The Application of System Dynamics to Project Management An Integrated Methodology (SYDPIM)

By

Alexandre G. Rodrigues

Thesis submitted to the Department of Management Science in partial fulfilment of the requirements for the degree of:

Doctor of Philosophy

at the

UNIVERSITY OF STRATHCLYDE

July, 2000
Glasgow, United Kingdom

Abstract

Over the last decades, project management has become increasingly important for social progress. As projects are the vehicle to implement change, "management by projects" has become the current way of living of many organisations. Since the project management discipline first merged in the early 1920s, a wide collection of processes, tools and techniques has been developed. This traditional approach to project management has been focusing on the operational issues. However, problems of systemic nature have been emerging, where the many various interactions among human and social factors determine behaviour. The traditional approach is not aimed at addressing these issues. System Dynamics (SD) modelling has emerged with considerable success to analyse these systemic issues in social systems. A number of past applications to project management suggests that SD is effective in addressing these issues. However, it has been used in isolation from the traditional project management process. This research proposes that it is beneficial to integrate SD within this process. The author has investigated the potential distinctive roles of SD, and has developed a formal integrated methodology. Following an exhaustive review of past applications, a conceptual integrated framework was developed. This framework was refined through tentative applications within a large-scale software intensive project, for the period of 18 months. The formal integration of SD has proven beneficial. As a result, a formal System Dynamics-based Project-management Integrated Methodology (SYDPIM) was developed. SYDPIM comprises two main methods. The project management method articulates the use of a SD project model within the modified project management process, while formally linked with a PERT/CPM model. The model development method provides a structured framework to support the development and validation of SD project models. Practical applications of SYDPIM undertaken within the fieldwork project are here described. The more important future developments are identified and discussed.

Acknowledgements

Undertaking a PhD research is a challenging project. While an individual effort, the interaction and support of others is essential. In the course of this work, I have been through various different experiences, from pure academic research, lecturing, fieldwork research, to professional management consulting. Most of these experiences focused on the application of System Dynamics to Project Management. They provided me with valuable understanding for the purpose of the research. However, not least important was the support I have received from friends and many of those with whom I was fortunate having worked with. To all of them I am very grateful. There are some to whom I would like to leave an explicit word of gratitude.

I would like to thank my supervisor, Professor Terry Williams, at the Department of Management Science, University of Strathclyde, for all the patience and unconditional support. Throughout this research, I happened to be working in different parts of the world. Wherever I was, his support, prompt advice and valuable feedback, were all unconditionally provided. Also at Strathclyde, I would like to thank Professor Colin Eden for all the good working conditions and support provided by the Department. I would also like to thank him for the valuable advice he volunteered at important stages of my research.

I started this research at the University of Stirling. I would like to thank my supervisor, Professor Eric Wolstenholme for his support and for having recommended the use of System Dynamics. I would also like to thank Mr. John Bowers, my additional supervisor, for all the time he dedicated to my work and for all his thorough feedback and comments.

A considerable part of this research was undertaken in the form of fieldwork, during a large-scale software intensive project (KDCOM) at BAeSEMA Ltd. I would like to thank the company for the opportunity and support. I would like to thank Mr. Phil Merison, the KDCOM Program Manager, for having allowed this valuable opportunity to take place. I would also like to thank Mr. Graham Muggeridge, the Productivity and Life-Cycle Improvement Manager, and Mr. Steve Webb, the Applications Manager and Project Controller of KDCOM, for all their interest, opportunities provided to apply SD, and for their continued support after the fieldwork was over. I would also like to thank Mr. Steve Dowdell, the Software Manager of KDCOM, for all his support and interest.

After having completed this research, I spent some time in professional management consulting, at PA Consulting Group, Pugh-Roberts Associates practice, in Boston. The learning and exchange of ideas with many of my colleagues was most valuable. I would like to thank them for their contribution to such a practical view of System Dynamics.

A PhD project requires various skills from the researcher. One of them, is the willingness to proceed with the individual work, despite all the difficulties. The support from those who believe in our ability to succeed from the fist day on, is most valuable. I would like to thank Professor Barbosa Romero, at the University o Minho, for all the encouragement and support, and to Mr. Meira Ramos and my family in general. Also, some thanks to the Department of Information Systems at the University of Minho, for the support in the later stages of the writing-up.

A PhD is not complete until the examiners provide their final judgement. I would like to express my gratitude to Professor Colin Eden, Dean of the Strathclyde Business School, and to Professor John Morecroft, Dean of the Executive MBA Programme at the London Business School, for their constructive feedback and encouragement to improve further the quality of this thesis.

Finally, I dedicate this work to my best friend, Claudia.

This research project was jointly funded by INVOTAN (the NATO's research division in Portugal) and by Fundação Para a Ciência e Tecnologia (FCT, Portugal) / Programa PRAXIS XXI *(grant BD/2996/94).*

Table of contents

Index of Figures

Index of Tables

1. Introduction

1.1 Motivation and research background

The motivation for the research here presented was grounded on evidence that projects are critical to society's progress and that our current performance in managing and implementing them needs to be improved.

Throughout human history, projects have always been the vehicle used to implement *change*. Examples of projects in the remote past include the pyramids of Egypt, the Roman bridges, among many other similar achievements of ancient civilisations. Humanity has always changed the face of the world through projects. With no disregard for "routine endeavours", which are essential to sustain human activity, under an evolutionary perspective one could say that "the world has been made of projects". Nowadays, they still continue to be the vehicle of change. However, there is one critical difference from the past: the environment within which they are implemented is characterised by a very fast rate of change. Prosperity therefore means implementing many projects, quickly and successfully. Organisations which are not able to cope with this challenge will find it hard to survive in today's business world, while failing to contribute to social welfare.

It is assumed in this research that current project performance is in general below the desired level, and that improvements are both possible and needed. Nowadays, the successful implementation of projects needs to cope with different and more "hostile" conditions, as compared to the past. For example, the "time to market" factor can turn into a complete failure a project which was being implemented carefully and thoughtfully, just because the "window of opportunity" has suddenly moved away. Another good example is the fast-changing technological environment: introducing new technologies in projects can be risky due to the uncertainty inherent to innovation; at the same time, failure to do so can lead project obsolescence.

Projects themselves have also changed. The overall trend has been an increasing complexity. As more powerful technology becomes available, our ambition and thus the scope of projects has been growing wider. Integration of different technologies dominates the scope of current projects. This has major technological and organisational impacts: the systems being developed comprise different types of components and services, and projects are often implemented by consortiums, with intensive outsourcing and sub-contracting. All this has brought new problems and challenges to the management of projects. It appears organisations have not learned yet to cope with them successfully. Several studies, empirical evidence, and even common-sense, all suggest that too many projects fail to achieve their targets, while others are cancelled prior to completion. This is a scenario that needs to be changed.

It is fair to wonder whether most failures are due to optimism, as an intrinsic human need to face complex challenges. It follows the question of whether without this optimism, better results would have been achieved. Or are current overruns, after all, good results? In first place, this argument does not apply to projects which are cancelled without providing benefits. Secondly, even if we attribute overruns and delays to optimistic targets, it still remains the question of whether better results could have been achieved – i.e. the same developments in less time, with higher quality, and requiring less effort. Of course, improvements are almost always possible. Rarely do organisations perform at their "optimum" level. However, the current scenario in project management is not just about seeking "unattainable perfection". There is a general feel that unnecessary expenditures and other dis-benefits tend to occur too often. At the same time, organisations appear to be failing on what is a basic instinct of survival: learning with mistakes. It has become common-sense that problems in projects are repeated over time, as if they were passed onto the next project as a "disease". This is certainly not a phenomenon original and unique to the project management field. As Peter Senge points out, only those organisations which become "learning organisations", will prosper and survive in the future.

The project management discipline has been developing much knowledge in those areas dominated by "physical laws". This is the operational arena, which includes issues like work specification, work and resources scheduling, cost budgeting, accounting and control. It is certainly important to develop scientific knowledge in this arena. Exerting control over the operational issues of projects is essential, and overall one can say that this has been done with success. The usefulness and merits of techniques like the WBS, PERT/CPM, earned value, among others, are unquestionable. However, problems of social, human and strategic nature in projects are not addressed by this type of knowledge and techniques. As discussed in the initial stage of this research, the main causes of project failure can be found in this area. These problems stem from the new environmental conditions and characteristics of projects mentioned above. They are problems of complex strategic nature, and thus dominated by social and human factors.

The project management discipline therefore needs to develop effective managerial knowledge in this area, capable of solving the practical problems. This is a difficult challenge because projects are complex social systems, where individual and organisational behaviour are crucial to the outcome. Those who need to exercise control over these systems are part of their working laws. Their observations are biased by their own emotions and desires – the Heinseberg's principle of uncertainty applies at the social level. It is under these conditions that project managers are faced with problems everyday, and struggle to solve them. Addressing social issues requires a different perspective than the analytical thinking of the operational arena. Problems cannot be split into elementary sub-problems, and solved separately. Problems in social systems are primarily generated by the reciprocal interactions of its constituent elements. Decomposing a problem thus eliminates these interactions, which therefore can no longer be observed, analysed and handled properly. A systemic view is required.

System Dynamics is a modelling approach aimed at studying the behaviour of social systems. It has four important characteristics: (i) it adopts a systemic and holistic perspective of social systems and of their problems, (ii) it focuses on modelling the dynamic interactions that take place within these systems, (iii) it captures explicitly subjective factors of human nature, and (iv) it leads to the development of structured knowledge, in the form of qualitative and quantitative simulation models. System Dynamics can therefore be the basis to developing scientific knowledge (and thus solving problems), in the strategic social arena of project management. There has been some successful applications in the past, which are reviewed in this research. These were valuable initial efforts. However, much work needs to be developed so that System Dynamics can become an effective standard project management tool, accessible to practising managers. Most organisations do not know about this technique, and even less about how to use it in practice.

It is concluded in the early stages of this research that System Dynamics can be used to support the project manager at the strategic level, where tools and techniques are scarce. A premise of this research is that in order to effectively apply System Dynamics in this way, its use needs to be integrated within the established project management process. Within this process, it must then be articulated effectively with the traditional techniques and tools. The development of a formal process methodology, capable of providing performance improvements, was the logical answer to this need. It therefore became the main aim of this research.

As it will be shown throughout this work, the methodology developed is based on two main findings: first, it is technically feasible to formally integrate a SD project model with the PERT/CPM based models, through the establishment of analytical links. This type of formal, quantitative integration had never been done before. This could be expected to be a difficult task because SD and PERT/CPM project models assume rather different perspectives of a project: they incorporate and quantify different types of factors, they capture different types of effects that take place within a project, and they consider different levels of detail. The second major finding is that the combined use of SD and the PERT/CPM based models holds the potential for valuable synergies, which cannot be achieved if the two types of models are employed separately. The methodology developed in this work establishes a well defined process to articulate the use of the two models, so that these synergies are effectively achieved. The resultant interleaving of the two models leads to a continuous dialect, from which novel questions and issues about the management of a project are raised. Based on fieldwork experimentation, this process was designed to maximise the synergistic effects.

1.2 Overview of the research conducted

The research here presented was conducted in four main stages: (1) initial literature review and theoretical analyses, (2) development of a conceptual framework, (3) implementation of extensive fieldwork, and (4) compilation of results achieved, and specification of a formal methodology.

