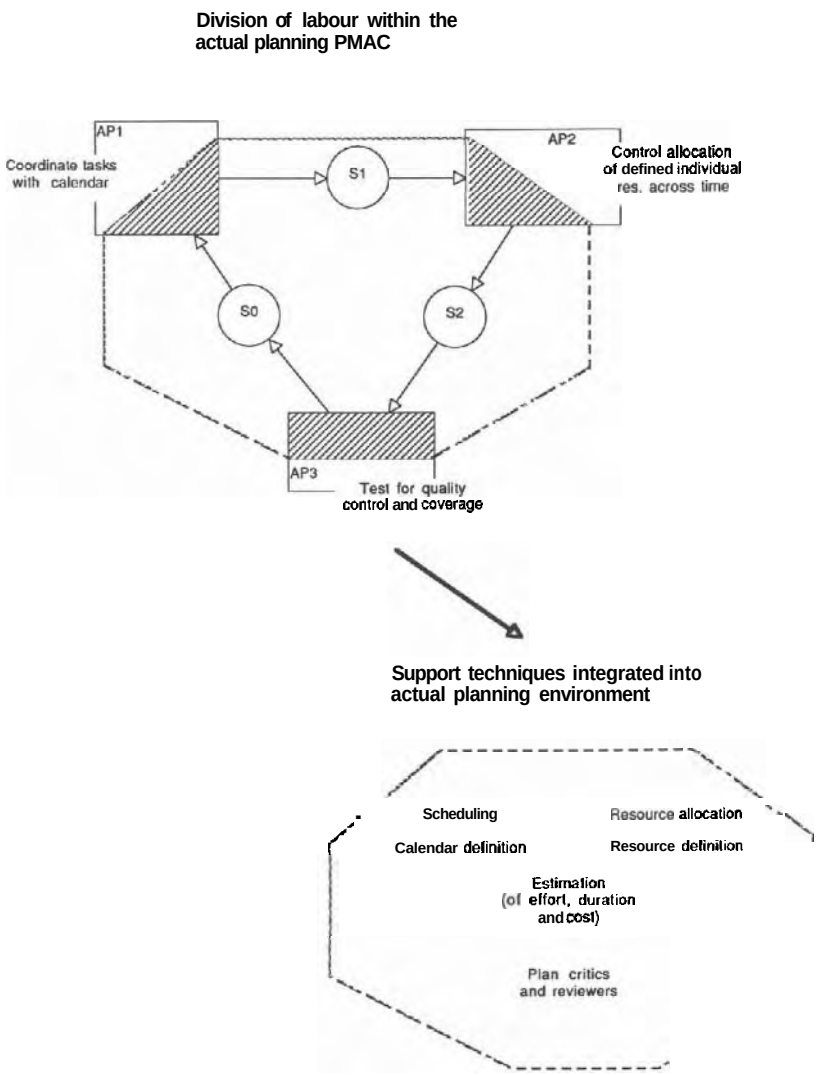


Figure 12.3 also indicates that additional techniques are required in supporting ***actual*** rather than ***standard planning.*** These are:

(1) a ***calendar definition*** technique, which allows one to coordinate the schedule with the company calendar, and to define calendars for the individual resources assigned to the project, showing their planned task assignment, personnel holidays, etc.

(2) a ***resource definition*** technique, which is required now that actual rather than "standard" resources are to be allocated to tasks.

When used in support of actual planning, the resource allocation technique will also have to take into account the additional constraints which may be imposed through the unavailability of actual resources during periods when they are assigned to other tasks or other projects, when they are on holiday, or otherwise indisposed. Resource substitution may be necessary, and additional tasks may need to be added to the task hierarchy?. Effort and duration estimates may have to be revised for tasks according to the problems of actual resources assigned to them. Cost estimates may be refined, now that the time spent on tasks can be converted to cost for estimates at the rates charged in reality for actual resources rather than those assumed for hypothetical standard resources.

One technique used to support standard planning, i.e., workbreakdown, is not shown in figure 12.3. This is because, during actual planning, the manager is assumed to be primarily interested in the scheduling and resourcing of leaf tasks in the task hierarchy rather than re-organising the task hierarchy itself. In fact, as we described in section 8.3, if the difficulties encountered in scheduling and resourcing during actual planning are sufficiently severe to indicate the need for redoing the workbreakdown, the preferred course of action should be to suspend the ***actual planning*** PMAC (and its support environment) and re-activate the ***standard planning*** PMAC and, perhaps, re-activate the ***negotiate changes*** PMAC as well.

However, the absence of the need for a workbreakdown support technique within the ***actual planning*** environment does not mean that there will never be a need during actual planning for the project manager to take a ***workbreakdown view*** on the task hierarchy during actual planning. Such a view may be very useful in revising effort, duration, and cost estimates as it allows the project manager to see how the changes in estimates on individual leaf tasks are propagated up the hierarchy in ***bottom-up*** estimation, and how, at the top, they affect the estimate for the total project. Where the resulted estimates exceed time or financial constraints at any point in the hierarchy, it is also useful to employ this view in revising estimates ***top-down,*** to establish how and where the necessary savings can be made to bring the project, as planned, back within its budget.

---

† For example, a useful strategy is to add a ***training task*** to train a resource who does not initially have the expertise to meet the resource requirements of a task to which he will subsequently be assigned, but who should be able to develop the necessary skills and experience as a result of the training he is now scheduled to receive.

From the above discussion, we can draw three major conclusions about the relationship between support techniques and support environments. These are:

(1) Techniques may be ***dynamically*** offered to the project manager to be added into a support environment at the time they are needed in support of the activation of a particular **PMAC.** Environments do not own techniques; a particular technique may be allocatable to a number of environments. Conversely, support techniques used in more than one **PMAC** do not need to be duplicated, they can be dynamically clustered into the relevant environment at the time that they are required which may then be immediately offered to the project manager in supporting his current project management goal.

(2) The particular techniques nominated to be clustered within a particular environment will depend upon the way the internal structure of the **PMAC** is generated in building or tailoring the project management model.

(3) It is desirable to define the views required on the PDCS at ***environment*** level rather than at technique level. The views available within an environment must be sufficient for the project manager to use the support technique successfully and efficiently, but views the creation of which was not the responsibility of the currently activated technique may, on occasion, be profitably employed by the project manager (providing they are kept up to date). In this way, the project manager is provided with useful contextual information, or alternative views, facilitating the checking of consistency, or the safeguarding of his goals as he works.

The above points summarise the basis for establishing PMSS functional design at the **PMAC** support environment level. In the next section, we will look in detail at the basis for establishing functional design at the technique level. We will take as our example a specific technique, which we have illustrated as having a central role within both the ***standard planning*** and the ***actual planning*** environments, that is, the estimation technique.

12.4. Determining the local process model for a support technique

The discussion in section 12.3 indicated how support techniques may be identified and grouped into support environments, but did not go into details about ***how*** the functional design of any particular technique could be derived. This involves addressing two separate but interrelated aspects:

(a) what kind of process support is offered to the manager by the technique;

(b) what types of information need to be stored, utilised and retrieved by the technique.

In the following, we take as an example the estimation technique which we showed, in section 12.3, to be of central importance within both the ***standard planning*** and the ***actual planning*** support environments. We start, in section 12.4.1, by examining the estimation process from the viewpoint of the project manager, and from this, in section 12.4.2, we derive the support requirements

for estimation. Then, in section 12.4.3, we review the ***local process conceptual schema*** (LPCS) for particular implementation of an estimation technique, showing how it interprets these requirements in the process of making estimates in interaction with the project manager. We also identify the entity classes within the PDCS which are operated upon by the estimation technique in storing, utilising and retrieving the relevant information in the formation of estimates.

12.4.1. The estimation process

Estimating cost, duration or the effort needed for the whole project or any of its constituent tasks is undoubtedly a difficult task as it involves envisaging the ***future*** on the basis of what one knows about the ***past.*** Rightfully, DeMarco's (1979) parting words on estimation are:

> "Estimates deal with the unknown, and the unknown has a perverse way of subjecting poor developers to all kinds of rude shocks. I h o w of only one thing that keeps these rude shocks to a minimum, and I shall take this opportunity to pass it on to you: Good Luck!" (p. 339)

Estimation is an iterative process where repeated attempts are made to improve already existing estimates as more information becomes available. Within this iteration, we may distinguish three phases with different user requirements for support. These are

(1) estimation within the pre-project establishment phase;

(2) estimation within initial planning once the project is established, and,

(3) estimation within re-planning during the running of the project when information becomes available which indicates that current estimates need to be revised.

Our discussion in chapter 3 has already addressed many of the issues involved in estimating during the pre-project establishment stage (i.e., for the purposes of bidding for the contract) under the topic of project risk. Optimistic estimates of cost, duration or effort at the time, we have argued, can endanger the success of the project as they provide an unrealistic brief to the project and its manager. The estimation methods selected for use in this phase typically reflect organisational policies and practices and may not correspond to the actual project manager's preferred mode of estimating, once the project has subsequently been established. Typically, cost estimation models like COCOMO (Boehm 1981) or SLIM (Putnam 1978) are used whereby certain global criterion variables (such as personnel productivity, product complexity, size, etc.) need to be assessed for the particular project and, through some algorithmic rules, the cost of the project or the effort needed to carry out the project is calculated.

The problem with such models is that some of the critical values on which they depend are not yet known but are themselves rather unreliable best guesses. For example, personnel productivity, if measured in terms of function points, can not be reliably estimated until much of the design work is complete (Albrecht & Gaffney 1983) or, if measured by lines of code per time unit, can

not be reliably estimated until after coding and product testing is complete.

Another method of estimating during the early bidding stage is through the use of a number of different estimators, themselves experienced project managers, who can use their collective rich experience to reach some realistic estimates for the project They typically start from an agreed-upon coarse work-breakdown structure for the project to which nodes they attach estimates according to their individual experience either working *top-down* through the task hierarchy or *bonom-up,* that is, starting from estimates of the lower level tasks. Differences and similarities between different managers' estimates are then discussed and an agreement is sought as to which of the diverging estimates (if any) can be adopted as the operative one. An averaging technique is prone to be biased by extreme estimates while a group discussion until agreed-upon estimates can be reached can be subject to group dynamics whereby dominant individuals' estimates may be finally adopted (Fairley 1985). The use of DELPHI methods sometimes succeeds in counteracting these problems (Helmer 1966, Boehm 1981).

Whether through the use of parametric models or through the collective experience of different estimators, the estimation process involves comparing the particular project to other, *analogous,* projects of the past which constitute part of the experience of the estimator (Cowdemy & Jenkins 1988). Thus, the ability to estimate accurately is as good as one's ability to extrapolate from past experience elements relevant to the current project. Moreover, defining a project as analogous to the one under consideration necessitates defining the precise similarities and differences between the two projects.

In estimation during the project planning phase, the project manager inherits the initial estimates made during the pre-project establishment phase, some of which will now operate as *constraints* on *his* planning, particularly where they have been used to define requirements for the project (budgets, delivery dates, committed resources, etc.). The project manager will, however, wish to check and reline these original estimates. They will need to be made to relate to the individual tasks in the task hierarchy produced through his workbreak-down creation activities, and may then form part of the brief that the project manager will pass on to his team members.

However, estimating effort and duration for individual tasks or groups of task may be just as difficult as making the global estimates in the pre-project establishment phase. Many of the issues and problems we discussed above concerning the process of making estimates materialise here as well, although now at the level of making estimates for tasks. This necessitates the need for information from past experience at this level that is, forming analogies with other similar tasks, either from this or other projects. It involves having to cope with many more entities to estimate although under the advantage of more information being available about the project itself.

In this phase, accuracy in estimation is even more critical than in the pre-project establishment phase, particularly as the outer bounds for the quantities being estimated will now have been set within the terms of the contract and requirements for the project, and may be very difficult to re-negotiate.

It is a common practice in this phase (and often put forward as a method of promoting staff motivation and commitment to a task) to ask the personnel who will carry out a specific task to produce their own estimates for the particular piece of work. These estimates are then checked by the project manager against his plan and his own experience, and used (or not) in deciding on the operative estimates for his project

Once the project starts running, more information is gained both on the nature of the work being undertaken and on how far the assumptions made during the earlier estimation processes have been correct (concerning, for example, productivity rate, complexity of project, and so on). If the work to be done had not been properly understood during the earlier estimation phases, this will usually become apparent around the time of system design and may result in re-estimating as a basis for re-planning. Re-estimating and consequent re-planning may also become a necessity when, during monitoring, actual effort and time used up by tasks are found to be more than their corresponding estimated ones. This, for example, could be due to incorrect information having been used in the forecasts, or the productivity rate not being as expected, or a higher level of unexpected problems having occurred than was originally anticipated. Alternatively, there might have been an inaccurate record of progress and the discrepancies may not be as significant as they appear.

In the case that it is obvious that the fault lies with over-optimistic estimates which did not stand up well to their reality-testing during the running of the project, a strategy which can be used (provided that data is available) is to analyse trends in estimates and to identify the persons who were responsible for these estimates. By doing this, the project manager may discover that particular persons consistently over-estimate or under-estimate. Then, by taking these trends into account, he may be able to develop more accurate estimates for the remaining stages of the project.

12.4.2. Support requirements for estimation

Supporting a manager in the estimation activity involves providing him with the opportunity to use an estimation technique in any preferred way of estimation he might have. Here, we are interested in providing support for the project planning and re-planning phases, while inheriting estimates made in the pre-project establishment phase. The making of these initial estimates occurs prior to the establishment of the project manager's role, and thus lie outside his responsibility. Therefore, in the following, we will not be concerned with how to support their production.

From the discussion of the estimation process in section 12.4.1, we can infer that the support technique should enable the project manager to estimate the ***cost, effort*** or ***duration*** aspect of any task at any level within the task hierarchy currently instantiated for the project. Moreover, it should enable him to do so in any of the following ways:

- by providing a ***direct,*** best guess, estimate;
- by entering an ***external*** estimate, i.e., one which has been made by someone else (e.g., an estimate provided in the contract or provided by a team member);
- by entering a ***negotiated*** estimate, i.e., one which has been created as a result of a negotiation process (e.g., with the client);
- through the use of an ***estimation model*** which is applicable in forming estimates at a detailed, task, level;
- by a ***top-down*** method, where estimates made on a father task can be divided between its sons;
- by a ***related-task*** method where the project manager identifies another (related) task within the hierarchy as being analogous to the one for which estimates are to be made by maintaining a ***relationship factor*** (defined by the project manager according to the analogy) with the current estimate on the related task;
- through the use of a ***conversion formula*** which enables the transformation of an estimate of one aspect into an estimate of another one (e.g., transforming an effort estimate into a cost estimate).