The initial literature review focused on two main issues: (i) analysing in more detail the need to improve project performance and the causes of failures, and (ii) investigating and studying the past applications of System Dynamics to project management. In the first issue, the research focused on identifying reasonable evidence that project failure is a current problem, and that this is mainly due to factors of social and human nature, which need to be addresses at the strategic level. This analysis concluded that the traditional approach to project management (including current and on-going developments), does not provide processes, tools and techniques to address problems at this level. The traditional approach is focused on the operational issues. Effective tools and techniques to be used at the strategic level require a more systemic view.

During the analysis of the current developments in project management, the application of System Dynamics modelling was identified as delivering such a systemic view. The initial literature review therefore shifted its focus to studying the past applications of System Dynamics to project management – interestingly, these applications were undertaken within the SD community, while there was little awareness of them within the project management community. System Dynamics modelling was identified as a technique grounded on a holistic and strategic perspective of social systems, and aimed at addressing human subjective factors. It therefore appeared most appropriate to address the current causes of project failure. The review and analysis of its past applications to project management confirmed this: SD had been used with notorious success in addressing effectively strategic issues, where the traditional approach had failed to do so.

Overall, there was strong evidence that System Dynamics could provide distinctive benefits to the project management process. However, in these past applications

System Dynamics had been used in total isolation from the traditional project management process. There was no reference regarding how and when it was used within the logical flow of the project management process, nor how it had been articulated with the traditional tools and techniques (in fact, it just had not). It was therefore concluded and suggested that for practising project managers to have access to the distinctive benefits of System Dynamics, this modelling technique needed to be formally integrated within the existing project management process. The benefits of a formal integration were grounded on the general principle that it brings discipline and synergies.

The second stage of this research was therefore to develop a conceptual framework, formally integrating the use of System Dynamics within the project management process. The initial steps immediately revealed a lack of understanding about this issue. In the first place, it had never been done before. Secondly, integrating the use of SD implies changes in the way management actions and procedures are carried out. It may even have impacts at the technical product development level, within a project. An appropriate integrated process also depends on several aspects regarding the practice of the traditional project management process within an organisation (e.g. data collected, availability of updated project plans). As a consequence, it was not appropriate to develop comprehensive methodology before clarifying these issues through a more in-depth understanding. The most effective way of achieving this would be to carry out "action research", through extensive fieldwork in a real project. At the same time, it was also perceived that some structuring of the existing initial understanding would be useful. It would help to clarify ideas, identify critical issues, advance some initial solutions, and it would also provide a valuable starting-point for the fieldwork. This could take the form of a high-level conceptual framework, which would be developed based on the conclusions drawn from the literature review and on logical reasoning. This conceptual framework was developed, consisting of an overall rationale. It proposed high-level solutions and ideas about the requirements of an SD project model, its links to the traditional PERT/CPM based models, its potential roles within the project management process, and its appropriate level of detail.

The third stage of this research was to conduct the fieldwork. This took place within a large-scale software intensive project, at a major world-wide defence company (British Aerospace Systems, UK). This lasted for about 18 months. The main aim was to seek a deep understanding about the practical integration of SD, through a direct involvement in the problem, and so that a formal methodology could be developed. In this project, the researcher became part of the planning team. The research approach being adopted became a clear case of "action research". The conceptual framework previously developed was used as the starting point to integrated the use of SD within the project management process. Throughout the fieldwork, this framework was readjusted and refined into more detail. This involved a series of iterations, which required that a number of application scenarios were available. The large-scale and long time-frame of the project provided this.

The fieldwork was carried out based on the expectation that a beneficial application of SD in a formally integrated manner would be identified. It was further expected that some form of synergistic effects would emerge, which would not be achieved were the SD and the traditional models used separately. This would not imply that any form of integration would be useful. On the other hand, the new resultant project management process needed to be practical, flexible and capable of responding to the many constraints of reality. This was precisely what motivated the choice for an action research fieldwork, against an up-front comprehensive theoretical development. The expectation that there would be at least one beneficial way of integrating SD, was motivated by the evidence of the SD benefits in the past applications reviewed. However, the possibility that there could be no such way, was not excluded. Therefore, another important aim of the fieldwork was to test alternative routes, select the appropriate ones, and identify and gather evidence of the benefits. The expectation of synergies was based on two main factors: first, formal integration would ensure that the SD and the traditional PERT/CPM based models were formally consistent one another, thereby preventing a same project to be inconsistently represented and analysed in the two different types of models. Secondly, as it will be discussed, the SD and the traditional models assume a rather different perspective of a project, focusing on different elements and phenomenon of reality. Formal integration would therefore establish "channels of communication" for these models to exchange findings, allowing for a valuable dialect of project analysis to take place. For example, the feasibility of a project plan developed in one model would be assessed in the other model. This dialect would probably raise questions and issues, which most likely would be ignored were the models used separately.

The fieldwork was completed successfully. In-depth understanding was gained and the conceptual framework was refined into the necessary detail, so that a formal methodology could be specified. The application of SD in this way was beneficial to the project, and evidence was gathered accordingly. The last stage of this research therefore consisted in compiling the results and conclusions, and develop a formal specification of the proposed *generic* methodology. This methodology is described in great detail in this work. It is proposed as generic, in the sense that any organisation implementing a project can use it to apply SD in an integrated manner. It is not specific to any particular industry or organisation. However, it does establish some requirements regarding the project management process adopted by the organisation. For example, it is not appropriate for organisations that follow an *ad hoc* process. Nevertheless, it is also a flexible approach in this respect: its logical process flow considers alternative paths, to accommodate different scenarios of project management practise. The methodology is formal in the sense that it specifies objectively all the steps of the process, as well as the links with the PERT/CPM model. It is important to note that this is not that type of formality which tends to impose unsustainable conditions and rigidity. On the other hand, a generic and simple specification language is used, so that the methodology is accessible to the practising manager. In addition, the proposed process and links can be implemented at various levels of formality, depending on the specific circumstances. It is not necessary to implement all the process steps and analytical links. A partial and thus less formal implementation is viable. Overall, the methodology delivers a potential level formality which can be relaxed to accommodate practical constraints. Finally, it is also important to note that while the methodology focuses on "how to use" the SD model within the project management process, it also includes a method to support the development of an appropriate project model for the specific project. This was not initially considered within the scope of the fieldwork. However, it emerged as an important requirement for the practicality of the methodology: any organisation interested in using SD in an integrate manner needs a project model. Most organisations do not have the expertise and relevant experience to develop such model. The method proposed as part of the methodology is aimed at assisting organisations in the task of model development.

1.3 Structure of the thesis

This is a long thesis (although a large number of illustrative tables and figures account for a good portion of it). There are some good reasons for this. Overall, it results mainly from two factors: (i) the wide scope of the research, and (ii) its inherent complexity.

This was a first novel attempt at developing a formal process to integrate the use of System Dynamics within the traditional project management process. As a consequence, in the author's view, it was crucial to address as much as possible of the whole project management process. A wide range of issues and problems needed to be covered. This is opposed, for example, to identify a particular problem (e.g. "automating the data-transfer in initial planning", or "update of SD model in monitoring"), and then focus on the development of a detailed isolated solution. But why was it so crucial to cover a wide range of issues? Project management is by nature an *integrated* process: decisions and policies in any one area must account for the decisions and policies in the other areas. It is common sense in the engineering field that the first design of a system must be global, and cover most of its functionality – the same happens with decisions in management. Only then, is it appropriate to refine individually its sub-components. Therefore, it was the author's view that starting with a local isolated solution, focused on a particular problem, would not be the right approach. This was the main conceptual reason. There was also another reason of practical nature. Action research is known to have the difficulty in providing the researcher with a clear end-point to the fieldwork process of gaining understanding. Obviously, a full complete understanding can never be achieved. But the more the researcher learns, the more he/she feels it is important to proceed. And the deeper the understanding gained, the more there is to incorporate in the theory being developed. This phenomenon took place in this research. In addition, it reflects not only the learning from an 18 months fieldwork experience carried out within a major project, but also a further 12 months of professional consulting experience, in the application of System Dynamics to the management of large projects and programs. It was hard to exclude much of the knowledge, directly or indirectly, developed.

Although the thesis is long, the large parts of it can be found only where the detail is essential. In particular, this applies to the description of the formal methodology, where additional details can also be found in appendix. The overall logical flow of the thesis is simple. An exhaustive reading of the detailed parts is not mandatory in order to develop a good view of the work developed. A road-map is here provided to guide those who may not wish to read the whole detail in a first pass. Depending on the objectives, level of interest and time available, the reader may follow different paths with different levels of detail. First, the thesis' logical flow is briefly summarised and the road-map is then presented.

The next chapter 2 provides a literature review of project management, an overview of the System Dynamics methodology, and a review of its past applications to project management. This is a chapter of review, analysis and conclusions. In the first part, the traditional approach to project management is briefly revised, along with the definition of some key underlying concepts. The importance of project management and the nature of project failure are discussed. Some of the relevant current developments are reviewed and analysed. It is concluded that these developments keep a focus on the operational issues and that a more systemic view is missing. The second part provides a brief methodological overview of System Dynamics modelling. This is followed by an exhaustive review of past applications to project management. These applications are first analysed and a comparative analysis with the traditional models is further developed. The chapter concludes that since SD addresses the systemic causes of project failure, a beneficial way ahead could be an integrated approach.

The brief chapter 3 presents an overview of the methodological research approach adopted in this work, and discusses some important related issues. The design of an appropriate research methodology was perceived essential for the validity of this work. The chapter starts by reviewing the background of the research, and from there it poses a summary research question and an hypothesis, and further outlines the aim and objectives of the research. It then provides a concise discussion of the methodological approaches to management research, focused on the dichotomy "positivism versus phenomenology". Action research, as a form of phenomenology, is also briefly discussed. The actual research approach adopted in this work is then described. This was predominantly of phenomenological nature, although it also included some elements of positivism. The appropriateness of this approach is discussed, and the research methods employed are identified. An initial high-level work plan is outlined.

Chapter 4 presents the conceptual integrated framework, which was developed prior to the implementation of the fieldwork. This is another brief chapter, which discusses important issues but without detailing them. The conceptual framework presented was developed on the basis of the review and analysis of the past applications of SD to project management (presented in chapter 2), and on logical reasoning. First, the general objectives and requirements of this framework are briefly discussed. A rationale is then developed around the requirements of a SD project model, the links with PERT/CPM models, the potential roles within the project management process, and the appropriate level of detail. Based on this rationale, the conceptual framework is proposed and the future research steps are identified.

According to the initial work plan outlined in chapter 3, the next step in this research was to conduct a fieldwork experience within a real project. Chapter 5 provides an overview of this fieldwork, which is herein referred to as "KDCOM case-study". This is another concise chapter. It starts with an overview description of the KDCOM project. While this is intended to be an informative description about the project, some elements were not presented by reasons of confidentiality. The second part of this chapter provides a concise description of the implementation of the fieldwork. This includes the initial objectives, the critical issues for success, and the results achieved. The main conclusions are presented and briefly discussed.

The following two chapters 6 and 7 provide a detailed description of the final integrated methodology developed (herein referred to as "SYDPIM methodology"). This constitutes the main and major output from this research. These are of course long and extensive chapters. Along with the formal specification of the methodology, they also discuss in detail various relevant issues of both conceptual and practical nature. As already noted, being a first novel attempt, all relevant issues were considered as much as possible. Even the ones for which detailed solutions were not devised, a rationale discussion is provided. The author perceived it was important to raise these issues within the logical flow of the specification of the methodology. Chapter 6 describes the model development method, which is proposed as part of the methodology. The first part provides a review of the more relevant past developments of SD project models, and the second part specifies the development method proposed. It comprises a life-cycle of two main phases of design and implementation, each with some sub-stages. It also includes a model validation framework, which is considered essential to develop useful SD project models. The chapter finishes with an overview discussion of critical implementation issues. Chapter 7 describes the method to use the SD project model in an integrated manner within the project management process (herein referred to as "SYDPIM Project Management Method"). This is the core part of the SYDIPM methodology. It is specified as a formal model comprising four main integrated elements: the process logic, objects, analytical links, and activities. These elements are formally specified using a proposed simple "specification language". This chapter finishes with an important discussion about the practical implementation of the SYDPIM Project Management Method, which includes issues like appropriate software platforms.

Chapter 8 describes some practical applications of the SYDPIM methodology. This includes both the model development method and the project management method. This is another somewhat long chapter, due to the number of real world examples considered for illustrative purposes. These examples were taken from the application of SYDPIM to the management of the KDCOM project. As already noted, the "KDOCM case-study" fieldwork was carried out under the perspective that there would be ways to integrate SD in a beneficial manner, thereby leading to improved project management performance. However, it was important to gather evidence of the actual benefits achieved when implementing SYDPIM in the way described in this research. The purpose of this chapter is therefore of dual nature: to illustrate the practical implementation of the methodology, and to provide evidence of its distinctive usefulness. It would not be feasible to include a detailed "step-by-step" description of these practical applications, but considerable detail is provided to the more relevant aspects. Regarding the model development method, the more important elements of model design are presented, along with an overview of the implementation phase. Overall, five examples of application of the project management method are presented, by chronological order of actual implementation at KDCOM. The first example refers to a diagnosing exercise, as part of monitoring. The two following examples refer to project planning. The fourth example refers to the calibration of the SD model to the initial work plan of the second system development increment of the project. As initially planned, this was the last exercise of the fieldwork. The last example refers to another application of SYDPIM to the development of a second batch of the "KDCOM system". This took place beyond the original scope of the fieldwork. This chapter finishes with a discussion about some important issues regarding the practical implementation of SYDPIM within a real project environment. This includes potential limitations and some useful lessons learned.

The final chapter 9 provides an overview discussion of the main conclusions drawn from the research work here presented. This includes issues about the SYDPIM methodology, the research approach adopted, the difficulties encountered, and some proposed future developments.

As already mentioned, although this is a long thesis not all of its parts need to be read in detail, in order to develop a basic understanding of the research findings and proposed methodology. Depending on interest and purpose, different readers may consider reading different parts. Figure 1.1 presents a proposed road-map for alternative readings of the thesis. The left column in green identifies the essential reading, which is mandatory to understand the whole concept underlying this research. The central column in red identifies additional materials for an extended reading, covering the whole thesis. The materials identified in this column can also be seen as the ones which the reader may skip in order to carry out the essential reading. In chapter 5, the reader may skip section 5.2 which provides a brief description of the KDCOM project, wherein the fieldwork case-study was carried out. These materials are specific to the KDCOM project. In chapter 6, the reader may skip the individual detailed analysis of the more relevant past SD project models reviewed in this research. The reader may also skip the detailed description of the two phases of the SYDPIM Model Development Method. However, if the reader wishes to implement the method, then these sections must be read. In chapter 7, the reader may skip the formal specification of the objects and analytical links, as well as the detailed description of the activities of the SYDPIM Project Management Method. Again, for an implementation of the method these materials must be read. Chapter 8 presents five practical examples of implementation of the SYDPIM Project Management Method. For an essential reading, the reader may consider only one example (the one in section 8.3.4 is proposed). The right column in blue presents the appendices. Appendixes E, F, G and H are extended materials that refer to the literature review of chapter 2. Appendices G and H are literally extended and alternative versions of sections 2.2 and 2.3.1, which the reader may consider for a more detailed and exhaustive literature review. Appendix E presents the author's personal view of the project management process, which is appropriate for the context of SYDPIM. Appendix F provides a more detailed description of the basic project management tools and techniques, which were briefly presented in chapter 2. Appendices C and D present the formal specification of all the objects and analytical links of the SYDPIM Project Management Method. Throughout chapter 7, only some of these elements are formally specified for the sake of example and basic understanding. These appendices should be consulted in the extended reading of chapter 7. Appendices A and B are not presented in this road-map, as they are supposed to be consulted at any stage throughout the thesis. Appendix A provides a definition of key terms and expressions used in SYDPIM, which are particularly relevant to understand the materials of chapters 6 and 7. Appendix B presents various tables which also refer to these two main chapters.

According to this proposed road-map, the reader may choose to follow different paths, depending on interest and purpose. The materials in the essential reading are mandatory. From here, the reader may wish to read some or all the materials proposed in the extended reading. Appendices E, F, G and H are recommended if the reader is interested on a detailed analysis of the literature review carried out in this research. Appendices C and D are essential in case the purpose is the full analysis and/or practical implementation of SYDPIM.

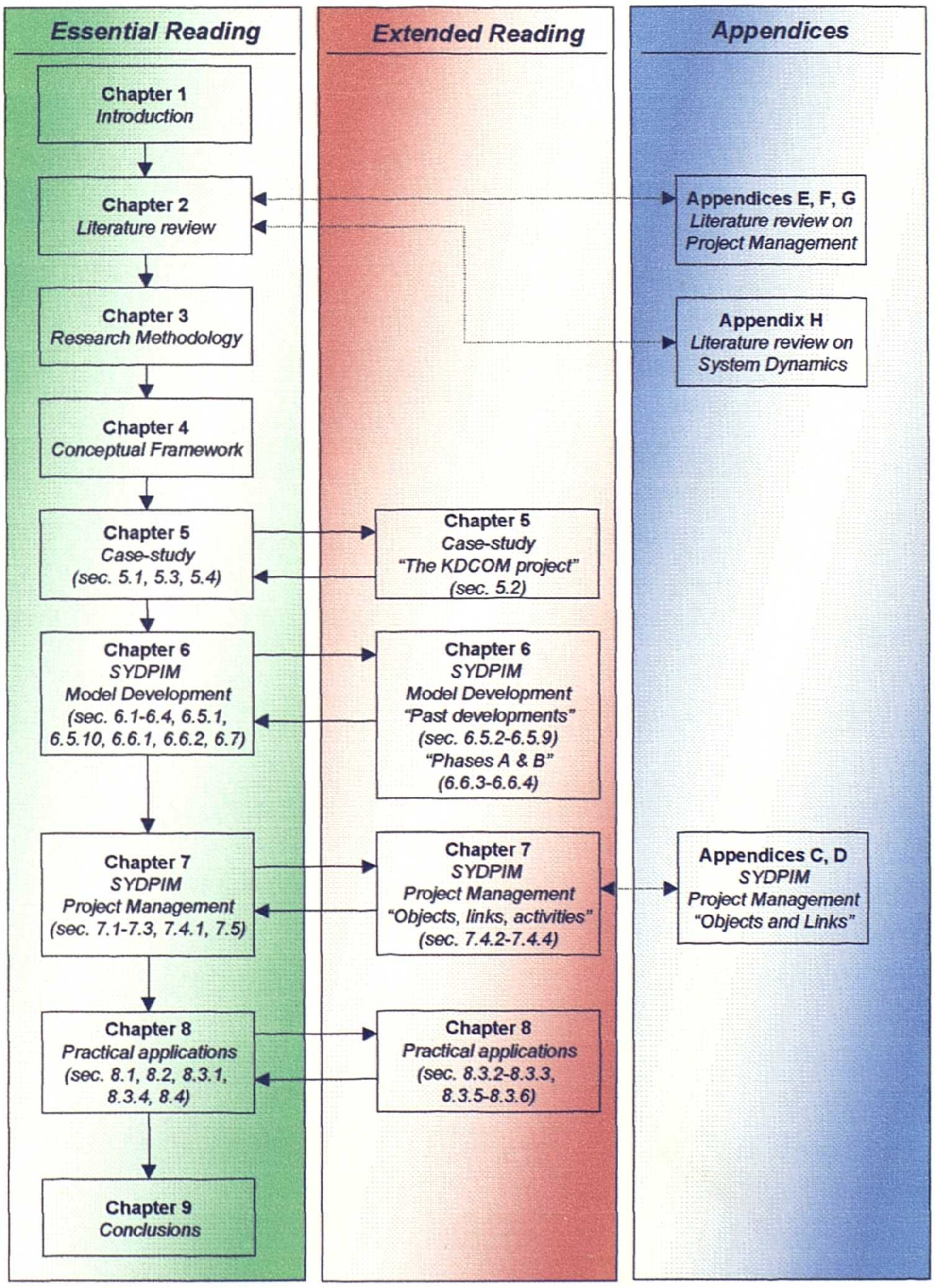

Essential Reading
Extended Reading
Appendices
Chapter 1 Introduction
Chapter 2 Literature review
Chapter 3 Research Methodology
Chapter 4 Conceptual Framework
Chapter 5 Case-study (sec. 5.1, 5.3, 5.4)
Chapter 6 SYDPIM Model Development (sec. 6.1-6.4, 6.5.1, 6.5.10, 6.6.1, 6.6.2, 6.7)
Chapter 7 SYDPIM Project Management (sec. 7.1-7.3, 7.4.1, 7.5)
Chapter 8 Practical applications (sec. 8.1, 8.2, 8.3.1, 8.3.4, 8.4)
Chapter 9 Conclusions
Chapter 5 Case-study "The KDCOM project" (sec. 5.2)
Chapter 6 SYDPIM Model Development "Past developments" (sec. 6.5.2-6.5.9) "Phases A & B" (6.6.3-6.6.4)
Chapter 7 SYDPIM Project Management "Objects, links, activities" (sec. 7.4.2-7.4.4)
Chapter 8 Practical applications (sec. 8.3.2-8.3.3, 8.3.5-8.3.6)
Appendices E, F, G Literature review on Project Management
Appendix H Literature review on System Dynamics
Appendices C, D SYDPIM Project Management "Objects and Links"

Figure 1.1 – Proposed road-map for alternative readings of the thesis

2. Literature review and preliminary conclusions

2.1 Overview and structure of this chapter

In this chapter, a brief literature review of Project Management and System Dynamics is carried out, as the basis to discuss the state of the art in these two areas. An exhaustive literature review is further carried out on the application of System Dynamics models to project management issues, as the basis to explore the potential roles and distinctive benefits of the System Dynamics approach. Some key preliminary conclusions are presented, suggesting the integration of SD within the traditional approach to project management.

Project Management is currently a well established discipline. There are numerous pieces of work in the literature where this is described in great detail. Therefore, only a very brief review of the key concepts is presented in section 2.2 of this chapter. Regarding System Dynamics as a generic modelling methodology, a brief overview is also presented in section 2.3.1. Although this is a much less mature and established discipline, there are also some good pieces of work available in the literature describing the methodology. The remainder of his chapter is dedicated to an exhaustive review and analysis of the past applications of System Dynamics to Project Management, which starts in section 2.3.2. This includes a brief discussion of why projects are appropriate candidates for the SD analysis, followed by a review of the major past applications based on an exhaustive literature survey. This shows the characteristics of the different studies and the several issues they consider. A final discussion about the potential range of applications is provided. Section 2.3.3 presents a comparative analysis between the traditional network based models and the System Dynamics project models, contrasting the different perspectives of the two approaches. This comparative analysis considers various aspects, like the modelling of the project work and the managerial needs addressed. The strengths, weaknesses, and the distinctive benefits of the System Dynamics approach are then identified and discussed. The chapter concludes that the distinctive contribution of System Dynamics is complementary rather than a complete alternative to the traditional approach, and therefore an integrated approach merging the best of both worlds could be the best way ahead.