The support technique should also be able to propagate automatically, on request, estimates made on leaf tasks ***bottom-up,*** that is, add up estimates made on son tasks to gain an estimate for the father task and, ultimately, for the project as a whole.

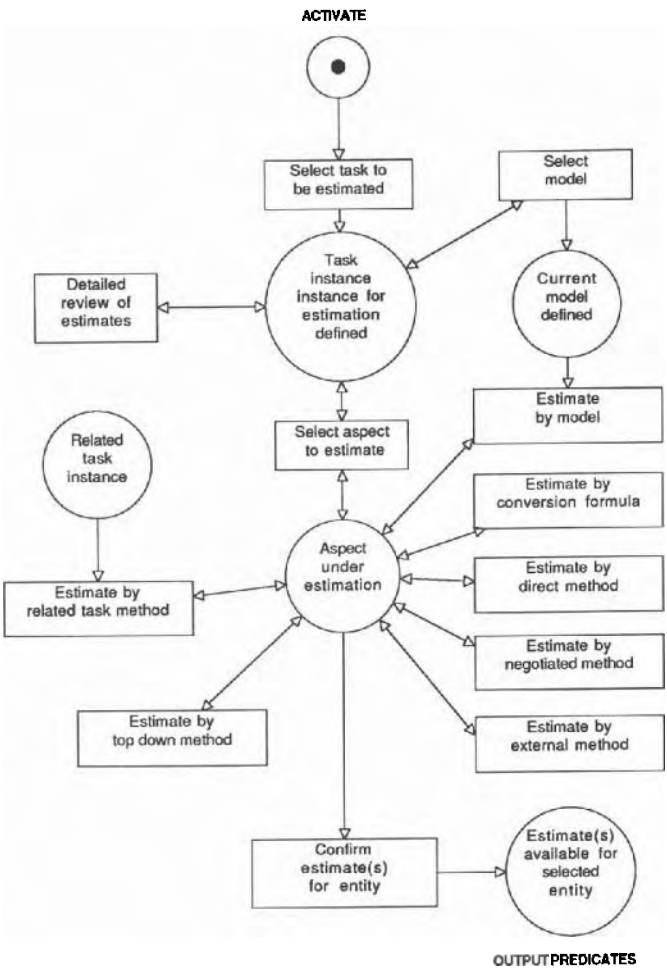
Effort, cost and duration estimates can be made for any task by any of the above methods, and, thus, ***multiple*** estimates may exist for a task. Each of these estimates should be marked with the point within the task hierarchy where it applies, the estimation method used to make it, and the name of the estimator. The last is an important piece of information since it can provide the manager with valuable information when estimates are found to be inaccurate when tested in reality. In the case where a particular individual is found consistently to provide under-estimates or over-estimates, then the manager can take corrective action by adjusting subsequent estimates received from that person.

In order to avoid confusion, only ***one*** of the multiple estimates made for each task should be marked as ***operative*** at any time, and only operative estimates should be employed in bottom-up and top-down estimation. This enables the support technique to check for, and maintain, consistency in top-down and bottom-up estimating throughout the task hierarchy. There is little point, however, in trying to maintain such consistency between non-operative multiple estimates.

12.43. A local process conceptual schema for an estimation technique

The local process conceptual schema (LPCS) for an estimation support technique can be developed and represented in an analogous way to that which we described for local process models for subPMACs in chapter 9. There are, however, some differences of emphasis and application. The LPCS for the estimation technique shown in figure 12.4 is a complete and consistent place-transition net as the technique must operate without the risk of crashing and having to be reset by the project manager†.

Figure 12.4: Local process conceptual schema for an estimation technique



† This LPCS is based on the development of the PIMS estimation support technique advanced by the ESPRIT working group PM2 (Humphreys 1989b).

Nevertheless, the **LPCS** for the estimation technique shown in figure 12.4 exhibits a high degree of indeterminacy in regard to the process of method selection. This is because the technique was designed to be supportive of, and interactive with, the project manager, rather than be prescriptive through always making estimates of its own accord. The aim is to offer, at each place of indeterminacy shown in the place-bansition net for the **LPCS,** the appropriate choice of support functions to the project manager who is using the technique. It is the responsibility of the project manager to choose exactly what kind of support function he requires from what is on offer. Thus, no additional process control component (as described in chapter 9 for local process modelling) is designed or implemented in the support technique, although advice should be offered by the technique to the project manager concerning what can be achieved through each of the available estimation functions.

The process of estimation, with the support of this technique, proceeds in the following way. The technique is activated withii the ***standard planning*** or the ***actual planning*** support environment. A view needs to be provided (at environment level) on the currently instantiated task hierarchy within either the ***activity*** perspective, or the ***time and resource perspective,*** as described in sections 8.2, 8.4 and 12.3†.

First, the project manager selects, within the presented view, the particular task for which estimates are to be made and/or reviewed and the specific aspect (effort, cost, duration) that he desires to estimate. If he is planning to use an estimation model to form his estimates, he can, at this point, select the required model from among those instantiated in the **estimation model** class in the PDCS (this class is described in section 12.5)‡.

The project manager may next select the method he wishes to use to make an estimate. If he selects the ***direct, external*** or ***negotiated*** method, the estimation technique does not provide assistance in forming the estimate, but supports the project manager through recording it as an instance of the **estimate** entity class in the PDCS, with property attributes ***method, date, estimator, estimation*** (value) and ***aspect*** (effort, cost or duration) set appropriately. Also, the ***estimates*** relationship attribute of the selected task instance is updated to reference this **estimate** instance.

If ***estimate*** by ***model*** is selected, the technique provides support by operating the (previously selected) model, eliciting from the manager the required

† Note, however, that a view provided in the time and resource perspective may show only leaf tasks and, thus, be inadequate for top-down estimation.

‡ Most of the published estimation models of the kind we reviewed in section 12.4.1 are inappropriate for use at the task level. However, a class of models which are generally useful at this level are thme described by Boehm (1984) as dart-multiplier ***submodels.*** In fact, the PIMS effort estimation support technique made use of the set of four submodels of this type which together constitute "intermediate COCOMO". In practice, each submodel in this set may be used to adjust an initial "standard" estimate made for the selected task according to how the product, computer, personnel or project-related features of the specific task vary from the "nominal" (in Boehm's terminology), i.e., standard. case.

input assessments and employing them to form the model-based estimate. In this case, the ***model used*** relationship attribute is set (in addition to those described above) for the **estimate** instance.

If the ***estimate by top down*** method is selected, the technique provides support by helping the project manager to apportion the estimation value for the selected (father) **task** instance amongst its sons and by checking that the total of the sons' estimates has the same value as that for the father.

If ***estimate by related task*** is selected, the ***related task*** relationship attribute and the ***relation factor*** property attribute are set as indicated by the project manager. The technique then provides support by automatically setting the ***estimation*** property value according to the value of the operative estimate for the related task, and automatically updating it whenever that estimate changes.

The project manager can decide at any time which of the multiple estimates made, by whatever method, for any task, may be made ***operative.*** The technique provides support in this case by maintaining consistency: it ensures that ***only one*** estimate may ***be*** operative for any aspect of any task at any time, and when this estimate changes, it automatically re-computes the bottom-up estimates which involve it.

The net representation of the estimation technique's LPCS in figure 12.4 indicates that the project manager has complete discretion and control in making, reviewing and revising multiple estimates by any of the methods outlined above. He can also move, at will, from one task instance to another and decide which aspect to estimate at any moment.

All operative estimates, and the results of automatic bottom-up estimation are displayed in the view of the task hierarchy provided in the support environment, regardless of whether the estimation technique is active at the time. The advantage of employing a single, integrated, PDCS is that this view is always kept up to date as it shows the relevant part of the current instantiation of the whole, ***integrated*** PDCS, rather than displaying isolated results produced by the activation of some particular technique. In the next section, we will examine the full set of entity classes that constitute a typical PDCS, and the relations between them which ensure its integration.

12.5. Example of a project data conceptual schema

In the following, we describe a project data conceptual schema (PDCS) which was developed from the generic core of the project data model described in chapter 10 for use within the PMSS developed on the ESPRIT I PIMS project (described in the preface to this book). The function of the PDCS is to hold all the information relating to the project work system and to the project environment, which is utilised by the PIMS PMSS's techniques. As such, this PDCS can be viewed as a subset of the generic core of the project data model, in the way we described in chapter 11. For instance, the project data model entity class **team** is omitted from the PDCS because the instantiation of its contents is

likely to be a responsibility which the project manager would wish to keep for himself, without direct support from any **PMSS** technique?.

However, the **PDCS** can also be viewed as an extension and customisation of the generic core of the project data model, specifically tailored to the data needs of the support techniques incorporated in the **PMSS.** Here, we will not give a complete account of all the extensions made in generating this **PDCS.** We will merely illustrate some of the major ones to give an idea of the types of functional customisation that may be achieved when using the generic core of the project data model as reference base in the design and development of an integrated **PMSS.**

We will also take the opportunity to adopt a different approach to the description of the **PDCS** from that which we followed in our description of the generic core of the project data model in chapter 10. There, we organised our description around the ***properties*** of the individual entity classes and provided only a subsidiary discussion of the ***relations*** between the various entity classes. Here, instead, we organise our description around the relations between the entity classes in the **PDCS,** showing how they may be viewed within the four project management perspectives we described in chapter 5. Each of these two approaches is applicable in describing either a project data model or a **PDCS** and, in effect, they complement each other. So, ideally, both approaches should be employed in order to provide the reader with a comprehensive account. This has actually been done in the case of the **PIMS PDCS** by Leclerc **(1989).** In the following, though, while we draw extensively on her description of perspectives on this **PDCS,** we do not have the space to reproduce the full lists of properties for each entity*.

The requirements for overall stability, coherence and consistency are much stronger for a **PDCS** than for a project data model, as it has to pre-empt the possibility of any confusion or ambiguity in the instantiations built by the **PMSS** techniques which operate on it. Otherwise, use of the **PDCS** might result in the inability of the **PMSS** to operate reliably on the basis of the instantiated data. Thus, the entities, and the relations between them that we view in the next four sections within each perspective on the **PDCS** constitute overlapping, but fully consistent and compatible, views on a single, unified, **PDCS.**

---

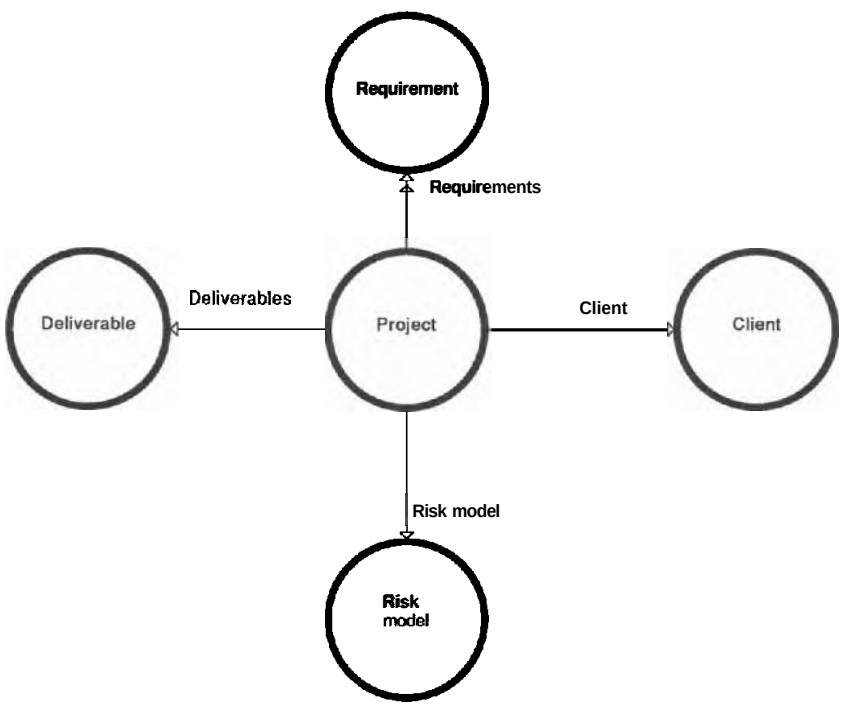
† Under the U. K. data protection act, it is very difficult to hold, legally, within any database, the kind of "soft" information about team membership described for this entity in the project data model, particularly as the project manager is likely to want to treat this information as confidential. Hence, an entity like this is best instantiated only in the memory of the project manager.

‡ Leclerc (1989) needed over 60 pages to present the full account of the PIMS PDCS. The properties of the entities given there are generally a superset of those in the generic core of the project data model, with approximately 50% more attributes added, mainly to disambiguate property information and increase the precision of the information held within the PDCS. Leclerc also provides a more detailed description of each property than we attempted in chapter 10 for the project data model.

12.5.1. Project function perspective on the PIMS PDCS

Within this perspective, the project is viewed as a unitary whole, located within an environment where requirements are set and to which deliverables are destined. The client organisation is, of course, a very important part of this environment but the project manager is not normally concerned with the internal structure of this organisation: information about addresses, telephone numbers and contacts within the client organisation usually suffice. The partial view on the PDCS provided by taking the project function perspective is shown in figure 12.5. At the centre, is the entity class **project,** which was not represented as a unitary entity in the generic core of the project data model. This entity serves as the root node in any instantiation of the PDCS, linking all that is known about, and planned for, the project through the operations of the PMSS.

Figure 12.5: PDCS entities salient in the project function perspective



The project data model entities **contract** and **standards are** subsumed under the PDCS entity class **requirement** which now groups all external requirements for the project, regardless of their source and nature. **On** the other hand, the **client** is now an entity in its own right, rather than a property of **contract** (as was described in chapter 10). This allows information about a client to be recorded independently of any particular project, and linked to a project, as and when required, simply by **instantiating** the ***client*** relation. The reason for the **change** is to facilitate the use of the PMSS in managing a number of projects (in interaction with their individual project **managers) within** an **organisation** which maintains **links** with particular clients extending over **several** projects.

As we described in section 8.2, taking a view on the project work system and the project environment within the ***project function*** **perspective** is essential in order for the project manager to be able to anticipate risks to the project through the use of a risk assessment **model†. While** such models can be very useful in understanding the risks which a project runs, the appropriate model to **be** employed in any particular context is likely to be **specific** to the contractor **organisation** and the particular application area. This is why the PDCS **risk model** entity is salient in the project function perspective, enabling the project manager to select the appropriate risk **model** instance for use within the PMSS support environment for the ***analyse risks*** PMAC.

12.52. Activity perspective on the PIMS PDCS

This perspective focuses on the interpretation of the requirements concerning the work that needs to be **carried** out in terms of **tasks** which produce products. Some of these products will be composed into the deliverables that are also viewed as salient, along with the requirements, within the ***project function*** perspective. The partial view on the PDCS provided **by taking** the ***activity*** perspective **is** shown in figure 12.6.