Because the reader may not be familiarised with the Project Management and System Dynamics disciplines, extended and detailed versions of sections 2.2 and 2.3.1 are presented in appendices G and H respectively.

2.2 Project Management

2.2.1 What are projects and project management?

There is some excellent updated literature available where more or less detailed discussions about the concept of project and project management are presented (e.g. Nicholas 1990, Turner 1993, Kerzner 1998). Appendix G provides a discussion, addressing the following topics: a basic definition of the concepts "project", "program" and "project management", the project main characteristics, the project life-cycle, the parties involved or stakeholders of a project, and the organisational project issues. A summary of the key definitions is here presented.

Regarding the definition of a project, Turner (1993) proposes: "*An endeavour in which human, material and financial resources are organised in a novel way, to undertake an unique scope of work, of given specification, within constrains of cost and time, so as to achieve a beneficial change defined by quantitative and qualitative objectives.*" This definition will be used as the basic concept for the purpose of this research. A sub-set of Turner's projects will be considered, narrowing the scope of the definition down to tangible projects – however, the work here proposed may well be validly extended to other Turner's projects. The proposed definition is as follows:

Project:
a complex and unique undertaking aimed at the design, realization and delivery of a tangible product.

This definition implies that the projects herein considered deliver a tangible product. The development of this product comprises the two main phases of design followed by realization. Design comprises the work required to conceive the product and specify what it is. The latter phase of "realization" comprises the work required to actually "physically" build the product.

In this research, the object of analysis is the individual management of a single project. However, just like most elements of the project management framework are applicable to program management (which implies managing various inter-related projects; Nicholas 1990), most of the output and conclusions of the research here presented are also transferable to the management of a program.

Regarding the definition of project management, the Project Management Institute (PMI), USA, proposes: "... the application of knowledge, skills, tools and techniques to project activities in order to meet or exceed stakeholders' needs and expectations of a project." (PMI 1996). In simpler terms, project management is a continuous undertaking primarily aimed at ensuring that the project is completed within the objectives, and which takes place within the context of the project itself. This definition has some important implications: (i) it requires effort, (ii) it is focused on the project objectives, (iii) it takes place continuously throughout the project life-cycle and (iv) it is part of the project implementation process. Turner (1993) argues that the management process can be considered at three different levels: integrative, strategic and operational. The present research will focus on the last two levels of project management.

Regarding the project objectives, it will be considered in this research that they can be mapped into four main dimensions: cost, time, requirements (what was achieved) and quality (how well was it achieved).

Just as the PMI assumes the pre-existence of some knowledge about how to manage a project, Turner (1993) also argues in favour of a process. For the purpose of this research, the traditional approach to the project management process will be assumed, as briefly described in the following section 2.2.2.

2.2.2 A review of the traditional project management framework

Overview

Since project management first emerged as a management discipline in the late 1920s, with the first uses of Gantt charts (Nicholas 1990), the ever growing community of practitioners and researchers has been developing a well established project management framework. This framework comprises various elements, some unique and specialised in project management, while others have been borrowed from other management and engineering disciplines. These elements include various generic processes, practices, procedures, tools and techniques.

Various national and international project management associations have been attempting to develop their own standard project management body of knowledge (PMBOK) (Wirth and Tryloff 1995). For the sake of a brief description, the PMI perspective will be herein assumed (PMI 1996). The author's own view of the project management process is presented in appendix E – the main purpose of this description is to provide a dynamic framework of the project management process, wherein continuous iteration, refinements, rework and interactions play a major role (as they do in the real world); later in this research, this dynamic framework will allow for an easier understanding of the use and integration of System Dynamics models.

The project management process

The PMI (1996) proposes a view of the project management process in which various elementary sub-processes take place within the two main processes of management and engineering. These elementary sub-processes are grouped according to five main types of activities: initiating, planning, executing, controlling and closing. These are inter-related as show in figure 2.1.

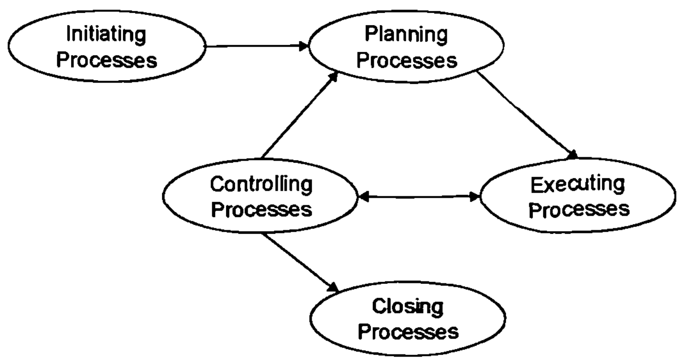


Figure 2.1 – Main project activities (process groups) and their interactions (PMI 1996)

This set of activities (or "process groups", as referred to by the PMI), takes place throughout the whole project life-cycle. Therefore, they take place within each project phase and also interact across these phases. The PMI considers a total of 37 elementary processes. Each takes place within a specific project activity where some are considered as "core" processes, while others are considered as "facilitating" processes. They are linked through inputs and outputs. The PMI therefore describes them in terms of inputs required, tools and techniques used, and outputs produced. They are also grouped into the nine main knowledge areas of: integration management, cost management, communications management, scope management, quality management, risk management, time management, human resource management and procurement management.

Procedures, tools and techniques

The project management process constitutes a framework wherein various tools and techniques are employed. The PMBOK (PMI 1996) identifies a specific set to be employed within each elementary process. The basic ones are as follows (a more detailed description can be found in appendix F):

(1) *product breakdown structure (PBS)* – aimed at decomposing the project product into elementary sub-components;

(2) *milestones and deliverables chart* – specifies the planned dates for these two elements throughout the whole project;

(3) *risk register* – a table listing the currently identified risks, ranked by "seriousness". It is aimed at identifying, analysing, monitoring, controlling and mitigating the relevant project risks;

(4) *front-end estimating techniques* – provide estimates for the project cost, schedules and resources required, prior to the development of a detailed work plan;

(5) *work breakdown structure (WBS)* – it is central to the whole process of project planning and control. It specifies all the work that needs to be performed, so that the project objectives are achieved;

(6) *organisation breakdown structure (OBS)* – specifies the hierarchical structure of the organisation which will accomplish the project work.

(7) *responsibility matrix (WBS x OBS)* – a matrix crossing the WBS against the OBS. In each cell, various types of responsibilities can be specified;

(8) *Gantt charts and PERT/CPM networks* – used to schedule the project tasks and allocate resources. They are the basis of the whole project planning function (see appendix E);

(9) *"earned value" and other control metrics / indices* – support the monitoring of the project status and identification of deviations. These techniques are based on specialised metrics and measurements.

The techniques described above are the more commonly used in project management systems, and most of them have been developed within the project management discipline. There are many other techniques that can assist the project manager. The PMBOK (PMI 1996) provides an extensive list. In appendix G a brief description of some of these other techniques and tools is presented.

2.2.3 Current scenario: "management by projects" versus "project failure"

The increasing rate of change and the complexity of the new technologies and markets impose the need for quick and effective responses. As a consequence many organisations started adopting "management by projects" as a general management approach (Turner 1993). At the same time, projects have become increasingly more complex (Williams 1997).

Unfortunately, project failure has also been a major problem. Overall, over-runs of 40% to 200% are common, while other projects are cancelled before completion and after considerable expenditure (Morris and Hough 1987). Various studies and surveys appear to support this evidence (Cooper 1999, Cooper and Mullen 1993). The tendency for project failure appears to be a fact (Davidson and Huot 1991, Turner 1993).

Major technical breakthroughs are being achieved in various industries to increase product development and support increasing productivity levels. Will this solve the problem? Probably not. The increasing complexity of projects tends to outweigh the power of the technical breakthroughs. Most likely, the main causes for project failure are not being addressed properly or are being ignored by the traditional approach. Williams (1997) argues that as modern projects became complex, strategic human issues became a crucial factor of success and the traditional approach does not address them properly. It is therefore fundamental to identify these causes, review and improve the traditional project management approach, so that they are addressed. This may imply changes to the general process logic of the approach as well as the introduction of new techniques and tools.

2.2.4 The nature of project failure and the traditional approach

Project failure can be blamed on many factors. Uncontrollable external forces are often cited but the real causes may well be internal: a defective project management system, with ineffective organisational practices and procedures (Nicholas 1990). Good project management should be able to cope with many of the adverse external influences and thereby ensure a successful completion, despite the environment.

However, even organisations that implement such project management process experience major failures. So, what is really causing projects to fail? Is the traditional project management approach missing something critical?

The main problem of the traditional approach appears to be the lack of a strategic perspective to address the importance of human "soft" factors" – in this research, the term "strategic project management" will be used as corresponding to the individual's project strategy (Rodrigues and Bowers 1996a) –, as well as an undue focus on the operational issues (Turner 1993). Some important studies suggest that the main causes of project failure are to be found in areas such as the political/social environment, legal agreements and human factors, with the majority of the factors relating to strategic issues of project management (Morris and Hough 1987). These factors are not addressed explicitly by the traditional project management techniques.

Human factors have a critical impact on the strategic issues of project management. At the strategic level, their influence is of greater magnitude and their subjectivity is more difficult to manage (see Rodrigues and Bowers 1996a for a more detailed discussion). There is a need to understand better the strategic issues of project management and to learn effectively from past failures; this can only be achieved through a more formal systemic analysis (Morris and Hough 1987, Davidson and Huot 1991, Cooper 1993, Williams 1997).

2.2.5 Further developments

Despite its usefulness, various limitations of the traditional approach have been recognised. Further developments have therefore been undertaken and others are underway. It is not the purpose of this research to investigate and present here all these developments exhaustively. They are numerous and can be found, for example, in the proceedings of the more important project management conferences, like the ones organised by the International Project Management Association (IPMA) and the PMI.

A good review of latest developments can be found in Williams (1997), where various emerging techniques aimed at coping with complex projects are described by their authors. Regarding modelling techniques, the interesting conclusions from this work were that PERT/CPM based models are inadequate to cope with complexity because they do not incorporate management actions, do not address uncertainty adequately, and do not capture human "soft" factors and "systemic" effects. These and other shortcomings of the traditional approach are also discussed in some detail in Rodrigues (1994).

For the purpose of this research, the more relevant developments are described in appendix G. This includes PERT/CPM based models, several other network based approaches, front-end estimating techniques, the Critical Chain Project Management (CCPM) approach, and risk management.

The main overall conclusions from the project management literature review carried out in this research are summarised in the following section.

2.2.6 Conclusions: what is missing?

The project management discipline has developed a well established and comprehensive body of knowledge. The overall approach is based on a top-down decomposition and analysis of the project, followed by the bottom-up aggregation of results. The WBS, OBS, responsibility matrix, PERT/CPM networks and earned value (EVM), are the main techniques employed to implement the project management process. This perspective has motivated a focus on the operational issues of projects. The tools and techniques based on the WBS and logical network cope effectively with problems at this level. The traditional approach has therefore some important merits. It provides a robust framework to implement control at the basic project level. While this is not sufficient to ensure control of the whole project, it is an essential requirement. The success and usefulness of the network based techniques at this level has motivated extensive research to develop more complete and flexible models. Even new approaches like CCPM, which take a different perspective of planning issues, are based on the project operational network.

Experience has shown that operational control is not sufficient to cope with emerging complexities in modern projects. The need for a complementary systemic analysis has been identified (Rodrigues 1994), in particular for risk management purposes (Williams 1998). Attempts to employ the traditional techniques in order to cope with these systemic problems prove counter-productive and are likely to fail. The required systemic view must focus on the various dynamic interactions among the project elements, where the whole becomes much more than just the linear sum of the parts.

As it will be seen in the following sections of this chapter, Systems Dynamics modelling appears to offer this strategic alternative, assuming a holistic view of the organisation with an emphasis on the behavioural aspects of projects and their relation with managerial strategies. There has been a number of academic and practical applications of System Dynamics to project management. The remainder of this chapter addresses the need for a better understanding of the nature, differences, similarities, and purposes of traditional and System Dynamics approaches. If System Dynamics models are to play a core role in the future developments of project management, it is important to understand their distinctive contribution to the current body of knowledge and their place in a future methodology.

2.3 System Dynamics

2.3.1 A methodological overview

Proceeding from previous work initiated at M.I.T, in the late 50s (see Forrester 1958, reprinted in Roberts 1978), Professor Jay Forrester published in 1961 a book entitled "Industrial Dynamics" (Forrester 1961). In this book, Forrester proposed a new computer-based modelling methodology and with it an underlying paradigm of thinking about managerial problems which, at that time, he summarised as follows: "... the investigation of the information-feedback character of industrial systems and the use of models for the design of improved organisational form and guiding policy." Initially, his work focused on analysing large industrial systems, and hence the methodology was termed as *Industrial Dynamics*. Further academic and practical developments would shift the focus to many other types of social systems, and this name soon gave way to the more general term *System Dynamics*.

As the motivation for Industrial Dynamics, Forrester identified the need for a solid scientific basis for the effective management of large industrial systems. At that time, this need was emphasised by the many observed failures in the design and management of this type of systems.

It is not the purpose of this research to present an exhaustive description of Forrester's modelling methodology, as this can be found in several books and other publications (e.g. Forrester 1961, Goodman 1974, Coyle 1977, Richardson and Pugh 1981, and more recently Wolstenholme 1990, and Coyle 1996). A brief outline of the underlying process of the methodology is presented.

The System Dynamics modelling process

As presented by Forrester in Industrial Dynamics, the SD modelling process should evolve towards the development and use of quantitative simulation models. Forrester's general method comprised several stages, starting with problem analysis, following onto formal model development, and finally to model application through repeated experimentation.

Like any other modelling methodology, the process comprises three main phases: (1) the problem is identified and described, (2) a model is developed with the purpose of analysing the problem, and (3) the model is used as a "tool" to help designing a satisfactory solution for the problem. The SD process is described by Forrester as being iterative, as opposed to a linear progression: at any stage it can cycle back to previous steps, thus feeding-back improved understanding and knowledge about the system and the problem. The emphasis is on interpreting the simulation results and revising the simulation model, in the search for better structures and policies. This emphasis on iteration and model revision highlights that the process is not intended to consist of a pure sequence of "model development" followed by "model use". A perfect model is never achieved first time, and changing the model is an essential requirement to analyse the problem and to identify solutions.

Over the years the need for methodological improvements has been recognised (Coyle 1973). Further advances have been made in an attempt to achieve a more formal and clear definition of the SD process (Forrester 1968, Goodman 1974, Coyle 1977, Roberts 1978, Richardson 1981, Wolstenholme 1982, Richmond 1990, Wolstenholme 1990, Coyle 1996). Much effort has also been directed towards formalising some particular aspects of this process (e.g. Burns 1977, 1979; Forrester and Senge 1980, Randers 1980, Wolstenholme and Coyle 1983, Wolstenholme 1994, Richardson 1995, Barlas 1996, Coyle 1996, Lane and Smart 1996).

As a result of these developments, several authors have proposed other descriptions for the SD process. Table 2.1, compares some of these descriptions. There is a main sequence of five steps which is recognised by all authors (left column), so is the overall iterative nature of the process. These five steps are as follows:

(1) problem definition and system conceptualisation;
(2) development of an ID;
(3) use of the ID;
(4) development of a quantitative simulation model;
(5) use of the simulation model.

In this research, two separate phases of qualitative and quantitative analysis will be considered explicitly in the SD process. It will be considered that IDs are the most appropriate precedent for simulation modelling and hence should be used in the qualitative phase (which may also find valuable support in other techniques like cognitive mapping; e.g. Eden 1994, Ackerman et al 1997). It will also be considered that the complete SD process should include the quantitative phase, through the development and use of simulation models. Figure 2.2 depicts the overall structure of this iterative process, highlighting the two phases of qualitative and quantitative System Dynamics, as well as the two main outcomes. Each of the phases is divided into the two sub-phases of "model development" and "model use". These two main phases can be seen as continuous activities which are carried out throughout several iterations, with the ID and the simulation model being continuously revised. In an ideal scenario, the two types of analysis will interact in a complementary manner, contributing to the continuous improvement of the modeller's understanding of the problem and of the models themselves.

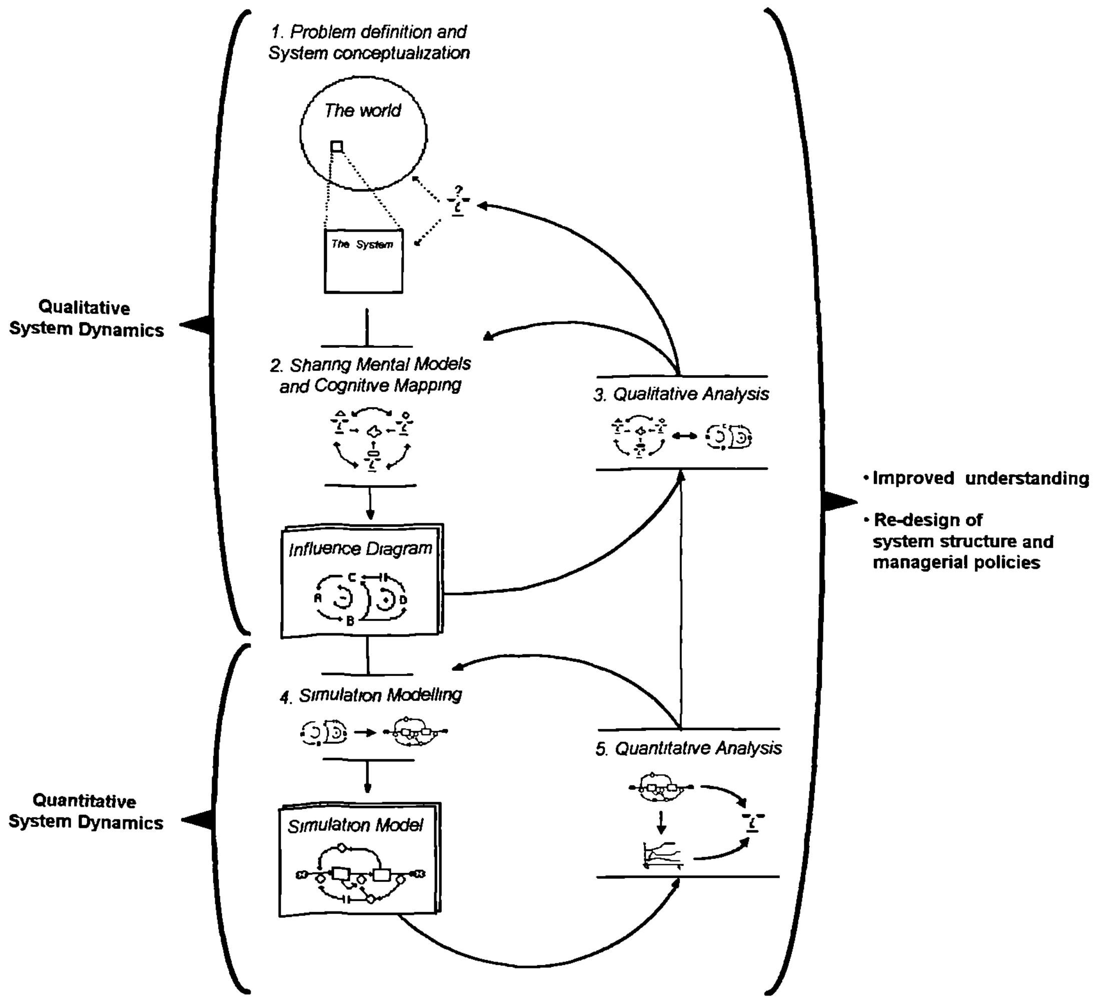

1. Problem definition and System conceptualization
The world
The System
Qualitative System Dynamics
2. Sharing Mental Models and Cognitive Mapping
3. Qualitative Analysis
Influence Diagram
• Improved understanding
• Re-design of system structure and managerial policies
4. Simulation Modelling
5. Quantitative Analysis
Quantitative System Dynamics
Simulation Model

Figure 2.2 – A generic view of the System Dynamics process

In appendix H, a description of each of the five main individual steps is presented, as well as the basic SD notation to be used throughout this research – the reader unfamiliar with SD should refer to this appendix. A brief discussion about the use of different modelling elements throughout the SD process is also presented; the different perspectives about the SD process advocated by different authors within the SD community are further summarised.

Model validation

Validation is a crucial issue in System Dynamics, just as in any formal modelling approach to social systems. Criticisms regarding both conceptual and practical aspects of SD have been following the emergence and evolution of the methodology. For a discussion of this issue the reader should refer again to appendix H. The key principles adopted in this research regarding model validation are as follows:

- what is "to predict"? In a model of a social system, to predict is to anticipate a future scenario implied in the human knowledge and "beliefs" incorporated in that particular model.
- what does it mean to say that a prediction is "accurate"? A prediction produced by the model is accurate if this anticipated future scenario has a good chance to be achieved;
- when is a prediction "correct"? A prediction will prove correct if the expectations are fulfilled;
- what is a "valid model"? A valid model is *not* the one that represents reality as it is, delivering a true image of the inevitable future. Instead, a valid model is the one that describes the system in accordance with the modellers' mental models, while delivering consistent and achievable images of the future;
- what is a "useful model"? In real situations, there is usually a range of possible predictions which are both consistent and achievable. The useful model is the one that helps the analyst to identify, select and plan better achievable futures.

The usefulness of the research here proposed rests on these principles. In appendix H a more detailed rationale is provided. The main conclusions from this rationale are: (i) the concept of validation in social systems requires a shift in perspective from the path of "realism" to the path of "constructivism" (Roy 1991), (ii) in System Dynamics there is no single test that can ensure the validity of a model. Instead, validation is a continuous process of building confidence as validity tests are progressively passed. Specific tests have been proposed (Forrester and Senge 1980, Homer 1893, Barlas 1996); and finally (iii) model legitimisation (Landry et al 1996) is crucial for the success o System Dynamics as a "valid" modelling approach.