Especially salient in this perspective is the linkage between the **project** and task entities. This records how the work defined for the project is broken down into a task hierarchy through the operations **carried** out within the ***standard planning*** PMAC which we discussed in sections **8.1** and **10.4‡.**

At the same time, the global estimates inherited with the project brief may be refined and revised at **task** level in the manner we described in section **12.4.** Hence, the **PDCS** entity classes **estimation** and **estimation model are** salient in this perspective.

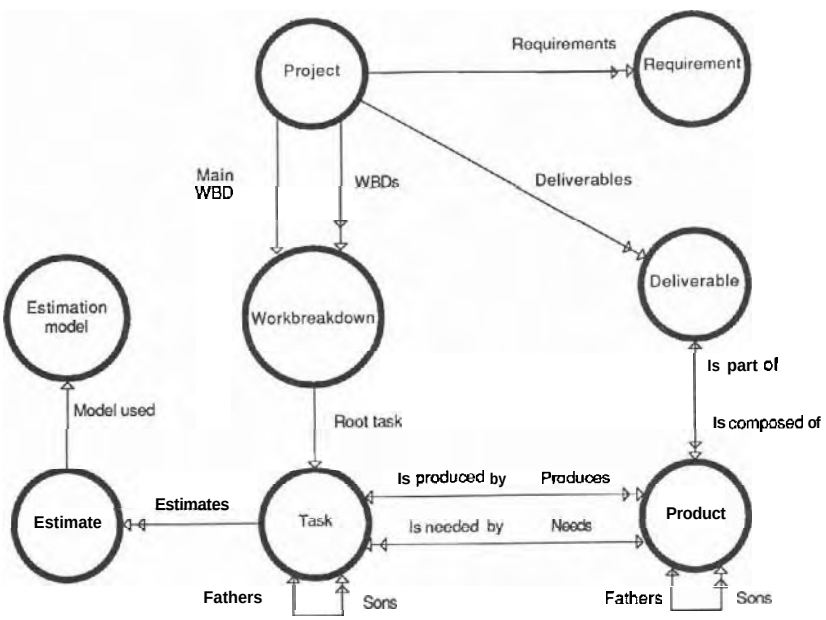
---

† **Risk assessment models were reviewed in** section **3.2.**

‡ The PIMS **PMSS** provides **support for** this activity **through** the **inclusion of** a *workbreakdown* **technique within the** *standard planning* **support environment in the** manner we **described** in **section 12.2.**

Often, the project manager will wish to develop and keep (at least, temporarily) more than one provisionally planned workbreakdown for his project. In this case, the root node of an instantiated task hierarchy is not sufficient to identify the project (as assumed in the project data model) as it only identifies the planning under a particular (and, maybe, hypothetical) workbreakdown. Thus, in the PIMS PDCS, the linkage between **project** and **task** entities is defined as being through the **workbreakdown** entity (with cardinality ***more than one).*** Each **workbreakdown** instance references a particular task hierarchy instantiation which the manager may wish, with the aid of the PMSS, to review, edit, develop or archive in the future. However, a ***single*** instance in the **workbreakdown** entity class in the PDCS identifies (via the ***main*** WBD relationship) the task hierarchy instantiated to form the basis for the actual planning and monitoring of the project at any particular time during its lifetime.

Figure 12.6: PDCS entities salient in the activity perspective



Product instances may, in the PDCS, be related to each other through ***fathers*** and ***sons*** relationships, with cardinality ***many to many.*** This does not mean that product instances need to be ordered into a strict hierarchy in the way that task instances are (where the ***father*** to ***son*** cardinality is ***one to many).*** Instead, while a product may be decomposed into several (sub)products, it may also have several fathers. Also, at any level in the task decomposition, a PDCS instantiation may indicate that a task needs or produces a product or group of products. These facilities allow the project manager to organise and decompose his products in whatever way he wishes and to link groups of products to compose deliverables as required.

12.5.3. Time and resource perspective on the PIMS PDCS

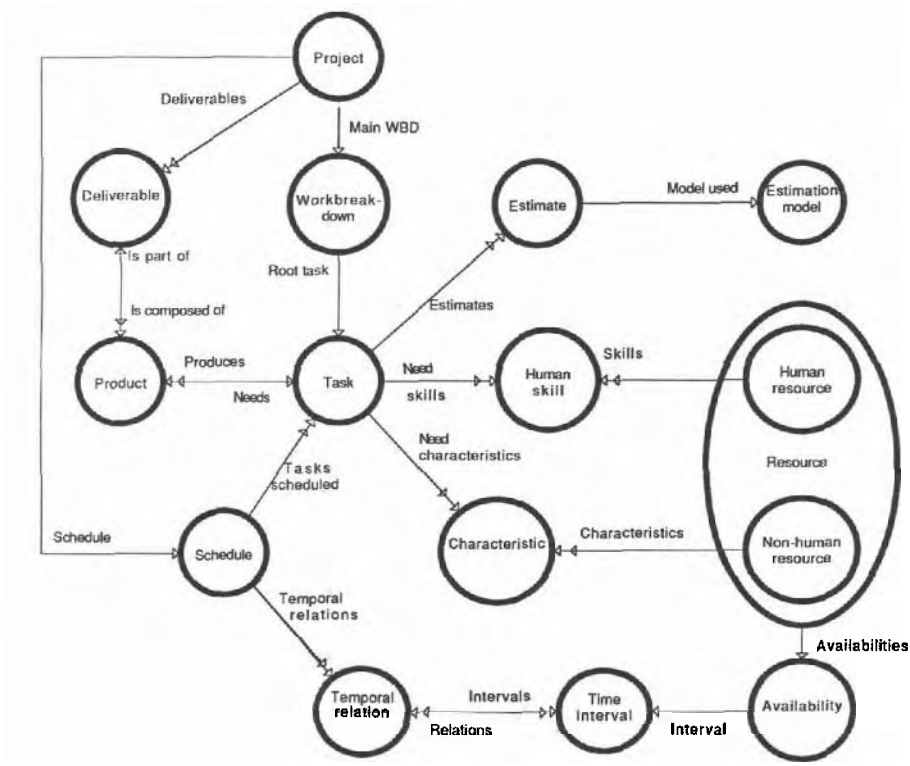
This perspective focuses on the resource aspects of the project in terms of ***time*** (in calendar and task schedule terms) and ***resources*** (human and non-human) which may be assigned to tasks according to needed skills or characteristics. The partial view on the PIMS PDCS provided by taking the ***time and resource*** perspective is shown in figure 12.7.

Deliverables remain in view in this perspective, as it is important to know ***when*** they will ***be*** delivered (as potential resources) by the project to its environment. Products also remain in view as they may be considered as resources ***produced by*** and ***needed by*** tasks. Estimates are made about resources consumed by tasks (in terms of effort, cost and duration) and so the entity classes estimate and estimation model are salient in this perspective as well.

Two additional entity classes were created in the PIMS PDCS in order to provide the possibility of instantiating the ***skills*** possessed by particular human resources, and the ***characteristics*** possessed by particular non-human resources, which are needed for carrying out particular tasks. As entities in their own right, each instance of characteristic and skill can now be described by setting the values of its properties (e.g., type of skill, level of skill), and then be related both to the task instance which needs it and to the resource instance which possesses it.

Several new entity classes had to be defined within the PIMS PDCS to increase the precision of PDCS instantiations to the level where the results of the manager's planning could be expressed in a sufficiently precise way to provide an unambiguous basis for running and monitoring the project. For instance, in the generic core of the project data model, the results of resource allocation during the ***actual planning*** activity are expressed in terms of instantiating the task ***uses*** resource relation between particular instances in these two entity classes. But this does not tell us ***when*** a particular task can expect to be able to use a particular resource, if that resource is not available to it all the time.

Figure 12.2 **PDCS** entities salient in the time and resource perspective



In the PIMS **PDCS,** the entity class availability was created to resolve this temporal uncertainty. Any resource instance may have a number of ***availabilities.*** **These** ***are*** relations to **availability** instances which actually **define** periods of unavailability, or **partial** availability. This is done by setting the properties ***percentage*** or quantity ***available*** during an ***interval***. The latter is a relationship, rather than a **property** attribute, referencing an instance of the time interval entity class.

The **reason** for defining time interval as an entity class in its own right in the PDCS (rather than just as an attribute) is that, during plan development, the boundaries of this interval which define ***when*** it occurs may not be fixed initially in time, but rather be expressed in terms of ***start constraints*** and ***finish constraints*** established within the current task schedule. Only later may it be finned up into **actual start** and end dates. As we described in section 8.1, the fundamental operation in task scheduling is to establish the precedence and

containment relations between **task** instances, such that it is possible to coordinate their needs and ***produces*** attributes through product instances. Within the PIMS PMSS, the scheduling technique, employed in both the ***standard planning*** and the ***actual planning*** support environments, operates on the basis of temporal knowledge through maintaining a network of time intervals connected through temporal constraints, and, so, the boundaries of the instances of time **interval** in the PDCS are defined in terms of start and finish date ***constraints,*** rather than just numerical **dates**†.

12.5.4. Personnel perspective on the PIMS PDCS

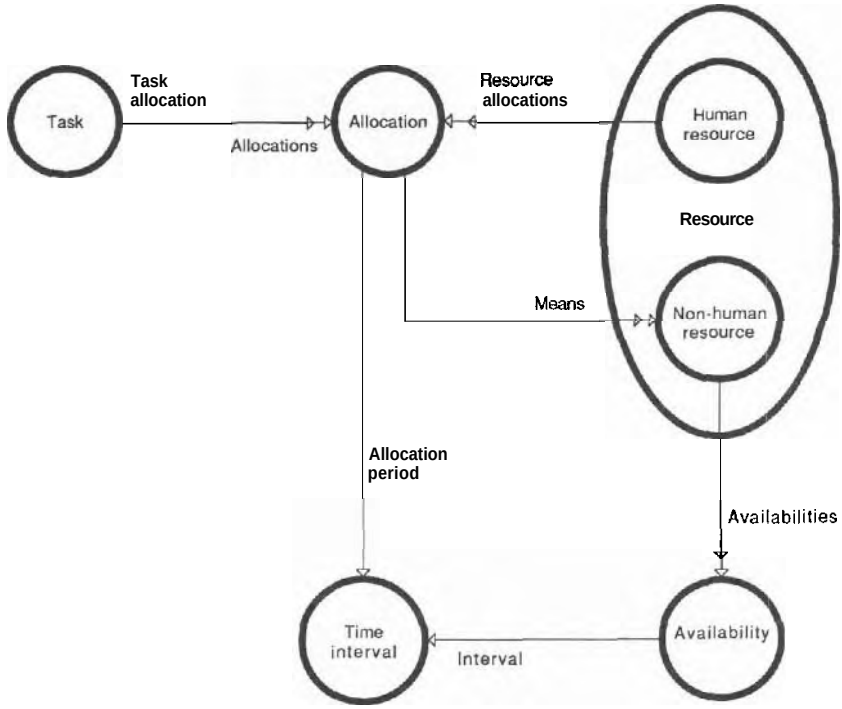
This perspective focuses on the concerns of the various human resources as they **carry** out the various roles allocated to them in the project work system. The **partial** view provided on the PDCS through taking the **personnel** perspective is shown in figure 12.8. **As** can be **seen,** it is a fairly impoverished view. This does not mean that the PMSS expects the manager to take a resmcted view within the personnel perspective. Rather, the responsibility for storing most of the knowledge salient in this perspective lies with the project manager rather than with the PMSS.

Central within this perspective is the allocation entity, which was created in the PDCS to clarify the meaning of the resource ***is used*** by **task** relation in the generic core of the project data model, from the point of view of a human resource considered as a person. This requires the provision of a clear and direct way of understanding the actual pattern of work (in terms of periods and tasks) assigned to **personnel** on the project. In fact, allocation expresses more than just the reciprocal linkage of availability (the PDCS entity created to clarify the reciprocal task ***uses*** resource relation). Instances in the allocation class refer to the allocation of a particular human resource for a particular period of time. The ***use*** **link** between tasks and human resources is now indicated by the **two-step** linkage: ***task allocation*** (indicating the task to which the particular allocation relates) and ***resource allocation*** (indicating the particular human resource allocated to it).

---

† **Allen (1983) describes a complete, orthogonal, set of relations between time intervals as having the following 13 members: "before". "after", "meets". "met by", "overlaps", "overlapped by". "during", "contains", "starts", "started by", "finishes", "finished by", "equal". This system of temporal relations, as extended by Vilain (1982) to include absolute dating and relations between time intervals and time constraints, provides the theoretical base for the PIMS scheduling technique (Hasle 1988). Absolute dating implies taking into account the company calendar and, for this purpose, the project data model entity calendar was refined into the PDCS entities calendar library and special day. These entities are not shown in figure 12.7 as they are employed by the PIMS PMSS *calendar definition* technique (discussed in section 12.3) but do not have direct relations, at the level of the PDCS, with the entities shown there.**

Figure 12.8: PDCS entities salient in the personnel perspective



Interpolating an allocation instance in this way between the relevant task and resource instances allows the setting of three atmbutes of the relevant allocation instance to record the answer to the question usually asked about to task allocations when taking the viewpoint of the ***personnel*** perspective: ***when and with what?*** These attributes are:

(1) ***allocated effort,*** a property attribute defining the amount of effort to be expected from the resource during the period of allocation;

(2) ***means,*** *a* relationship atmbute indicating the specific non-human resources that will be used by the human resource during the allocation period; and,

(3) ***allocation period,*** a relationship atmbute which identifies the instance of the time interval class describing ***when*** the resource is planned to be **allo**cated.

Defining the relationships of allocation in this manner provides a basis for the understanding of the roles played by the personnel in the project team, but the actual ***interpretation*** of those roles lies beyond the scope of the PMSS and so does not materialise in ***any*** view taken on the PDCS.

12.6. Safeguarding goals through setting watchdogs in a PDCS

In our initial discussion in section 5.2 of the achievement of project management goals through project management activities, we stressed that activating a PMAC in support of a particular goal may have side-effects on the achievement of other goals. Some of these side-effects may be negative, unintentionally subverting the achievement of a particular goal. Table 5.1 identified, for each of the typical PMACs listed, those goals whose achievement may need to be ***safeguarded*** against potentially damaging side-effects resulting from the execution of the PMAC.