Further critical issues

Validation is not the only critical issue in System Dynamics. An interesting discussion about the future of System Dynamics is presented by Richardson (1996). A more recent review of the current trends in System Dynamics undertaken by the same author (Richardson 1999), reveals that while some progress has been made, the need for further deeper developments still remains. As a modelling technique, the methodology itself has important characteristics which affect its scope of application. Some of the critical issues identified by the author in this research are discussed in appendix H, which includes: aggregation, continuity, endogenous perspective, quantitative simulation, incorporation of human factors, and incorporation of decision-making processes, among others. These are some of the SD features that must be handled with special attention when applying the methodology. However, it should be noted that these potential difficulties steam from the ambitious aim of the methodology. Any other modelling approach attempting to address the same type of problems within complex social systems, is just as likely to face these same difficulties. In other words, these limitations are not a characteristic of the SD methodology but rather a result of its ambitious aim.

2.3.2 Review of applications to project management

Introduction

System Dynamics is aimed at analysing complex problems which occur within social systems. The problems are of dynamic nature and hence can be related to an undesired behaviour. This behaviour can be described as patterns over-time and is assumed to be of endogenous nature, resulting primarily from the system's internal feedback structure. A SD model helps to identify the counter-intuitive aspects of a system's behaviour, explaining the problem through the identification of important feedback loops and how these alternate their influential dominance over the system. Most of the feedback effects considered in SD models are non-linear and many are affected by delays.

The appropriateness of System Dynamics and analyse problems in projects follows from the fact that projects are also complex social systems. Their behaviour is difficult to understand and manage. Many problems in projects can easily be related to dynamic patterns of behaviour. For example, the persistent occurrence of schedule slippage and cost over-runs, as the project moves throughout its life-cycle. Projects tend to exhibit counter-intuitive behaviour. For example, adding more staff often delays the project even more. This type of behaviour is generated by typical feedback loops, such as: the more the project is late the more staff is hired; this causes training and communication overheads leading to lower the overall productivity; as a result the project is delayed even more. Many of the key factors in projects responsible for this type of loops are subjective and of human nature. Most of the interrelationships and causal effects within a project are non-linear and time-lagged (e.g. consider, intuitively, the effect of schedule pressure on productivity). Most of the critical problems in projects result from poor strategic management of these factors, and hence a high-level holistic perspective is required to analyse them.

Projects are complex systems and exhibit the type of problems which SD is aimed at modelling (Cooper 1996). Project management is therefore an ideal arena to apply the methodology. Indeed, over the last two decades there has been considerable number of applications in this field. These are reviewed and discussed in the following sections.

Motivation

The System Dynamics approach to project management is based on a holistic view of the project management process, focusing on the feedback loops that take place within the project system. SD offers a rigorous approach for the description, exploration and analysis of project systems, which are mainly comprised of organisational elements, the project work packages, the project work processes, and the environmental influences. In a typical SD application, the project management process is considered in a wide context, which includes many soft factors often external to the project work. There is a strong focus on human factors which appear to dominate the feedback processes.

In general, the application of System Dynamics to project management has been motivated by various factors (Rodrigues and Bowers 1996a) , in particular:

- a concern to consider the whole project rather than a sum of individual elements (the holistic approach);
- the need to examine major non-linear aspects typically described by balancing or reinforcing feedback loops;
- a need for a flexible project model which offers a laboratory for experiments with management's options;
- the failure of traditional analytic tools to solve all project management problems and the desire to experiment with something new.

An overview of past applications

Some of the major developments of System Dynamics in project management are summarised in table 2.2, which includes a representative sample of studies drawn from a previous exhaustive review (Rodrigues 1994, Rodrigues and Bowers 1996a). The first description of the use of System Dynamics to analyse projects appeared in 1964, but it was not until the 1980's that the first project specific applications began to be reported. Since 1990 there have been many more reported examples of the use of System Dynamics in project management. Typically, the application areas are those where budgets are high and the risks greater, such as the aerospace industry, large-scale software development projects and recently the Channel Tunnel.

A first model was proposed by Roberts (1964) to explore the basic dynamics of R&D projects, where the concepts of *perceived* progress and real progress were first introduced, addressing explicitly the fact that managerial decisions are based on perceptions which may be at significant variance with reality. This model was further improved by Kelly (1970) to consider the management of concurrent projects.

Author	Project Type	Problems Addressed
Roberts (1964)	R&D	- Dynamics of R&D: perceptions *vs.* reality (1)
Kelly (1970)	R&D	- Dynamics of R&D among concurrent projects
Cooper (1980)	Large Design and Construction Program	- (1) - Rework cycle: staff productivity, work quality, rework generation and discovery (2) - Staff hiring and manpower allocation (3) - Material acquisition and allocation - High-level work scheduling - Progress monitoring - Interdependencies between work phases: schedule slippage, work quality, resource sharing - Interdependencies among concurrent programs
Richardson, Pugh (1981)	R&D	- (1) - Productivity and Rework generation (2) - Policy of hiring staff: increase workforce *vs.* schedule slippage (3)
Jessen (1988)	R&D, Construction, Decision-Support	- (1), (2), (3) - Project team motivation *vs.* productivity - Client and project team relationship
Keloharju, Wolstenholme (1989)	R&D	- (1), (2), (3) - Cost-time trade-off
Abdel-Hamid, Madnick (1991)	Software Development	- (1), (2), (3) - Cost and Schedule estimations - Quality assurance policies
Abdel-Hamid (1988; 1989; 1992; 1993)	Software Development	- (1), (2) - Project staffing policies (3) - Multiproject staffing policies - Multiproject scheduling - 90% syndrome - Quality assurance policies - Cost and schedule estimations - Managerial turnover/succession
Barlas, Bayraktutar (1992)	Software Development	An interactive simulation game to evaluate staffing policies (3) in quality assurance and rework (2)
Cooper (1993, 1994), Mullen (1993)	Programs, Defence and Commercial Software Development	- The rework cycle: quality, productivity, and time to discover rework (2) - Project Monitoring (1)
Pugh-Roberts Associates (1993)	Large Design and Production Programs	PMMS – a software simulation tool: - Diagnosis of over-runs - Impact of design and workscope changes - Estimation of cost and duration of on-going programs - Risk analysis of prospective programs - Effectiveness of management strategies The models focus on: resource acquisition and allocation (3), high-level work scheduling, progress monitoring(1), rework cycle (2)
Ford, David (1996)	Product development	The impact of product development policies and processes on project performance (1), (2)
Williams, Eden, Ackerman, Tait (1995a; 1995b)	Large Design and Manufacturing	*Post mortem* diagnosis for dispute resolution (delay and disruption). Analysis of the 'vicious circles' of parallelism and their role on the impacts of design changes and delays.

Table 2.2 – Summary of some work and research developed since 1964 on the application of System Dynamics to Project Management.

The model develop by Cooper (1980) at Pugh-Roberts Associates was the first major practical application of System Dynamics to Project Management in the real world. The model was first developed as a post mortem diagnosis tool to support a major claim of "delay and disruption" in a large-scale shipbuilding program. Further versions of the model were developed and used to support the strategic analysis of prospective shipbuilding programs. The major novelty of this work were the concept of the *rework cycle*, a structure at the core of the model which incorporates explicitly the concepts of undiscovered rework, time to discover rework, work quality, and varying staff productivity. Another important novelty was the concept of *monitoring ramps*, an interesting XY plot of the "real progress" against the "perceived progress". These concepts consider explicitly that rework is generated in the project, remains undiscovered until the later stages, and is then discovered re-joining the initial pool of "work to do". The consequent gap between the perceived and the real progress explains the occurrence of the "90% syndrome" (see Cooper 1994 and Cooper and Mullen 1993 for details). These interesting concepts played an important role in subsequent models.

Richardson and Pugh (1981) presented a model for the management of R&D projects, which summarises the basic feedback structures of the project management process, and focuses on the trade-off between the managerial decisions of allowing schedule slippage and hiring more staff, in the face of delays that always emerge throughout the project.

The model developed by Cooper (1980) was the basis for the development of the Program Management Modelling System (PMMS) at Pugh-Roberts Associates (1993). This is perhaps the most complex model developed so far as and is currently being used to support the management of several large programs. Pugh-Roberts (1993) claims that the PMMS is a flexible system that provides a way of capturing the project work structure: a model is developed based on generic "building blocks" to capture the major project activities. This includes specialised sub-models for design, construction, procurement, testing, staffing categories, and program management. The procedures used to apply the model in practice are based on the calibration for a "as occurred" scenario, followed by "what-if" analysis where disturbances are introduced. In some reported applications (Cooper and Mullen 1993), the models developed using the PMMS were set to recreate the past behaviour of completed projects, and quantitative measures for the factors "work quality" and "time to discover rework" were extracted as project performance "benchmarks". Their findings suggest that gains in project performance can only be achieved by directing efforts to increase work quality and to detect errors earlier. Although the major practical applications reported in the literature refer to cases of post mortem analysis, these authors claims that the models have also been used with success to support the management of large on-going programs where a relevant past is available.

Other studies have examined the dynamics of specific types of projects, the most relevant being the models developed by Abdel-Hamid and Madnick (1991), used for post mortem diagnosis of a software development project, Ford (1996) used to analyse the importance of the product development process, and in Europe the work developed by Williams et al (1995a, 1995b), used to the support a major litigation case. Over the years, both academic and practical work has continued to be developed, while the usefulness of System Dynamics in project management has been gaining considerable recognition

The major three problem areas addressed by the models indicated in table 2.2 are: (1) project monitoring and control, (2) rework generation, and (3) human resource management. These System Dynamics studies were undertaken in R&D, software development, and design and development applications, perhaps reflecting the relatively high proportion of project failures in these areas and an accepted need to improve management methods. Also important to note is that these three categories of projects can be characterised as being of continuous nature, contrasting with other more discrete-type of projects. The work developed by Williams (1995a; 1995b) is more singular and of particular interest, using a System Dynamics model for a post mortem diagnosis in which the project behaviour is described under a network perspective. It identifies important feedback processes responsible for the vicious circles of parallelism: work being developed in parallel increases cross-relations between concurrent activities, increasing activities' duration and hence prompting a revision of the plan to incorporate yet more parallelism, as the project struggles to achieve its original completion date.

While most projects experience problems in the above three problem areas, and the interactions between the three can be critical, each is now considered separately for the sake of clarity. The discussion below reflects the results from the analysis of the applications listed in table 2.2.

Project control

The basic cycle of the project control model is indicated by the sequence of broad arrows in figure 2.3. Management respond to a perceived slippage in the schedule by deploying more resources, typically staff, in the hope of increasing the progress rate. This action should reduces the perceived effort remaining and eventually brings the forecasted completion date forward thereby eliminating the slippage. An alternative response to a perceived slippage is simply to adjust the schedule, as suggested by the top cycle of figure 2.13. However, there are many disruptive factors which might prevent the effective employment of more resources. Some of the factors influencing the employment of human resources are discussed below. Other disruptive factors can influence other stages of the cycle; political factors can be particularly important, encouraging a too optimistic view of the project's useful progress thereby restricting adjustments to the schedule.

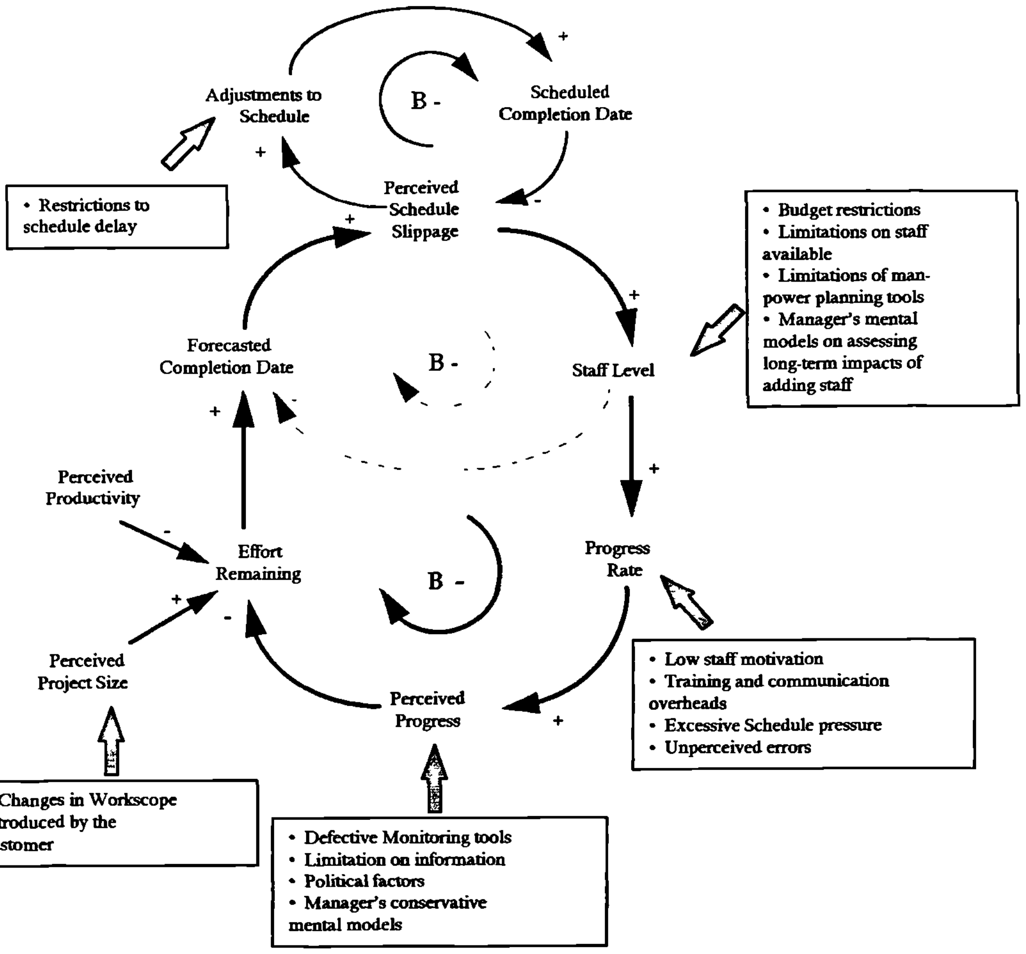

Adjustments to Schedule
Scheduled Completion Date
B -
Perceived Schedule Slippage
Forecasted Completion Date
B -
Staff Level
Perceived Productivity
Effort Remaining
B -
Progress Rate
Perceived Project Size
Perceived Progress
• Restrictions to schedule delay
• Budget restrictions
• Limitations on staff available
• Limitations of man-power planning tools
• Manager's mental models on assessing long-term impacts of adding staff
• Low staff motivation
• Training and communication overheads
• Excessive Schedule pressure
• Unperceived errors
• Changes in Workscope introduced by the customer
• Defective Monitoring tools
• Limitation on information
• Political factors
• Manager's conservative mental models

Figure 2.3 – The basic project control cycle as considered in a SD model

The rework cycle

The difference between the perceived and actual progress is explored in more detail in the rework cycle illustrated in figure 2.4. The work rate is determined by the availability of resources and their productivity. As time advances so the amount of work remaining should reduce. However, the quality of the work may not be perfect and errors may be generated. After some time, these errors are detected and rework is identified, increasing the amount of work remaining. The amount of rework required will also be dependent on the age of the errors; if the error is fundamental, necessitating an important specification change, all the perceived progress subsequent to the error may be wasted. The main cycle loop in this structure is a reinforcing loop (indicated by "R+"), with more work generating more errors and more work in turn, though the two balancing loops should help counteract this accumulation of work remaining. The gap between the perceived and actual progress can be difficult to close; it may appear that all is nearly finished but the project can remain obstinately at the 90% completion level – this phenomenon is usually refereed to as the "90% syndrome" (Cooper 1993, Abdel-Hamid 1988).

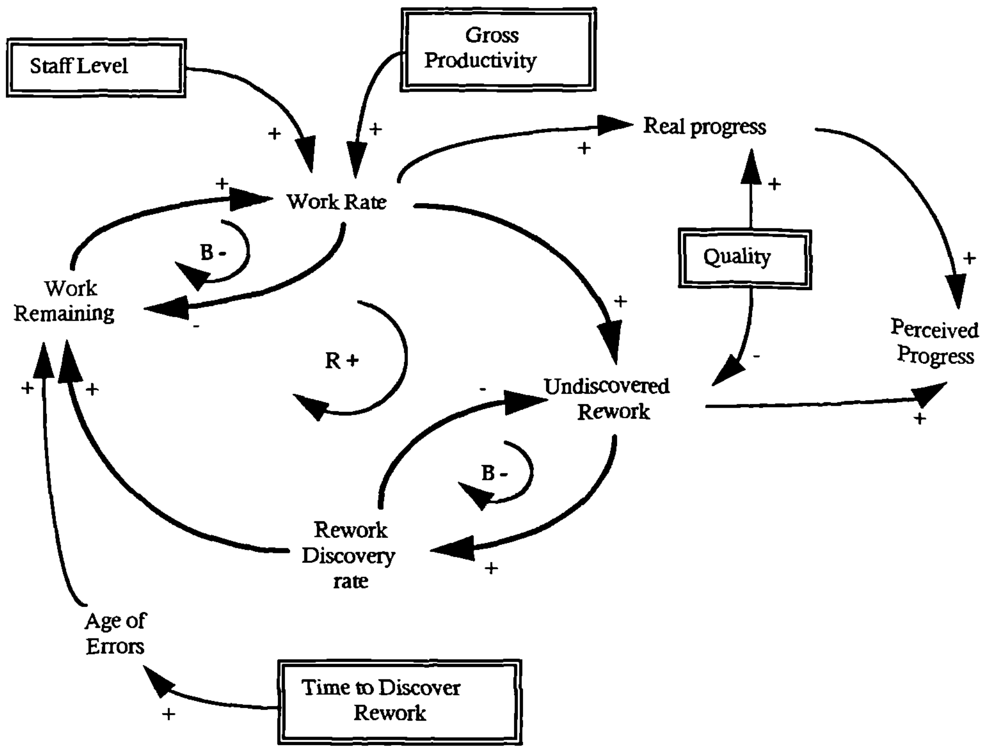


Figure 2.4 – The *rework cycle* (represented by an ID) as considered in a SD project model

The rework cycle structure identifies four factors partially under management control: resource level, productivity, quality and the error discovery time. Typically, management focus on the resource level and productivity as the keys to successful implementation. However, experiments with the rework cycle suggest that the quality and the error discovery rate are the more important factors (Cooper and Mullen 1993). Simply throwing resources at the project does not solve the fundamental problems; a more effective approach should be to reduce the number of errors, or at least the time for their detection. Such a conclusion is not unexpected, but system dynamics provides a deeper understanding of its background and hence a greater possibility that the message will be taken more seriously by the project team.

Human resource management and project control

The basic project control cycle of figure 2.3 included reference to various factors that might limit the effectiveness of employing more resources in order to accelerate a project. These factors are examined in the context of human resource management in the cycle of figure 2.5. The cycle assumes that additional staff may be recruited in response to a perceived schedule slippage, though different managers will have different approaches with some readily hiring and firing while others prefer a more stable policy. While hiring additional staff can be valuable, there are secondary, negative effects which often conspire to reduce the effectiveness of such action, indeed the immediate result of recruitment may well be a reduction in the real progress rate (Abdel-Hamid 1989). Figure 2.15 includes the training requirements of the new staff: learning curves are incorporated into the model with the work rate of the new recruits taking some time to achieve that of the experienced personnel. The model also includes the reduced availability of existing staff while they devote a proportion of their time to training activities. Communication overheads can be included explicitly, reflecting the need for more bureaucracy and configuration control: as the number of staff involved grows the effective availability of each member of the project team may reduce, as the project is ever sub-divided and time has to be devoted to discussions and the checking of design changes. Such a negative feedback reduces the expected impact of the additional staff and contributes to the possibility that their recruitment may even reduce the project's progress rate.

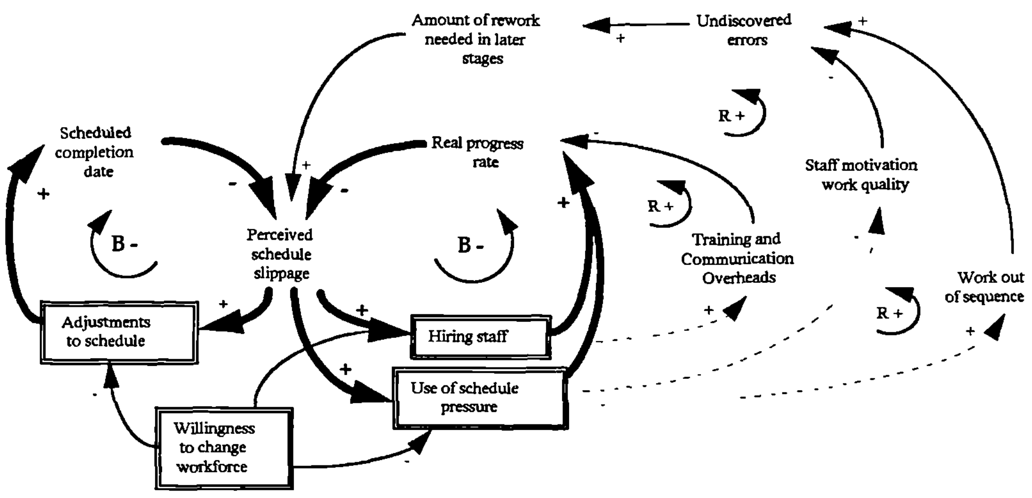

Amount of rework needed in later stages
Undiscovered errors
Scheduled completion date
Real progress rate
Staff motivation work quality
R+
R+
B-
B-
Perceived schedule slippage
Training and Communication Overheads
Work out of sequence
R+
Adjustments to schedule
Hiring staff
Use of schedule pressure
Willingness to change workforce

Figure 2.5 – Human resource management and the project control in a SD project model

An alternative approach to improving the project's progress might be to place staff under greater schedule pressure. Again this should increase the progress rate but there can be critical secondary effects. Staff motivation and the quality of their work may fall and there can be a tendency to undertake work out of sequence in the struggle to achieve the new performance targets. The consequences of these effects would be an increase in error generation and hence rework, ultimately resulting in further schedule slippage. Again, SD models have the capability to include many important, though difficult factors explicitly.

Range of applications

According to the past applications reviewed in table 2.2, the use of System Dynamics models in project management has been implemented in different types of industries, and within a range a various types of application.