This, of course, begs the question: how can we support the project manager in safeguarding his goals? In our discussion of PMACs in chapters 5 through 8, we proposed that, for any PMAC, goal safeguarding should be an issue addressed in the ***post-condition test of goal achievement*** incorporated in that PMAC. This proposal was made in recognition of the fact that project managers exhibit the general tendency of human beings to concentrate their, necessarily limited, cognitive information processing capacity on the achievement of a ***particular*** goal in the execution of any management activity (Jungermann, von Ulardt & Hausmann 1983). However, this very concentration risks the subversion of other management goals (Janis & Mann 1977). Hence, there is little point in exhorting the project manager to take care to safeguard goals at the time he is working within the PMAC at the level we described in terms of local process models for the subPMACs which constitute its body. Only when the local process operations are completed is it reasonable to expect the project manager to direct his information processing capacity, within the post-condition test, to examine the effects of the results of his operations on the achievement of ***every relevant*** management goal.

Division of labour between the project manager and the PMSS within PMACs raises the possibility that the PMSS could take over some of the responsibility for safeguarding goals. Better still, as the PMSS does not have to focus contemporaneously on exactly the same goals as does the project manager perhaps a mechanism could be devised which would watch, semi-autonomously within the PMSS, for the possible subversion of goals which need to be safeguarded as a result of the operations currently being carried out by the project manager.

This would offer a considerable increase in the efficiency of PMAC execution, as the project manager would not nearly so often have to re-activate subPMACs consequent on the failure of a post-condition test. Moreover, it would offer a considerable reduction in the amount of frustration experienced by a project manager on finding that he has to revise his work within the body of a PMAC, just at the time he is anticipating its successful conclusion.

We end this chapter by outlining the ***local process conceptual schema*** for such a mechanism within an integrated PMSS. We call it a ***watchdog*** mechanism by analogy with the role of a (real, live) watchdog when set to guard valuable property. The watchdog is semi-autonomous in that it receives its instructions on what to watch, and what to watch ***for,*** from its master who, in this

case, is the project manager. However, once instructed, the watchdog will keep watch of his own accord and, on detection of a suspicious action, will alert his master (the watchdog ***barks)*** or will act unilaterally to prevent a violation of the current status of the watched property (the watchdog ***bites).***

It is important to locate a watchdog mechanism correctly within the functional architecture of the PMSS. It is not much use locating it at technique level, as this would be equivalent, in our analogy, to having the watchdog follow the project manager around during his activities. The watchdog would then focus on much the same things as does the project manager and would have no useful independent function. Rather, we propose that watchdogs should be sited so that they watch directly the entities in the PDCS which may be the recipients of potentially goal-subverting operations†. How this is achieved is indicated in figure 12.9, which sets in context the general form of a ***local process conceptual schema*** for a watchdog mechanism. It shows how the watchdog mechanism spans and integrates aspects of the project management conceptual schema (PMCS), the project data conceptual schema (PDCS), and the PMSS's model of the needs of the project manager.

Instructing watchdogs is defined in the local process conceptual schema in terms of a ***watchdog setup*** procedure located within the PMCS. This procedure should facilitate a smooth transition from the project manager's desire to safeguard the achievement of a particular goal to the selection and parameterisation of one or more specific watchdogs which will provide the means to this end. While, ideally, it would be nice for the project manager to be able to generate watchdogs from scratch, it is more realistic to consider watchdog setting to be an activity within the ***tailor management process PMAC,*** where the aim of deciding which of the (pre-structured) watchdogs which are potentially available within the PMSS should be left ***asleep*** (i.e., turned off), and which should be set ***awake*** (i.e., activated). The awake watchdogs can themselves be tailored through parameterising them by setting ***warning*** and ***violation*** levels for the values which their property attributes may be assigned. For example, a ***spent budget*** property may ***be*** set a ***warning*** level corresponding to 90% of the cash limit on that budget, and a ***violation*** level corresponding to 101%‡.

Each watchdog is implemented as an ***after-modify*** procedure attached to the entity to be watched within the PDCS. At this level within the watchdog local process conceptual schema, the activation sequence is fully determined within the predicate-transition net shown in figure 12.9. That is, there are no indeterminate places of the type we discussed when considering the LPCS for

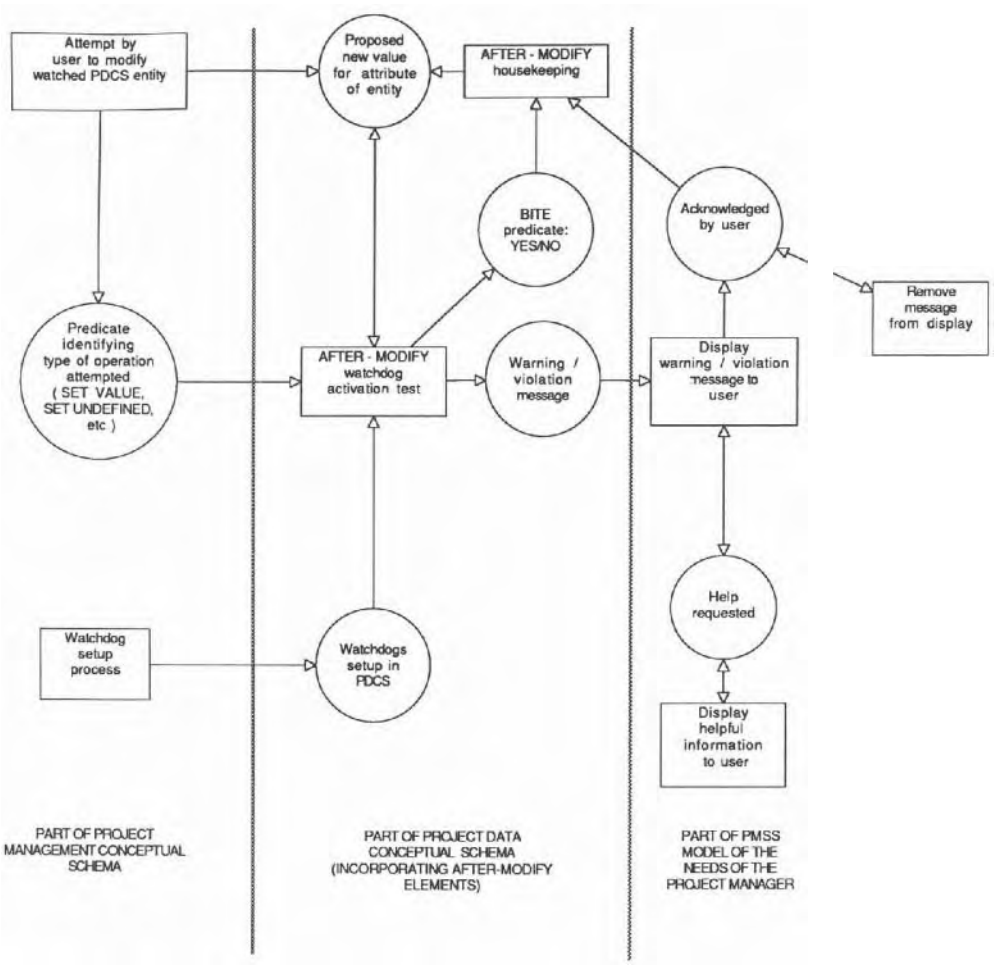
---

† These operations, emanating from PMSS techniques, may inadvertently he instigatcd by the project manager himself. Alternatively, they may result from the processing of reports from team members in the way we described in our discussion of IP2 subPMAC in section 9.4.

‡ The opportunities for setting parameters depend upon the aspect of the PDCS entity which is being watched. Cats-Baril et *al* (1988) distinguish five types of watchdogs in this respect as unconditional, self-value-conditional, other-valuetonditional, frequency-conditional and structure-conditional. Examples are given there of various types of watchdogs and their parameterisation.

an estimation support technique in section 12.3. Once again, there is no need for an additional process control component of the type we discussed in sections 9.3.2 and 9.4.2. The reason why it is not needed is, however, exactly the opposite of why it was not needed for the estimation technique. The watchdog, once instructed (i.e., set awake and parameterised), is *autonomous:* it has to control its own activities without further direction from the project manager, who is likely to be preoccupied elsewhere.

Figure 12.9: Local process conceptual schema for a watchdog mechanism



Whenever any attempt is made to modify any instance of the entity watched by the watchdog, the ***after-modify*** will check the new modification made to the ataibute to which it is attached against the parameters currently defined for the watchdog. If the results of this check indicate that the warning level defined for the value of any watched property has been exceeded through the modification, then the relevant warning is issued to the project manager. The content of the warning message, and the manner of its presentation, will depend on the PMSS's model of the needs of the project manager. This will also inform the helpful advice which is displayed, if the project manager requests it, about what might be done to overcome the detected threat to the goal which the watchdog is safeguarding. When the project manager acknowledges the warning, control is returned to the ***after-modify*** housekeeping procedure and thence to the PMSS technique which made the modification to the PDCS entity in the first place.

If the results of this check indicate that the ***violation*** level has been exceeded, then a violation rather than a warning message is issued to the project manager. Help may be offered, on request on how to overcome or remove the source of the violation. When the project manager acknowledges the message, control is passed to the ***after-modify*** housekeeping procedure, but this procedure now reinstates the value of the watched property which existed prior to the attempted modification. Control is then returned to the PMSS technique which attempted the modification of the watched PDCS entity, together with the message that its attempt was unsuccessful.

The project manager, alerted by the watchdog to the fact that his current management activity in support of the achievement of a particular goal had the effect of subverting another one, is now able (and, if required, helped) to do something about this ***immediately.*** This is much better than having to wait to discover, in a post-condition test of goal achievement, that much of his recent well-intended work is going to have to be undone, and then done again in a different way, if he is still to achieve his project management goals.

Chapter 13

Implications

The final chapter of a book typically draws together what has been described in earlier chapters and summarises in some way the major points made throughout the book. In this book, providing a concludiig summary of this kind would probably have been a misguided venture as it risks negating the book's didactic potential. We believe that this potential stems from taking the reader, step by step, through a discursive account of the project management process and its difficulties, to the development of a model for this process and to the implications of this enterprise for designing computer-based project management support systems (PMSSs) which can provide integrated support for this process to the project manager. A reader who has travelled this route with us would probably find a concluding summary to be unnecessary and irrelevant; one who has not, would probably find it meaningless.

Still, there is another level at which the contents of a book may have implications. This is the level at which the book as a whole is taken as the point of reference rather than some specific aspects of its content. There are many books available on project management and numerous software packages which offer support to the project manager in his work. What distinguishes the present book is that it has brought these two aspects together: that is, it has provided a theoretical, and practical, framework for understanding the process of project management and has developed this same framework to inform the functional design of project management support systems.

This approach has two direct implications. The first one is that a project manager can draw upon it in an attempt to formalise whichever aspects of his practice are susceptible to such formalisation. It can also, at the same time, provide to the manager a point of reference for reviewing software packages which are made available to him and evaluating their relative merits and shortcomings. The other direct implication of this approach is that it provides to a developer of a project management support system a consistent frame of reference for understanding the process he purports to support through the system he is developing. In that this book has focused explicitly on the complexity of the project management process, it may encourage PMSS developers to appreciate

this complexity and develop support systems which respect it, as and when appropriate.

In the following, we take the approach to modelling and supporting the process of project management developed in this book as our reference framework in reviewing currently available software packages which purport to support project management in various ways. We organise this review, in section 13.1, according to the support which is provided for each of the PMACs we described in chapter 7, as it is at PMAC level that we derived support environments in chapter 12. Finally, in section 13.2, we discuss the forms of support that are needed to be provided *in* general. within a PMSS, (i.e., across all the techniques it offers the users in its various PMAC support environments) in order for it to deserve the title of an *integrated* PMSS.

13.1. Support capabilities of commercial software packages

Since the mid 1980s, an increasing number of software packages have appeared on the market for supporting project management. Currently, there are more than 100 such systems. These vary widely in their prices and capabilities. The most striking aspect of the majority of these systems is that they function irrespective of the specific domain or application area for which they are used. Almost all packages are applicable to a project, independent of its nature (that is, the software can be used to manage a project that builds houses, software or dykes, or designs and implements organisational restructuring).

This wide applicability of these systems has the advantage that having access to a system that can provide support for *any* conceivable project maximises the utility of the specific software package bought as there is no need to buy another package when the application area changes. However, this generality has the disadvantage that it deprives the user of the opportunity to benefit from the *domain-specific* knowledge which could be provided to him through the package, and facilitate his activities in his particular area. In other words, the potential buyer, in his decision to buy a package, faces a choice between general applicability packages which lack domain-specific knowledge and packages which include knowledge tailored to the application area but fall short of usability in other fields of application.

Most of the available commercial project management software packages have evolved through a series of versions by which improvement is sought in satisfying the potential user. However, in most cases, improvement is usually attempted through

- providing more facilities than the previous version;
- having the ability to handle more activities or resources;
- developing a better user interface;
- achieving better response times; and so on.

In general, most design improvements tend to be directed at the technical aspects of the system (increasing performance and capability) rather than reconsidering conceptual aspects which could widen its scope and deepen the kind of support it can provide to the project manager. Thus, although more powerful commercial PMSSs are being developed, they still primarily provide support of the kind that is offered by a package of tools (e.g., for scheduling, reporting), and fail to reach the promise of an integrated system to support the project manager ***throughout*** his various activities.

Management expert systems are also being developed mainly focusing on particular techniques needed in project planning and tracking (Ernst 1988). Naturally, these address well-defined areas such as scheduling or diagnosis (Blanning 1984). However, as we pointed out in chapter 11, management problems, in general, are less well-structured and well-defined than those addressed by the techniques found in such expert systems, and require support systems characterised fundamentally by flexibility and adaptability (Vari & Vecsenyi 1984, Ford 1985, Humphreys 1989a).

Moreover, commercial project management software packages generally ignore the fact that companies tend to have preferred project management practices for the projects they take on. Often, this leads a company (if it has the resources and necessary capabilities) to develop its own software for project management in-house, shaped once and for all to reflect the project management practices of the particular company. This strategy may solve the particular company's problems of imposing project management methodologies and quality assurance standards on the work of their employees. These software systems are unlikely to be applicable under other companies' procedures, and are likely to be difficult to customise to meet different clients' requirements on what methodology to adopt. As a result, little attempt is made to market them commercially, as they would be unlikely to gain many satisfied customers.

In the following, we indicate some major trends in, and requirements for, project management software which have importance for the future. We will refrain from providing a comprehensive review of packages that currently exist in the market as it is unavoidable that such a review, given the ever-growing market, will be outdated as soon as the book is made available†.

Instead, in table 13.1, we provide a global indication of whether each of the PMACs we described in chapter 7 is supported by existing commercial software packages, and an indication of which PMACs are better supported than others, without making particular reference to specific, named, software packages.

---

† **Such reviews are most appropriately carried out by leading magazines in the field, on an occasional basis, allowing them to keep abreast with any new developments. See, for example, *Byte*, issue of November 1988, *Practical Computing*, issues of December 1988 and January 1989.**

Table 13.1: PMAC support available in current commercial software packages

PMACs	Existing Support
Actual planning	Widely supported, mainly through application of Critical Path Methods. Some packages also provide for (automatic) resource levelling. Estimation of effort and time is only supported by the more expensive packages which include estimation models based on COCOMO or Function Point Analysis. Cheaper packages simply ask the user to supply direct estimates.
Analyse risks	Not supported by the overwhelming majority of available packages. However, there are some independent risk analysis packages (not necessarily related to project management).
Anticipate future use of project	None
Confirm reqs and deliverables	None
Debriefing project team	None
Design working environment	None
Develop case studies	None
Diagnose & remedy exceptions	Only supported by a few packages.
Handle team problems	None
Handover results	None
Identify & predict discrepancies	A number of packages can issue a warning when one or more tasks are going over their deadlines or are overspending. None of these, however, can give context-dependent warnings (as described in chapter 9) and the facility to predict discrepancies is virtually absent, though some packages provide facilities for "what-if" simulations.
Negotiate changes	None
Negotiate resources	None
Negotiate working conditions	None
Project archiving	Only supported by a few packages, especially those at the high price range.
Project initialisation	By most packages in a very basic way.
Publicise project	None
Quality assurance planning	Not supported by the vast majority of the packages. Quality assurance is still very much the stepchild of project management software. This is probably due to the strongly domain-specific nature of this task. Given the bias of most of the packages towards generality, this lack of attention for the very important quality assurance and quality control aspect is not surprising.

PMACs	Existing Support
Report on progress	There are large differences between packages concerning the support they provide for reporting on progress.
Set up change control procedures	None
Set up liaison	None
Set up monitoring procedures	None of the packages cover the whole process as described in chapter 9. Only a few permit the explicit introduction of the procedures, but they do not offer any help in creating them.
Specify requirements	None
Standard planning	This centres around making a workbreakdown. As this is an activity whose quality is strongly domain-dependent, not much support is available in general. Most packages only offer the opportunity to enter tasks and create a workbreakdown tree in a more or less sophisticated way and pmv'de different "views" of the tree later on. Obviously, this is an area where domain-dependent support is desirable if the package is to function as more than a simple editor.
Tailor management process	Most of the available packages have no explicit representation of the management process other than the ability to use the techniques they provide.
Team building	None
Trigger plan	None

The main conclusion drawn by **Wood (1988),** in his review of project management software packages, is that many of the software packages have one or two attractive features, but none of them has them all. He defined the optimal package as one which would involve a graphical point-and-shoot interface, automatic resource levelling and manual editing capability using the graphics interface. Given the range of project management activities which deserve support, this seems a rather modest and resmcted set of requirements, which makes **Wood's** conclusion all the more damning.

Three important kinds of support which should be provided in any comprehensive, integrated, PMSS seem to be almost entirely lacking in current commercially available software. These are:

(1) The provision of support for the ***tailor management process*** PMAC of the kind we discussed in section 12.1. Support for this PMAC was an issue of central concern in the global design of the PIMS PMSS (Leclerc, Paris & Ribot 1990, Paris & Leclerc 1990), but is otherwise neglected in ,current commercial project management software packages.

(2) Integration of project management and software development methodology and standards. In most situations, the project manager is not free to do as he likes. Instead, he is bound by standards and procedures prescribed by the organisation he works for. These standards and procedures address two areas of concern to the manager: (i) how to ***manage*** the project, and, (ii) how to ***develop*** the product. (The latter area of concern lies in the realm of system development methods and techniques.) None of the currently available software products permits the user to bring these two aspects of software development project management together. In our opinion, this should be a priority in future PMSS developments, as it would greatly extend the scope of the support that could be offered in the ***standard planning*** environment that we discussed in sections 12.2 and 12.3.

(3) ***Multi-user*** support environments ***are*** desirable (especially in large projects), particularly where the support offered is tailored to the role within the project of each of the users. This could, for example, (i) enhance the support provided within the ***actual planning*** environment (discussed in section 12.3) by offering each team member, on demand, a customised view of the current state of his task allocations, etc., within the personnel perspective on the PDCS (described in section 12.5.4); (ii) enhance the support provided for the ***identify/predict discrepancies and exceptions*** PMAC (described in section 8.5) by enabling team members to enter reports (of the kind we discussed in section 9.4) directly into the system, thus saving the manager the effort and delay involved in having to enter all the reported data himself; (iii) enhance the support provided for the ***analyse risks*** PMAC (described in section 8.2) by enabling senior managers in the contractor organisation, as well as the project manager, to determine and review the risks being run by the project in its environment; and so on.

(4) ***Watchdog mechanisms*** (discussed in section 12.6) should be defined within the PMSS so that they can be set as required to safeguard project management goals that might otherwise get subverted through the project manager's own interactions with the PMSS.

All this said, we are still left with the general question: ***when and why does a manager need computer-based support?.*** This question cannot be answered in a simple, straightforward, way; it very much depends on the manager and his project (Wood 1988). This highlights the need for flexibility and adaptability in a fully ***customisable*** PMSS. We have laid the basis for providing for customisation through our discussions in chapter 12 of how techniques may be ***dynamically*** incorporated into support environments for PMACs which themselves are customised according to how the management model is generated and tailored, and according to how management methodology is interpreted. (All these support facilities may be provided, in our framework, to the manager in his ***management expert*** role.)

One thing, however, is certain: no PMSS, customised or otherwise, is likely to find success and acceptance in practice if the benefits which accrue to the project manager through using the system do not exceed the ***data entry***

effort costs he thereby incurs†. Otherwise, as we discussed in section 11.3, there is no motivation for the project manager to invest any of his precious time in using that PMSS, rather than no system at all.

In the long run, increased pay-off for the project manager can only come from greatly enhanced functionality on the part of the PMSS he uses. In this respect, the software packages which are currently commercially available still have a long way to go and, along the route, they will need to benefit from a major re-think of the ways they can enhance the support they deliver.

13.2. Provision of integrated support

The supporting function of an ***integrated*** project management support system goes beyond just serving as a repository of data providing structured access to project-related information upon explicit request from the project manager. Providing ***integrated*** support implies providing support with ***what*** is needed, ***when*** it is needed, ***in the form*** that is needed (or, at least, offering the possibility to the manager of defining the desired form). Of course, the question arises how could an intelligent, integrated, PMSS know how to do this?

To get the full picture of the support which may be offered by an ***integrated*** PMSS, we need to go beyond a consideration of support which may be offered by an aggregation of ***individual*** techniques. For instance, an integrated PMSS affords its user the possibility of carrying with him (and with the PMSS) information collected and lessons learned through his previous management activities on the project, to the benefit of his current management activities as the pmject progresses. In this way, an integrated PMSS can, for example, provide intelligent support for ***risk management*** throughout the lifetime of the project. We illustrated this capability in section 8.6, where figure 8.12 shows how risk advice, generated within the ***analyse risks*** suppon environment, can subsequently be employed to inform and guide risk management within the support environments for ***actual planning, set up monitoring procedures,*** and ***identi'predict discrepancies*** and ***exceptions*** PMACs, whenever these management activities are carried out.

Support to the manager in his activities is actually delivered through the dialogue between the manager and the PMSS, taken as a whole. It is important to consider the balance of the ***initiative*** in this dialogue (Jensen 1983). At one extreme, the project manager has the complete initiative, in which case the support system, in principle, serves as a repository of data providing structured access to project-related information upon explicit request from the project manager. In a mixed dialogue, the support system also offers procedural assistance: requests from the user trigger more complex operations in which the support system temporarily takes over the initiative. A completely balanced system

† These benefits will probably be greater for large projects (100 tasks or more), but this will also require more effort spent entering data into the system.

involves a division of the initiative between user and system, the system being responsive to requests from the user and, at the same time, offering spontaneous guidance to him (for example, by providing him with reminders and early warnings).

As we have illustrated throughout this book, the operations involved in carrying out PMACs vary, both within and across PMACs, from being quite concrete and closed (e.g., in monitoring progress, allocating resources) to being very abstract and open (e.g., in negotiating changes). As the level of concreteness (or abstraction) of the operations varies, so does the way in which the initiative in the dialogue should be balanced between the project manager and the PMSS. In chapter 7, we distinguished between ***boundary-spanning*** PMACs and ***internal*** PMACs. By definition, the degree of concreteness of the support that can be provided within the PMSS is different in the case of each of these two types of PMACs. For boundary-spanning PMACs, the support that can be provided by a PMSS technique is rather abstract (advising on possibilities and implications), whereas for internal PMACS the support that can profitably be provided by a PMSS centres on more concrete issues (tracking, reporting, sensitivity analysis), although appropriate advice is often welcome too.

Needless to say, it is inappropriate for a PMSS to operate in a concrete and closed way in a dialogue with a manager carrying out operations within a PMAC which requires abstract and open thinking, and vice versa. While the distinction between boundary-spanning and internal PMACs is important here, it is not, on its own, a sufficient guide to ensure that the dialogue between the PMSS and the project manager is properly balanced. In fact, a comprehensive dialogue must be balanced at no less than five levels of (increasing) abstraction, typified by monitoring, simulation, coordination, re-organisation and negotiation activities, respectively (Berkeley, Fernstrom & Humphreys 1987, Humphreys 1990).

For instance, in attaining the goal to ***maintain the relationship between plan and reality,*** for effective **monitoring,** the project plan must be treated as fixed since assessments of slippage, lateness of deliverables, etc., must be made against the provisions of the plan currently in operation for the project. It is inappropriate at this level to attempt to adjust the project plan in such a way that slippage disappears. Here, it is important that the manager understands the capabilities (and limitations) of the PMSS in providing or checking assessments based on project monitoring (the part of the PMAC for which the system is responsible). These will involve the PMSS having capabilities for acquiring, storing and accessing the relevant data in its PDCS. The PMSS should be able to report the appropriate set of data in response to a manager's command, formatted in a way that the manager can understand (e.g., reporting on slippage, or closeness to the critical path rather than just providing task progress data). In supporting the attainment of this project management goal, the PMSS needs to provide information about what ***is,*** rather than about what ***could be,*** as the focus is on the relation between a fixed plan and the immediate reality.

Exploring what "could be" involves **simulation,** i.e., the ability to explore "what-if" situations within the current project plan, the structure of which is

treated as fixed. This temporary imposition of fixed structure is necessary for the support system in order to be able to perform a ***sensitivity analysis,*** indicating the other points in the plan where changes occur or problems arise (and to what extent they do so) as a result of a specific change considered by the manager. Successive changes of values by the project manager, with the support of feedback and guidance from the associated sensitivity analyses, can provide for effective exploration of possibilities under consideration.

Knowing which changes to consider in the simulation depends upon the diagnosis of the problem to be investigated within it. The initiation of this diagnosis usually lies with the project manager rather than the support system, as it is the manager's responsibility to take action to remedy the problems which may arise. Conversely, effective problem diagnosis means identifying **(i)** those particular aspects of the project where monitoring has identified primary (rather than secondary) symptoms, and, (ii) those aspects of the project plan which may be affected as the deviant process which led to the initial symptoms continues. Moreover, in the simulation, these aspects should be investigated both within the current instantiation of the project data model (thus setting the effect of no managerial intervention as a base line) and within the hypothetical structure of the project work system (as conditionally instantiated within the PDCS) that would result from the manager taking the action he is considering to alleviate the problem.

Thus, managerial action in the real world here implies ***acting on the structure*** of the project work system itself. This may involve a direct attempt by the manager to improve the coordination of its parts, or to re-organise it. Or it may involve negotiating with third parties in the project environment who will be affected by, or be involved in, or take over the manager's proposals to handle the problem

Effective **coordination** of all the activities within the project boundary is a major objective for the project manager in creating and maintaining the project plan. Within a balanced dialogue, a support system, operating at this level, should be able to ***police the coherence of the structure*** of the project plan as it is being built. The system can then advise on gaps, and inconsistencies, and warn when a proposed re-organisation of one part of the structure may require much subsequent re-planning to restore overall coherence.

In re-planning, the manager focuses on **re-organisation** of the project work system. Alternative planning structures (as conditionally instantiated in the PDCS) may be compared. The support system can help here by ***advising on the implications*** of particular alternative plans **(e.g.,** on potential bottlenecks in a schedule, where slippage could produce wide-ranging problems, or on potential over-reliance on the capabilities of a specific resource within a particular plan). This facilitates the comparative evaluation of plans, and also indicates where the manager might best direct his efforts in trying a improve a particular plan.

Where re-organisation has potential repercussions across the project boundary affecting other stakeholders in the project environment, the project manager will predominantly be involved in **negotiation** activities. Effective

support for these activities is likely to be ***active*** in mode rather than responsive: that is, ***advising on possibilities.*** However, such advice should be offered as provisional in the dialogue with the manager, as the support system, like the project manager, is likely only to have ***partially structured*** information available about the client organisation and the wider domain of the contractor organisation.

At the end of the day, the precise type of support and style of support is best developed as the **PMSS** is customised or fine-tuned to the wishes of a potential user through actually interacting with that user. We have argued that the pre-requisites for the development of a **PMSS** which can successfully achieve this degree of integration with the needs of the project manager, as well as within itself, are:

- a clear understanding of the ***project management process*** that is to be supported;
- the generation of a ***project management model*** that provides the basis for ***both*** (i) guiding and interpreting the activities carried out in this process by the project manager and the **PMSS** working together, ***and*** (ii) deriving the functional specifications of the support techniques to be provided, and their integration within the **PMSS** in terms of both process and data;
- a reluctance to insist that the project manager, when using the **PMSS,** should fit in with the aims of its developer; and,
- a respect of the user's freedom to define what ***he*** wants from the **PMSS** which, in turn, should be customisable to meet this definition.

It is easy, of course, to pay lip service to these ideals. In this book, though, we have tried to go further through providing process modelling ideas, concepts, techniques, together with examples which can help to close the gap between these ideals and practical reality, both in **PMSS** design and development, and in project management itself.

References

Ackoff, R. L. (1967) Management Misinformation Systems. *Management Science 14,* No *4*

Albrecht, A. J. (1979) Measuring application development productivity. In: *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium.* Guide Int. Corp., Chicago

Albrecht, A. J. & Gaffney, J. E. (1983) Software function, source lines of code and development effort prediction: a software science validation. *IEEE Transactions on Software Engineering 9,* 6,639-648

Albrecht, K., Jensen, K., & Shapiro, R. (1989) *Design/CPN: a tool package supporting rhe use of colored Petri nets.* MetaSoftware Corporation, 120, Cambridge Park Drive, Cambridge, Mass. 02140

Allen, J. F. (1983) Maintaining knowledge about temporal intervals. *Communications of the ACM 26,* No 11

Argyris, C. (1977) Double-loop learning in organizations. *Harvard Business Review 55* 115-125

Baker, F. T. (1972) Chief programmer team management of production programming. *IBM Systems Journal 1* 57-73

Baron, R. (1986) *Behavior in organizations.* 2nd ed. Allyn & Bacon, Newton, Mass.

Bartol, K. (1977) Building synergistic EDP teams. In: *Proceedings of the 15th ACM SIG-CPR Conference,* pp. 18-30

Beach, L. R. & Mitchell, T. R. (1987) Image theory: principles, goals and plans in decision making. *Acta Psychologica* 66 201-220

Benge, E. J. (1976) *Elements of modern management.* AMACOM, American Management Association, New York

Bennett, J. L. (1983) Analysis and design of the user interface for Decision Support Systems. In: Bennett, J. L. (ed.) *Building Decision Support Systems.* Addison-Wesley, Reading, Mass.

Bergen, S. A. (1986) *Project management.* Blackwell, Oxford

Berkeley, D., Fernstrom, C., & Hurnphreys, P. C. (1987) Supporting the process of software project management In: *Proceedings of International Society for General Systems Research Conference of Problems of Constancy and Change,* Budapest.

Berkeley, D., Humphreys, P. C., & Thomas, R. T. (1990) Project risk action management. *Construction Management & Economics* (in press)

Blacker, F. & Shimmin, S. (1984) *Applying psychology in organizations.* Methuen, London

Blake, R. & Mouton, J. (1964) *The managerial grid.* Gulf, Houston

Blanning, R. W. (1984) Management applications of Expert Systems. *Information and Management 7* 311-316

Boar, B. H. (1984) *Application prototyping: a requirements definition strategy.* Wiley & Sons, New York

Boehm, B. (1981) *Software Engineering Economics.* Prentice Hall, Englewood Cliffs, New Jersey

Boehm, B. (1984) Software Engineering Economics. *IEEE Transactions on Software Engineering 10* 4-21

Bowers, D. G. & Seashore, S. E. (1966) Predicting organizational effectiveness with a four-factor theory of leadership. *Administrative Science Quarterly* pp. 238-263

Brech, E. F. L. (1967) *Management: its nature and significance.* 4th ed. Pitman & Sons, London

British Institute of Management (1985) *Managing new patterns of work.* British Institute of Management, London

Brodie, M. L. (1984) On the development of data models. In: Brodie, M. L., Myopoulos, J. & Schmidt, J. W. (eds) *On conceptual modelling.* Springer-Verlag, New York

Brooks, F. P., Jr (1982) *The mythical man-month: essays on software engineering.* 2nd ed. Addison Wesley, Reading, Mass.

Bunyard, J. M. & Coward, J. M. (1982) Today's risks in software development - can they be significantly reduced? *Concepts: Journal of Defense Systems Acquisition 5,* 4, pp. 73-94

Bums, R. N. & Dennis, A. R. (1985) Selecting the appropriate applications development strategy. *Data Base* 16, 1, pp. 19-23

Carnap, R. (1938) Foundations of Logic and Mathematics. *Foundations of the Unify of Science 1, 3,* pp. 1-71. University of Chicago Press, Chicago

Cash, J. I., Jr, McFarlan, F. W., & McKeeney, J. L. (1983) *Corporate Information System management: text and cases.* Irvin, Homewood, Ill.

Cats-Baril, W. & Wanless, C. (1988) *Global design: external risk (PIMS#2).* Doc. LSE-T4-GD-001, PIMS Consortium, London

Cats-Bail, W., Humphreys, P. C., & Wanless, C. (1988) *External specifications: external risk (PIMS#2).* Doc. LSE-T4-SP-002, PIMS Consortium, London

Cats-Baril, W., Humphreys, P. C., Poulymenakou, A., & Wanless, C. (1988) *Watchdog mechanisms in PIMS#2.* Doc. LSE-T22-SP-001, PIMS Consortium, London

Checkland, P. (1981) *System thinking, systems practice.* Wiley & Sons, Chichester

Chen, P. (1976) The entity-relationship model: towards a unified view of data. *ACM Transactions on Data Base System 1* 9-36

Chen, P. **(1980)** *Entity-relationship approach to systems analysis and design.* North Holland, Amsterdam

Cheney, P. H. **(1984)** Effects of individual characteristics, organizational factors and task characteristics on computer programmer productivity and job satisfaction. *Information and Management 7* **209-214**

Cherlin, M. **(1984)** Marriage: DP Style. In: Brill, A. E. **(ed.)** *Techniques of EDP project management.* **Yourdon** Press, New York

Cleland, D. I. **(1985)** A strategy for ongoing project evaluation. *Project Management Journal,* Special Summer Issue, pp. **11-17**

Cleland, D. I. & King, W. R. **(1972)** *Management: a systems approach.* McGraw-Hill, New York

Cleland, D. I. & King, W. R. **(1983)** *Systems **analysis** and project management.* **McGraw-Hill,** New York

Conte, S. D., **Dunsmore,** D. E. & Shen, V. Y. **(1986)** *Software Engineering Metrics and Models.* **Benjamin/Cummings, Menlo** Park, Ca.

Cooper, D. F. & Chapman, C. B. **(1987)** *Risk analysis for large projects: models, **methods** and cases.* Wiley & Sons, Chichester

Cooper, J. **(1984)** Software development management planning. *IEEE Transactions on **Software** Engineering 10* **22-26**

Cori, K. A. **(1985)** Fundamentals of master **scheduling** for the project manager. *Project Management **Journal,*** June, pp. **78-89**

Couger, J. D. **& Zawacki,** R. **(1978)** What motivates DP professionals? *Datamation 24* **116-123**

Cowderoy, A. J. C. & Jenkins, J. **O. (1988)** Cost-estimation by analogy as good management practice. **In:** *Proceedings of the 2nd IEEIBCS conference on Software Engineering,* Liverpool

DeMarco, T. **(1979)** *Structured analysis and systems **specification**.* **Prentice-**Hall, **Englewood** Cliffs, New Jersey

DeMarco, T. **(1982)** *Controlling software projects - management measurement and estimation.* **Yourdon** Press, New York

Druding, F. **(1984)** Looking for the right pond. *Datamation* 30 **104-110**

Durchholz, R. (1984) Konzeptionelles Schema. ***Informatik-Spectrum*** 7 **245-247**

Dyer, W. G. **(1987)** ***Team** building: issues and alternatives. 2nd* ed. **Addison-**Wesley, Reading, Mass.

Endres, A. **(1975)** An analysis of errors and their causes in system programs. ***IEEE Transactions** on **Software** Engineering 1, 2,* pp. **140-149**

Ernst, C. J. **(1988)** Management expert systems. In: **Ernst,** C. J. **(ed.)** *Management **expert** systems.* Addison-Wesley, Reading, Mass.

Fagan, M. E. **(1986)** Advances in software inspections. *IEEE Transacrions on Software Engineering 12* **744-751**

Fairley, R. **(1985)** ***Software** engineering concepts.* McGraw-Hill, New York

Farbman, D. M. (1980) Myths that miss. ***Datamation*** 26, 11, pp. 109-111

Ferratt, T. W. & Starke, F. A. (1978) Avoiding systems mismanagement. ***Journal of Systems Management 29,*** 7, 6-8

Ferrentino, A. B. (1981) Making software development estimates "good. ***Datamation,*** September, pp. 179-181

Fiedler, F. F. (1967) A ***theory of leadership effectiveness***. McGraw-Hill, New York

Fiedler, F. F. & Chemers, M. M. (1974) ***Leadership and effective management.*** Scott Foresman, Glenview, Ill.

Fife, D. W. (1988) How to know a well-organized project when you find one. In: Thayer, R. H. (ed.) ***Tutorial: Software engineering project management.*** Computer Society Press of the IEEE, Washington, D.C.

Ford, F. N. (1985) Decision Support Systems and Expert Systems: a comparison. ***Information and Management*** 8 21-26

Freedman, D., Gause, D., & Weinberg, G. (1984) Organizing and training for a new software development project: that big first step. In: Brill, A. E. (ed.) ***Techniques of EDP project management.*** Yourdon Press, New York

Genrich, H. J. (1986) Predicate/Transition nets. In: Brauer, W., Reisig, W., & Rozenberg, G. (eds) ***Petri nets: central models and their properties***. Lecture notes in Computer Science, Vol. 254. Springer-Verlag, Berlin, pp. 207-247

Glahn, J. & Borg, L. (1988) Decision making before go - support by risk management. In: ***Proceedings of the 9th INTERNET World Congress on project management,*** Glasgow

Gordon, R. L. & Lamb, J. C. (1984) A close look at Brooks' Law. In: Brill, A. E. (ed.) ***Techniques of*** EDP ***project management***. Yourdon Press, New York

Gregson, S. & Livesey, F. (1983) ***Management and the organization.*** Heineman, London

Gustafson, H. W. (1984) Implementing work system change. In: Hendrick, H. W. & Brown, O., Jr (eds) ***Human factors in organizational design and management***. Elsevier Science Publishers (North Holland), Amsterdam

Hackman, J. R & Oldham, G. R. (1980) ***Work redesign.*** Addison-Wesley, Reading, Mass.

Hales, K. A. (1979) Software management lessons learned - the hard way. In: Reifer, D. J. (ed.) ***Tutorial: Software management.*** The Computer Society Press of the IEEE, Washington, D.C.

Handy, C. B. (1981) ***Understanding organizations.*** Penguin, Harmondsworth

Hasle, G. (1988) ***Global design for the schedule functionality.*** Doc. SI-T2-GD-001, PIMS Consomum, Oslo

Helmer, O. (1966) ***Social technology.*** Basic Books, New York

Herbert, T. T. (1976) *Dimensions of organizational behavior*. MacMillan, New York

Hersey, P. & Blanchard, K. H. (1977) *Management of organizational behavior*. Prentice-Hall, New York

Herzberg, F., Mauser, B., & Snyderman, B. (1959) *The motivation to work*. Wiley, New York

Howes, N. R. (1984) Managing software development projects for maximum productivity. IEEE *Transactions on Software Engineering 10* 27-34

Howes, N. R. (1988) On using the user's manual as the requirements specification. In: Thayer, R. H. (ed.) *Tutorial: Software engineering project management*. Computer Society Press of the IEEE, Washington, D.C.

Huber, P., Jensen, K., & Shapiro, R. M. S. (1989) Hierarchies in coloured Petri nets. In: *Proceedings of the 10th International Conference on Application and theory of Petri nets*, Bonn

Humphreys, P. C. (1989a) Intelligence in decision support: A process model. In: Doukakis, G. I., Land F., & Miller, G. (eds) *Knowledge based management support systems*. Ellis Horwood, Chichester

Humphreys, P. C. (1989b) Estimation. In: PM2 working group (eds), *Consolidating our project management models*. Commission for the European Communities, Brussels

Humphreys, P. C. (1990) Risk analysis tools and techniques for project management. *Decision Support Systems Journal* (in press)

Humphreys, P. C. & Berkeley, D. (1987) Organizational knowledge for supporting decision making. In: McLean, E. & Sol, H. G. (eds) *Decision support systems: a decade in perspective*. North Holland, Amsterdam

International Standards Organization (1982) *Concepts and terminology for the conceptual schema and the information base*. Public. No ISO/TL47/SC5-N695

Janis, I. L. & Mann, L. (1977) *Decision making*. Free Press, New York

Jaques, E. (1976) *The general theory of bureaucracy*. Heineman, London

Jenkins, A. M. & Wetherbe, J. C. (1984) Empirical investigation of systems development practices and results. *Information and Management 7* 73-82

Jenks, J. M. & Kelly, J. M. (1985) *Don't* do. *Delegate! the secret power of successful managers*. Kogan Page, New York

Jensen, K. (1986) Coloured Pem nets. In: Brauer, W., Reisig, W., & Rozenberg, G. (eds) *Petri nets: central models and their prope es*. Lecture notes in Computer Science, Vol. 254. Springer-Verlag, Berlin, pp. 248-299

Jensen, S. (1983) Software and user satisfaction. In: Otway, H. J. & Peltu, M. (eds) New *office technology: human and organizational aspects*. Pinter, London, pp. 68-85

Johnson-Laird, P. N. (1983) *Mental models*. Cambridge University Press, Cambridge

Jones, H. W. (1983) Custom Software: you need precise contracts. *Information Systems* **30** 86-88

Jones, T. C. (1978) Measuring programming quality and productivity. *IBM Systems Journal* 17 39-63

Jungermann, H., von Ulardt, I., & Hausmann, L. (1983) The role of the goal for generating actions. In: Humphreys, P. C., Svenson, O., & Vari, A. (eds) *Analysing and aiding decision processes.* North Holland, Amsterdam

Kahn R. L., Wolfe, D. M., Quinn, R. P., Snoek, J. D., & Rosenthal, R. A. (1964) *Organizational stress: studies on role conflict and ambiguity.* Wiley & Sons, New York

Kakabadse, A., Ludlow, R., & Vinnimmbe, S. (1988) *Working* in *organizations.* Penguin, Harmondsworth

Keen, J. S. (1987) *Managing systems development.* 2nd ed. Wiley & Sons, Chichester

Keen, P. G. W. & Garnbino, T. J. (1983) Building a decision support system: the mythical man-month revisited. In: Bennett J. L. (ed.) *Building decision support systems.* Addison-Wesley, Reading, Mass. pp. 133-172

Keen, P. G. W. & Gersch, E. M. (1984) The politics of software systems design. In: Brill, A. E. (ed.) *Techniques of EDP project management.* Yourdon Press, New York

Keider, S. P. (1979) Why projects fail. In: Reifer, D. J. (ed.) *Tutorial: Software management.* The Computer Society Press of the IEEE, Washington, D.C.

Kelly, M. (1988) Pulling the strings. *Practical Computing,* December, pp. 42-44

Kugel, H. (1984) The three faces of leadership. *Datamation* **30** pp. 250-256

Landry, M., Pascot, D., & Briolat, D. (1985) Can DSS evolve without changing any view of the concept of "problem"? *Decision Support Systems Journal* **1** 25-36

Larichev, O. I. (1984) Psychological validation of decision methods. *Journal of Applied Systems Analysis 11* 37-46

Larichev, O. I., Moshkovich, H. M., & Rebrik, S. B. (1988) Systematic research into human behaviour in multiatmbute object classification problems. *Acta Psychologica* 68 171-182

Lasden, M. (1984) Career vs family: are you being pulled apart? In: Brill, A. E. (ed.) *Techniques of EDP project management.* Yourdon Press, New York

Leavitt, H. J., Dill, W. R., & Eyring, H. B. (1973) *The organizational world.* Harcourt Brace Jovanovich, New York

Leclerc, A. (1989) *PIMS status conceptual schema presentation.* Doc. CSI-T21-SP, PIMS Consortium, Grenoble

Leclerc, A., Paris, J., & Ribot, D. (1990) PIMS: An integrated environment for supporting project managers. *Techniques of Sciences Informatics 9,* No 2

Lehman, J. H. (1979) How software projects are really managed. *Datamation* 25, 1, pp. 119-21, 123-124, 129

Levinson, H. (1981) When executives burn out. *Harvard Business Review 59* 73-82

Licker, P. S. (1985) *The art of managing software development people.* Wiley & Sons, New York

Likert, R. (1961) *New patterns of management.* McGraw-Hill, New York

Lybrook, C. W. (1982) Quality Assurance in a large commercial data processing installation. In: *Proceedings of the 1982 AFIPS National Computer Conference,* Montvale, New Jersey, pp. 415-426

McCosh, A. M. (1984) Factors common to the successful implementation of twelve decision support systems, and how they differ from three failures. *Systems, Objectives, Solutions 4* 17-28

Mantei, M. (1981) The effects of programming team structures on programming tasks. *Communications of the ACM 24,* 3, pp 106-113

Mason, R. & Mitroff, I. I. (1981) *Challenging strategic planning assumptions.* Wiley & Sons, New York

Miller, G. A., Galanter, E., & Pribram, K. H. (1960) *Plans and the structure of behavior.* Holt, Rinehart &Winston, New York

Miller, G. D. (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review* 63 81-97

Miller, W. B. (1978) Fundamentals of project management. *Journal of Systems Management 29,* 11, pp. 22-29

Minsky, M. (1977) Frame-system theory. In: Johnson-Laird, P. N. & Wason, P. C. (eds) *Thinking: Readings in Cognitive Science.* Cambridge University Press, Cambridge, pp. 355-376

Mintzberg, H. (1973) *The nature of managerial work. Harper* & Row, New York

Mintzberg, H. (1975) The manager's job: folklore and fact. *Harvard Business Review,* July-August, pp. 49-61

Mitroff, I. I. (1983) *Stakeholders of the organizational mind: toward a new view of organizational policy making.* Jossey-Bass, San Francisco

Moms, P. W. G. & Hough, G. H. (1987) *The anatomy of major projects: a study of the reality of project management.* Wiley & Sons, Chichester

Moynihan, T., McCluskey, G., & Verbruggen, R. (1989) RISKMAN1: A prototype tool for risk analysis for software project managers. In: *Proceedings of 3rd International workshop on Computer-Aided Software Engineering (CASE),* London, pp. 3-16

Mumford, E. (1983a) *Designing human systems for new technology.* Business School Press, Manchester

Mumford, E. (1983b) Successful systems design. In: Otway, H. J. & Peltu, M. (eds) New *office technology: human and organizational aspects.* Pinter, London, pp. 68-85

Nocentini, S. (1985) The planning ritual. *Datamation* 3 122-128

Paprika, Z & Kiss, I. (1985) Interactions in Decision Support Systems: division of labour in DSS. *Engineering Costs and Production Economics* 8 281-289

Paretta, R. L. & Clark, S. A. (1979) Management of software development. In: Reifer, D. J. (ed.) *Tutorial: Software management.* The Computer Society Press for the IEEE, Washington, D.C.

Paris, J. & Leclerc, A. (1990) Environments supporting the process of project management. In: Sol, H. G. & Vecsenyi, J. (eds) *Environments for supporting decision processes.* North Holland, Amsterdam

Parkin, A. (1983) *System management.* Arnold, London

Petri, C. A. (1982) *Fundamentals of the representation of discrete processes.* ISF- Report 82.04, Gesellschaft fur Mathematik und Datenverarbeitung, St Augustin, FRG

Powell, G. N. & Posner, B. Z. (1984) Excitement and commitment: keys to project success. *Project Management Journal,* December, pp. 39-46

Pumam, L. G. (1978) A general empirical solution to the Macro Software sizing and estimating problem. *IEEE Transactions on Software Engineering 4* 345-360

Raudsepp, E. (1981) Delegate your way to success. *Computer and Communications Decisions,* March, pp. 157-158 & 163-164

Reisig, W. (1985) *Petri nets: an introduction.* Springer-Verlag, New York (EACTS Monographs on theoretical Computer Science Vol. 4)

Reisig, W. (1986) Petri nets in software engineering. In: Brauer, W., Reisig, W., & Rozenberg, G. (eds) *Petri nets: applications and relationships to other models of concurrency.* LNCS 255, Springer-Verlag, Berlin

Richter, G., Humphreys, P. C, Voss, K., Berkeley, D., Genrich, H., Domke, M., Griebler, H., & Wisudha, A. (1987) *Generic Office Frame of Reference (GOFOR).* Annex to final report, ESPRIT project 56; Functional Analysis of Office Requirements. Commission for the European Communities, Brussels

Robertson, L. B. & Secor, G. A. (1986) A game plan for software management *Information Week* , October, pp. 20-30

Scacchi, W. (1984) Managing software engineering projects: a social analysis. *IEEE Transactions on Software Engineering 10* 49-59

Schank, R. C. & Abelson, R. P. (1977) *Scripts, plans, goals and understanding.* Lawrence Erlbaum Associates, Hillsdale, New Jersey

Schlumberger, M. (1986) *PIMS prototype user requirements.* Doc. CSI-C-PR-001, PIMS Consortium, Grenoble

Schwartz, J. I. (1975) Construction of software, problems and practicalities. In: ***Practical strategies for developing large software systems*** (a compilation of papers from the 1974 USC Seminar entitled "Modem techniques for the design and construction of reliable software"). Addison-Wesley, Reading, Mass.

Shaw, D. E. (1984) Managing a software emergency. In: Brill, A. E. (ed.) ***Techniques of EDP project management.*** Yourdon Press, New York

Sneed, A. M. (1989) ***Software engineering management.*** Ellis Horwood, Chichester

Stabell, C. B. (1983) A decision-oriented approach to building DSS. In: Bennett, J. L. (ed.) ***Building Decision Support System.*** Addison-Wesley, Reading, Mass., pp. 221-260

Sweet, F. G. (1986) Milestone management. ***Datamation,*** October, pp. 107-114

Tausworthe, R. C. (1980) The workbreakdown structure in software project management. ***Journal of Systems & Software 1*** 171-186

Taylor, W. J. & Watling, T. F. (1979) ***Successful project management.*** 2nd ed. Business Books, Communica-Europa, London

Telfer, P. J. (1989) Risk analysis during project definition. In: ***Proceedings of Conference on project risk in the Aerospace Industry,*** Royal Aeronautical Society, London

Thamhain, H. J. & Wilemon, D. L. (1986) Criteria for controlling projects according to plan. ***Project Management Journal,*** June, pp. 75-81

Thayer, R. H. (1988) Software engineering project management: a top down view. In: Thayer, R. H. (ed.) ***Tutorial: Software engineering project management. The*** Computer Society Press of the IEEE, Washington, D.C.

Thayer, R. H. & Lehman, J. H. (1979) Software engineering project management: a survey concerning US Aerospace industry management of software development projects. In: Reifer, D. J. (ed.) ***Tutorial: Software management.*** The Computer Society Press for the IEEE, Washington, D.C.

Thayer, R. H., Pyster, A. B. & Wood R. C. (1981) Major issues in software engineering project management ***IEEE Transactions on Software Engineering 7*** 333-342

Tsichritzis, D. & Klug, A. (1978) The ANSI/X3/SPARC DBMS framework. Report of the study group on database management systems. ***Information System*** 3 173-191

Turner, R. (1984) ***Software engineering methodology.*** Reston Publishing Co., Reston

Ullman, J. D. (1985) ***Principles of database systems.*** Computer Science Press, Rock Ville, Md

Unger, B. & Walker, S. (1977) Improving team productivity in system software development In: ***Proceedings of the 15th ACM SIG-CPR conference*** pp. 104-115

Vari, A. & Vecsenyi, J. (1984) Selecting decision support methods in organizations. ***Journal of Applied Systems Analysis 11*** 25-36

Vilain, M. B. (1982). A system for reasoning about time. In: ***Proceedings of AAAI***. Pittsburg, PA

Waldrop, J. H. (1984) Project management: have we applied all that we know? ***Information and Management 7*** 13-20

Weinberg, G. M. (1971) The ***psychology of computer programming.*** Van Nostrand, New York

Weinberg, G. M. & Freedman, D. P. (1984) Reviews, walkthroughs, and inspections. IEEE ***Transactions on Software Engineering 10*** 68-72

Weinwym, G. (1970) ***On the management of computer programming.*** Auerbach Publications, Princeton

White, J. P. (1986) ***Roles of boundary-spanning individuals in decision-making involving organizational-environment communication.*** Unpublished PhD thesis, University of London, London

White, K. B. (1988) Cognitive styles, project structure and project attributes: considerations in project team design. In: Carey, J. M. (ed.) ***Human factors in Management Information Systems.*** Ablex, Norwood, New Jersey

Wood, L. (1988) The promise of project management. ***Byte,*** November, pp. 180-192

Wynne, B. (1983) The changing roles of managers. In: Otway, H. J. & Peltu, M. (eds) New ***office technology: human and organizational aspects.*** Pinter, London, pp 138-151

Yeh, R T., Zave, P., Conn, A. P., & Cole, G. E., Jr (1984) Software requirements: new directions and perspectives. In: Vick, C. R. & Ramamoorthy, C. V. (eds) ***Handbook of software engineering.*** Van Norstrand Reinhold, New York, pp. 519-543

Glossary of key terms

The entries in this glossary comprise mainly terms which are specific to this book rather than entries which relate to its substantive area (i.e., project management). A range of definitions of the latter may be found in most textbooks and tutorials on project management (e.g., Thayer 1988). The information on each entry should not be taken as a complete and prescriptive definition but should be read together with the corresponding entry in the subject index, which will refer the reader to the contexts in which the term is defined and used within this book.

activity perspective The perspective which relates to the work that needs to be carried out within the project work system for the project to achieve its aim (i.e., to produce specified products and deliverables within the constraints identified within the project function perspective). See also ***perspective, project function perspective*** and ***project work system.***

approximate net modelling Modelling which indicates unambiguously the types of the specific structural elements in the net and the linkage between them. However, the inscriptions describing the elements may be imprecise, expressed as legends (e.g., in natural language), and no strict rules of change need be defined to regulate transitions between elements. See also ***exact*** net ***modelling.***

attribute Within entity-relationship modelling, as employed in constructing the project data model and the project data conceptual schema, an entity class may have any number of class atmbutes attached to it. Each attribute should charactense, to varying degrees and with various values, all the instances of the entity with the given class name. Attributes are divided into two types: ***property*** atmbutes, which describe intrinsic characteristics of the entity class, and ***relationship*** attributes, which define the way in which instances in the entity class may be related to particular instances of another entity class within the project data model or the project data conceptual schema. See also ***entity-relationship modelling, project data conceptual schema*** and ***project data model.***

blackbox approach to risk analysis An approach to risk analysis which does not rely on a structured view of the inside workings of the project itself but, instead, involves identifying risk drivers which could affect the success of the project, and assessing the likelihood of the occurrence and the magnitude of their individual or combined impact. See also ***risk analysis*** and ***whitebox approach to risk analysis.***

boundary-spanning project management activity A project management activity which involves transactions of the project manager with either the project work system or the project environment. See also ***internal project management activity*** and ***project management activity.***

conceptual schema See ***local process conceptual schema, project data conceptual schema*** and ***project management conceptual schema.***

coloured Petri net A net constructed by means of an extension to the standard predicate-transition modelling technique where, when modelling a **PMAC,** predicates constitute ***expressions*** when they act as pre-conditions for the **PMAC,** and constitute ***functions*** when they ***are*** incorporated within the internal structure of the **PMAC** itself. See also ***pre-condition, predicate-wansition net modelling*** and ***project management activity.***

entity-relationship modelling In entity-relationship modelling, the object system, as represented in the model, is split into three distinct types of elements: entities, property attributes (sometimes described simply as ***properties*** or ***attributes)*** and relationship attributes (sometimes described simply as ***relations).*** What precisely constitutes an entity or an attribute is rather arbitrary, and the definitions **vary** considerably across different approaches to entity-relationship modelling. In the approach taken in this book in building a project data model, the entities in the model describe states of objects that may be viewed in the perspectives the manager can take on the project work system and the project environment. An entity class is a pre-structured component of the model, describing a particular type of object **(e.g.,** ***task*** or ***deliverable)*** about which state information may be maintained. An entity class may have any number of class attributes attached to it. Each attribute should characterise, to varying degrees with various values, all the instances of the entity in the project data model (or project data conceptual schema) with the given class name. See also ***attribute, generic model, instantiation, object system, project data conceptual schema, project data model,*** and ***project work system.***

environment See ***project environment*** and ***support environment.***

exact net modelling As for approximate net modelling except that (i) inscriptions must have precise denotation and predication, (ii) strict rules of change must be specified, **(iii)** structural elements in the model must be coherently linked, and, (iv) the full rule set (predicates, rules of change) must be coherent. See also ***approximate net modelling*** and ***net modelling.***

formal modelling Modelling through the use of formal means of representation. A formal model may be either approximate or exact according to the definitions given for these terms. See also ***approximate net modelling*** and ***exact net modelling.***

generative model A collection of well-defined concepts which help one consider and express the static and dynamic properties and constraints for a range of uses of the model within the domain of the object system. Each use will involve the generation of a particular instantiation of a representation of the model. While each representation is, of necessity, temporarily closed (so that it can be represented), the union of the set of representations which could possibly be generated through the use of the modelling concepts remains open. See also ***model*** and ***instantiation.***

generic core of project data model This consists of entities (and their attributes) which are considered generally necessary and to be specifiable in a stable way across many application of the project data model. It forms the basis on which further customisation, refinement and extension of the project data model can be carried out in specific applications. See also ***generic model, project data model*** and ***refinement.***

generic model In a generic model, the ***core*** of the representation of the object system is pre-structured. This is usually achieved through grouping model elements that may be instantiated into classes defined in terms of pre-structured entity types. Then, each instance that is made of the modelled entity described by the class will necessarily share the same generic specification of its attributes, relationships and constraints. A particular instantiation of the model can thus be made by assembling instances drawn from the classes describing the entities to be represented withii the instantiation. As the relationships between the various entities are pre-specified (as part of their class definitions), there is no need to employ modelling concepts to generate the model structure each time: the structure is automatically given as the instantiation is made. See also ***entity-relationship modelling, generic core of the project data model, instantiation*** and ***model.***

goal See ***project management goals.***

inference structure A structure comprising reasoning activities, reasoning entities and their linkages within a predicate-transition net. See also ***predicate reasoning entity, reasoning activity, reasoning process control component*** and ***virtual reasoning entity.***

instantiation A structured set of instances of elements of a model (or a conceptual schema) which provides a specific representation of part of that model. Thus, an instantiation of the project data model will describe states of the project work system and the project environment in a particular context. Similarly, an instantiation of the project management model will describe specific management activities which may be carried out in a particular context. See also ***entity-relationship modelling, project data model, project management activiry*** and ***project management model.***

internal project management **activity** A project management activity the success of which is not dependent on successful transactions of the project manager with other systems such as the project work system or the project environment. See also ***boundary-spanning project management activity*** and ***project management activity.***

local process conceptual schema **(LPCS)** A conceptual schema which provides a detailed description of a process **localised** and implemented **within** a PMSS support technique. See also ***project management conceptual schema*** and ***project management support system.***

local process model A model which provides a detailed description of a process **localised** within a **subPMAC.** See also ***project management model*** and ***subPMAC.***

LPCS See ***local process model conceptual schema.***

model In general, the term ***model*** is used to refer to some abstraction of certain elements in the reality of the system the model is meant to represent (the object system of modelling) and a representation of the relationships between them. The differences in the nature of different types of models are usually located at the degree of detail they use to address the object system, the model representation formalism they employ, and the method through which the model is arrived at. Models, as pure abstractions, are without intrinsic meaning. Meaning is achieved through interpreting the model and, thus, restricting the generality of the elements and relationships in the model through making references to objects, or classes of objects, which are familiar (or, at least, identifiable) in the domain of application of the model. These objects, and their relationships **are** thus considered to constitute the object system **being** modelled. See also ***project data model*** and ***project management model.***

modelling Use of concepts and techniques to develop a model of an object system. See also ***model, entity-relationship modelling*** and ***net modelling.***

modelling language ***A*** formal language which is used for the construction and description of a model. See also ***model*** and ***precision.***

net modelling According to the fundamental rule of net modelling (Petri 1982), a net is generated by connecting two types of elements: places and links. In the applications of net modelling described in this book, places **are** interpreted as states of the project management model being generated, and links are interpreted as the project management activities which effect transitions between these states. See also ***place-transition*** net, ***predicate-transition net modelling*** and ***coloured Petri net.***

object system The system to be modelled. See also ***model, project management system, project work system*** and ***system.***

personnel perspective The perspective which addresses the concerns of the human resources in carrying out the roles allocated to them within the project work system. See also ***perspective.***

perspective As employed in this book, the concept of ***perspective*** capitalises on the analogy from visual perception: the project manager, viewing his project, is likely, at any point in time, to take only a partial view on the full workings of the project work system and the environment within which it is situated. However, what is viewed is seen ***in perspective.*** The notion of perspective is used here (a) to identify the contents of partial views on the project work system and project environment which form the substantive basis when constructing a project data model, and, (b) as a means of providing a structured view into a project data model, or project data conceptual schema, which prioritises, in the foreground, those entities of special interest and their relations with the background context of the wider model. See also ***activity perspective, personnel perspective, project function perspective, time and resource perspective*** and ***viewpoint.***

PDCS See ***project data conceptual schema.***

Petri net See ***colowed Petri net*** and ***net modelling.***

place-transition net A net which is composed of two basic kinds of elements: places and transitions. The place in the net are linked through active elements which effect the transitions between them. In the applications of net modelling in this book, these active elements comprise PMACs and subPMACs. A place may carry pieces called tokens, which are individual objects with properties. The tokens on a place are called its markings. The simplest tokens, used in state-transition nets are merely distinguished by their presence or absence. More complex tokens, carrying information expressed in the form of predicates, are used in predicate-transition nets. See also ***colowed Petri net, net modelling, predicate-transition net modelling, project management activity*** and ***subPMAC.***

PMAC See ***project management activity.***

PMCS See ***project management conceptual schema.***

PMSS ***See project management support system.***

post-condition In predicate-transition net modelling (as employed here in modelling PMACs), predicates may be employed to determine which tokens, with what pmpemes, may mark which of the places linked outwards from the

PMAC, hence defining its post-conditions. See also ***predicate-transition net modelling, project management activity,*** and ***project management goals.***

precision A ***low precision*** modelling formalism is usually employed in modelling which tries to capture the form in which the elements in the model were "intuitively thought up". This may be contrasted with ***high precision*** modelling where the resulting model will have been worked over and precised to the stage where its degree of coherence and consistency permits a formal, mathematical treatment of the relationships between the elements of the model. Increasing the degree of formality of the language used to express a model may offer the possibility for building future instantiations exhibiting greater precision than that possible with the current modelling language. However, translating a current instantiation of a model into a more formal language does not automatically increase its degree of precision. See also ***model, formal modelling*** and ***instantiation.***

PRE See ***predicate reasoning entity.***

pre-condition A condition which must be verified, through evaluation of the relevant input predicates, before a PMAC can be activated. See also ***predicate-transition net modelling*** and ***project management activity.***

predicate reasoning entity (PRE) An operand in a reasoning process (represented by predicates in the project management model) which references the requirements for, and results of, reasoning within a local process model. See also ***local process model, predicate-transition net modelling*** and ***virtual reasoning entity.***

predicate-transition net modelling In ***predicate-transition*** net modelling, as employed in this book, ***pre-conditions*** (activation conditions) for PMACs are tested as predicates on the properties of the tokens which appear on the places which are linked inwards to the PMAC. Activating the object may, in some net structures, lead to more than one (alternative) result. Here, predicates may be employed to determine which tokens, with what properties, may mark which of the places linked outwards from the PMAC, hence defining its ***post-conditions.*** More complex tokens may carry denotative information describing their enduring properties, or may carry information which reflects the history of the transitions in which the token has been involved in the net. See also ***coloured Petri net, net modelling, place-transition net, pre-condition*** and ***project management activity.***

pre-formal modelling The process of developing pre-formal models through the collection of propositions concerning phenomena relevant to the object system of interest and their organisation (in a global way) in terms of the common features they possess. See also ***model*** and ***object system.***

process control component See ***reasoning process control component.***

project data conceptual schema (PDCS) Within a PMSS, the PDCS is used to represent and store information about the states of the project work system and the project environment. The PDCS operationalises, within the PMSS, part of the full project data model (i.e., the one employed by the project manager and the PMSS, working in interaction). Although developed via the concepts of the project data model, it is specified to have a bounded and coherent structure, matching to the requirements of the operations which may be made on it by the PMSS's techniques. Therefore, the PDCS consists of an enumeration of the classes of entities that the PMSS deals with, the relationships among these entities and the constraints on their instantiation. See also ***entiry-relationship modelling, instantiation, project data model*** and ***project management support system.***

project data model A model of states of the project work system and the project environment reflecting only managerial concerns. It is defined in terms of an instantiation of an entity-relationship model. See also ***entity-relationship modelling, generic model, instantiation*** and ***project data conceptual schema.***

project environment The organisational systems with which the project work system and the project manager wansact. Important here are the client organisational system, the wider contractor organisational system and systems operated by third party stakeholders (e.g., regulatory agencies). See also ***projecr data model*** and ***system.***

project function perspective This perspective relates to the function of the project as a whole, including the product which is the end result of the project, issues related to client satisfaction with the product, the reputation of the contractor organisation concerning quality of the product, the contract and the binding conditions it reflects concerning the functional specifications for the product, delivery time, quality standards expected, available budget, etc. See also ***perspective.***

project management activity (PMAC) A management activity carried out either by the project manager or by the project manager working in interaction with a PMSS. Collectively, management activities constitute the project management system. They operate on the project work system, controlling, coordinating and re-organising its elements, and are represented in the project management model. Pre-formal or formal descriptions of Project Management ACtivities within the project management model are defined as PMACs. The basic structure of a PMAC comprises three active entities: (i) a pre-condition test of input predicates, (ii) a functional body consisting of a structure of sub-PMACs effecting operations on entities instantiated in the project data model, and, (i) a post-condition test of goal achievement. See also ***predicate-transition net modelling, project management goals, project management***

model, project management system, project management support system, project work sysrem and ***subPMAC***.

project management conceptual schema (PMCS) Within a PMSS, the PMCS operationalises part of the full project management model which guides and constrains the project manager. It consists of an enumeration of the techniques within the PMSS which, collectively, constitute its capabilities for management support and it structures the information required for the operation of these techniques. This information concerns constraints due to management methodologies, models for effort estimation and risk analysis, process control information, etc. ***See*** also ***local process conceptual schema, project management methodology, project management model*** and ***project management support system.***

project management goals These are the pmject manager's goals within the context of the role he plays in the project. They are not necessarily coincident with his individual, personal, goals. Three global goals to be achieved are to manage the project well, to transact with the project environment well, and to advance expertise. These may be further subdivided into more specific goals which constrain and shape the execution of project management activities aimed at advancing these goals. These specific goals also inform the post-condition test of goal achievement in PMAC modelling. See also ***project management activity, post-condition*** and ***watchdog mechanism.***

project management methodology A structured set of advisory and mandatory prescriptions informing the process of pmject management. It may be viewed in terms of a set of partial constraints on project management model instantiation, serving to regulate the conditions under which particular management activities may be executed. This results in shaping the project management process so that it conforms to the specific current requirements for good management practice. See also ***instantiation, project management model, pre-condition*** and ***project management activity.***

project management model A model of the project management system defined in terms of a structure of PMACs linked through input and output predicates. The structure of the project management model may be constrained according to a particular project management methodology. See also ***generarive model, predicate-transition net modelling, project management activity, project management methodology*** and ***project management system.***

project management support system (PMSS) An integrated computer-based system which is designed to provide interactive support to the project manager in the project management process. See also ***project management conceptual schema, project data conceptual schema, support environment*** and ***support technique.***

project management system The system which includes all management activities which operate on the project work system and the project environment in order to fulfil project management goals. See also ***project work system, project management model,*** and ***project management goals.***

project work system The human and technical system which represents the means by which the project will be carried out and produce the required results. It is operated on by the project management system. See also ***project data model*** and ***project management system.***

property See ***entity-relationship modelling*** and ***atm'bute.***

RAC See ***reasoning activity.***

reasoning activity (RAC) A modelled activity which describes operators in a reasoning process about the project RACs are modelled as functions which effect transitions within subPMACs in the project management model. See also ***inference structure, project management activity, project management model*** and ***local process model***.

reasoning entity See ***predicate reasoning entity*** and ***virtual reasoning entity.***

reasoning process control component The component of a local process model for reasoning knowledge which describes in what sequence or under what conditions the reasoning activities in the inference structure should be activated to reach a specific goal. See also ***local process model*** and ***inference structure.***

refinement Refinement involves decomposing a unitary element of the modelled representation of an object system to show its internal structure. Thus, ***refining*** a model refers to the process of developing a more ***fine-grained*** structure in some part of the model in order to capture more detail in subsequent instantiations of it. The process of refinement, on its own, does not increase the degree of precision of a model: it only reveals greater detail. See also ***model*** and ***precision.***

relation See ***entity-relationship modelling.***

relationship attribute See ***attribute.***

risk analysis The procedure by which project risk is assessed. Conventionally, it consists of four phases: (i) identification of reliable risk factors, (ii) measurement on those factors of the risk present in the project (i.e., development of a risk profile), (iii) evaluation of the risk profile to identify areas of managerial

concern, and (iv) risk management. See also ***blackbox approach to risk analysis, risk factor, risk management, risk profile*** and ***whitebox approach to risk analysis.***

risk driver An observable phenomenon which is likely to drive up the possibility of some risked consequence whose future occurrence depends, in part at least, on the occurrence of this phenomenon.

risk factor An observable criterion which is scaled in a way that indicates the degree of project risk. Scores on risk factors may be combined using an algorithm to indicate project risk expressed as a profile of the project's potential disbenefits in a particular domain of the organisation's functions. See also ***risk profile*** and ***risk driver.***

risk management The process of taking managerial action to restructure the project work system and/or its transactions with the project environment in such a way that the project's risk profile is improved. See also ***risk analysis.***

risk profile The vector of scores for a particular project on a set of risk factors. See also ***risk factor.***

subPMAC The functional body of a PMAC may be refined into a structure of subPMACs linked within a predicate-transition net SubPMACs comprise operations which perform transformations on instances of entity classes defined in the project data model (i.e., creating, modifying, deleting them). Thus, these operations are identified as activities transforming project data model instantiations. Each operation may be defined in terms of a ***local process*** model. See also ***instantiation, local process model, predicate-transition net modelling, project data model, project management activity*** and ***refinement.***

support environment Within a PMSS, the techniques which support the activation of any of the subPMACs constituting a PMAC for which the PMSS aims to provide support are clustered into the ***support environment*** for that PMAC and define it. Any of the clustered techniques should be available for use at any time, at the discretion of the project manager during PMAC activation. See also ***project management activity, project management support system*** and ***subPMAC.***

support technique A technique offered by a PMSS to the project manager to provide support for a management activity. See also ***project management support system*** and ***support environment.***

system A complex object seen in terms of the elements that comprise it and their interconnections. See also ***object system, project management system, project work system*** and ***project management support system.***

time and resource perspective The perspective which relates to resource aspects of the project in terms of time (in calendar terms and effort terms), and in terms of physical resources such as finance, personnel, machines and machine time, space, etc., which are instrumental to achieving what has been identified within the project function perspective. See also ***perspective*** and ***project function perspective.***

viewpoint A point from which a phenomenon is viewed either in the physical world or, as in its use in this book, in a conceptual world (i.e., a world established to scope ways of thinking about the phenomenon). See also ***perspective.***

virtual reasoning entity (VRE) An operand in a reasoning process the structure of which is created out of selected (relevant) attributes of one or more project data model entities and which is instantiated with the values of these attributes, as and when required. A VRE does not exist as an entity in the project data model itself. See also ***entity-relatiomhip modelling, inference structure, instantiation, predicate reasoning entity*** and ***project data model.***

VRE See ***virtual reasoning entity.***

watchdog mechanism A mechanism that can be implemented within a PMSS to safeguard project management goals while the project manager/user is employing the PMSS's support techniques in working towards the attainment of other management goals. See also ***project management goals, project management support system*** and ***support technique.***

whitebox approach to risk analysis An approach to risk analysis which involves building a workbreakdown structure for the project, explicitly identifying outcomes which would indicate that a task has gone wrong, and assessing their estimated probabilities and costs, and then, calculating bottom-up the total expected cost distribution. See also ***blackbox approach to risk analysis*** and ***risk analysis.***

Author index

Subject index