The initial project models developed were aimed at helping to understand the general dynamics of R&D projects, within a competitive market (Roberts 1964, Kelly 1970). Further work (Richardson and Pugh 1981, Keloharju and Wolstenholme 1989, Barlas and Bayraktutar 1992) focused on this type of high level models, which were developed to work within fictitious project scenarios where there was no need to match the of history specific real projects. These applications were primarily aimed at providing clearer explanations of how projects work and how managerial policies affect their performance, based on the causal-descriptive feedback perspective of the System Dynamics approach. In practical terms, models developed in this way are typically used as "flight simulators" to train managers and support policy improvement through "what-if" analysis which take within fictitious project scenarios.

The major practical applications emerged in the arena of dispute resolution and post-mortem diagnosis (Cooper 1980, Abdel-Hamid and Madnick 1991, Cooper and Mullen 1993, Ford 1996, Williams et al 1995, Ackerman et al 1997). In this type application a SD model is developed to recreate the past behaviour of a real project, and is further used to explore the causes for the observed over-runs. The main aim of this exercise is to identify and quantify the responsibilities of the different parties involved in the project. This is the type of application where System Dynamics has given the most convincing evidence regarding practical credibility, in particular the cases reported by Cooper (1980) and Williams et al (1995).

Although some authors claim the appropriateness of specific SD project models to support the management of prospective projects (e.g. Lin and Levary 1989), the effective application to real projects prior and during implementation has only been claimed by Pugh-Roberts Associates (1993), through the use of their PMMS tool. In this type of application, a PMMS based SD model is used to *forecast* possible future outcomes for a real project within specific scenarios. Predicting about the future is indeed the most ambitious claim of a SD application, since it demands a high level of confidence in terms of model validation. The most strong argument about the PMMS is its successful application in a large number of real projects, where its authors claim it has produced very accurate results, often above the traditional tools (Cooper 1996). Furthermore, in this type of applications a relevant past of the project was available, where problems had occurred, and this was reproduced by the model prior to forecasting. Nevertheless, it should also be noted that these applications took place within a consulting environment. Explicit materials explaining such studies have never been made available to the public domain, in particular regarding model validation. This certainly restrains the credibility of the claims, regarding their scientific value.

The range for the various types of application of SD project models can be defined based on different criteria. Cooper (1996) proposes four main categories:

- pre-project bidding and risk analysis: from a previous application, a model calibrated to a similar type of project is used as a base-line to model a prospective new project. The model is used to forecast costs, schedules, and resource requirements for this new project. The model is also used to assess the impacts of risk occurrences;
- mid-project management support: when a project exhibits problems throughout the life-cycle, a model is set to recreate the present scenario and further to assess the performance of possible corrective actions, anticipating future scenarios. This is probably the most challenging applications because of its real-time requirements. Not surprisingly, this has hardly been done apart from possible consulting claims;
- post-project diagnosis and dispute resolution: in this typical application a model is used to recreate the history of the project, where major over-runs occurred. Through retrospective "what-if" analysis the model is used to help identifying and quantifying the responsibilities for over-run of the different parties involved in the project;
- training and improvement: in this application a model is used to recreate a fictitious or a past project. The model is then used to investigate and assess purely fictitious scenarios. The main aim of this type of application is to create learning environments where managers can conceive, test, and assess the performance of alternative policies, thereby improve their current project management skills. Cooper (1996) stresses that in this type of application a SD project model can be purely qualitative (e.g. causal maps).

This classification implicitly considers differences regarding the type of environment, the main purpose of the application, and the level of analysis. Two main types of environments can be considered: (i) the project model is being used to analyse a real project, or (ii) the model represents a purely fictitious project. The purpose of an application can vary considerably from case to case, but in general the main purpose falls into two categories: (a) the model is developed to represent the general dynamics of how projects work and is used as a learning tool for policy improvement, or (b) the model is developed to represent a specific project within real scenario and is used to produce quantitative estimates or forecasts to be used in practice. Finally, two levels of analysis can be considered: causal analysis using qualitative influence diagrams, or quantitative analysis using simulation models. These differences are interrelated and have to do with the way in which the SD process is implemented, as discussed in the previous sub-section 2.3.1. Table 2.3 below provides a generic classification of a SD project model according to these three criteria.

Type of environment	Main purpose	Level of analysis
1. Fictitious project	1. Policy improvement	1. Qualitative analysis
2. Real project 2.1 prior to implementation 2.2 on-going 2.3 post mortem	2. Estimating or forecasting	2. Quantitative analysis: 2.1 high-level 2.2 detailed

Table 2.3 – Generic classification of the range of applications for SD project models

This classification is not intended to impose a strict relationship between the three criteria. As an example, a qualitative influence diagram can be developed to help understanding the problems of a real on-going project and support current policies. However, the suggested relationships are according to most of the practical implementations.

2.3.3 A comparative analysis with the traditional models

A review of the traditional approach

The traditional approach to project management was reviewed in section 2.2. For the sake of comparison with the SD based approach, an overview is here provided.

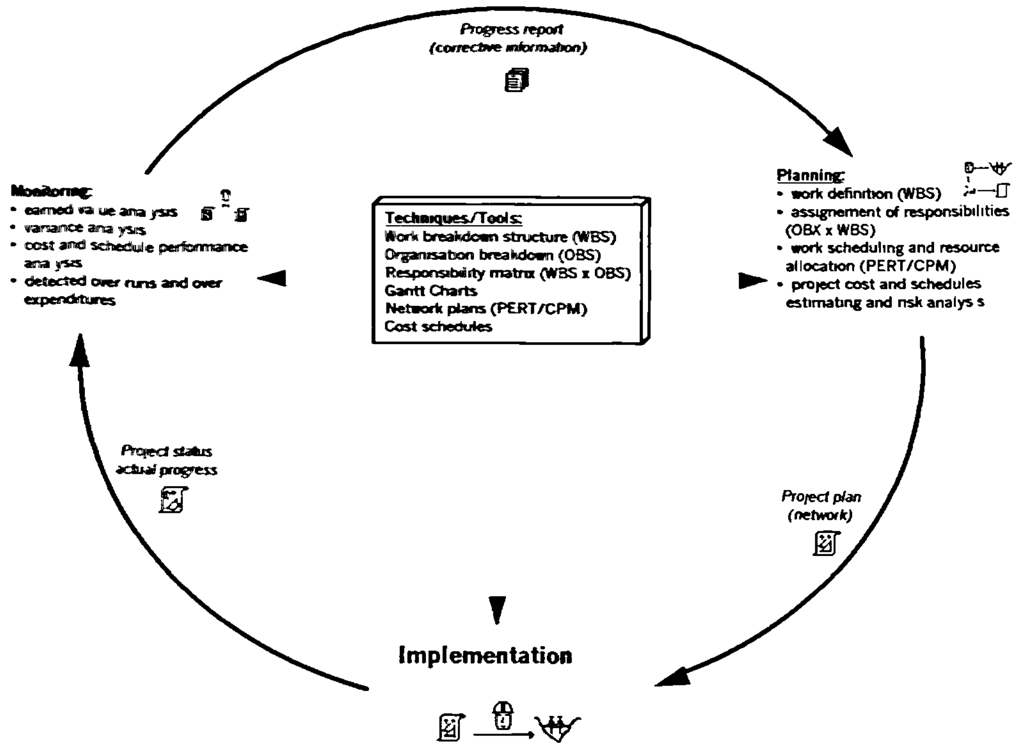
Over the years, a large collection of techniques has been developed in response to the practical problems of project implementation. These techniques focus on the definition of the project work structure and the production of detailed schedules and budgets for monitoring and controlling performance throughout the project life cycle (Nicholas 1990).

The traditional techniques are founded on the premise that while each project may be unique many of the constituent elements have been experienced before. The project work is therefore decomposed into elements, for example activities, which can be individually related to previous experience. This is typically represented in the work breakdown structure (WBS). It is then possible to produce reasonable estimates for each element's duration, cost and resource requirements. The logic of the project, is then represented in a network plan (typically referred to as PERT/CPM networks), where the work packages in the WBS are related according to their precedence dependencies. The identified inter-relationships supply the basis for reconstructing the project from its parts and deducing the whole project's duration, cost and resource requirements from those of its elements.

The traditional approach assumes a well ordered project that progresses in well defined stages to completion. This may suggest a strictly ordered project. This view also implies an assumption that all the required information is available at the start of the project, allowing the design of an optimal plan where the only concern of management is to keep the project on the specified track. One of the concerns about such an approach is that while the estimates for the individual elements may be very accurate, the reconstruction of the project may ignore important intra-project relationships and interactions. In many cases, this imposed discrete view, inherited from the construction industry, might not be appropriate to model the more continuous nature of design and development type of projects.

In practice management need to be dynamic, responding to new information and adapting the plan rather than keeping rigidly to the original. When implemented properly, the traditional methods are used in a more responsive manner, deployed within the dynamic environment of the classical control feedback loop: the original plan is used to set targets which are then compared to progress and where there is a significant deviation, action is taken including revisions to the project plan.

While the individual tools might be very linear in nature, the overall framework of traditional project control exhibits the classic characteristics of a dynamic system. The ideal of the traditional approach is based on a systems' methodology: the classic control cycle. It considers that project management is based on a dynamic control process that takes place within a project system and interacts with the external environment: the project system comprises a human organisation, called the project organisation, and a sub-system of materials, equipment and facilities. The project organisation is integrated with the project work structure, providing the assignment of responsibilities to the people involved in the project. Control and planning are continuously practised through the use of various techniques and tools, while the implementation process proceeds, as shown in figure 2.6 below.



Progress report
(corrective information)
Monitoring:
• earned value analysis
• variance analysis
• cost and schedule performance analysis
• detected over runs and over expenditures
Techniques/Tools:
Work breakdown structure (WBS)
Organisation breakdown (OBS)
Responsibility matrix (WBS x OBS)
Gantt Charts
Network plans (PERT/CPM)
Cost schedules
Planning:
• work definition (WBS)
• assignement of responsibilities (OBX x WBS)
• work scheduling and resource allocation (PERT/CPM)
• project cost and schedules estimating and risk analysis
Project status
actual progress
Project plan
(network)
Implementation

Figure 2.6 – The dynamic perspective of the traditional approach to Project Management

Nevertheless, while the underlying process is of dynamic nature, the traditional tools employed within struggle to incorporate many of the important non-linear project dynamics. While these tools can be adapted, they do not encourage managers to examine the feedback loops which rule a project's dynamics. There are many accounts of problems escalating in a project, with the knock-on effects producing unexpectedly dramatic overrun and overspend: in most cases these are examples of undesirable positive feedback. The failure to consider these project dynamics may be one reason behind the general record of project overruns, as discussed in section 2.2.

The System Dynamics perspective

The System Dynamics application to project management is normally centred around the development of a *project model*. The study is based on a holistic perspective of the feedback processes, and the model aims to incorporate the major forces that drive the general project behaviour.

The core feedback structure captured by a typical project model includes two major control loops and several longer-term reinforcing loops, as shown in figure 2.7. The major characteristic of a project in terms of management, is its "natural" tendency to deviate from the initial targets, as the work progresses throughout the life-cycle. This can be caused by internal problems, like low productivity, or by disruptive factors external to the project, like the "Client" requiring workscope changes. Generally, these deviations have an immediate impact on the project schedule, delaying the completion date. In order to balance this slippage, managers can follow two different routes of action: (1) negotiate schedule extensions with the "Client", or (2) implement several actions with the intent of increasing the work rate within the project, like using staff over time, schedule pressure, recruiting extra staff, increasing work concurrency, or even reducing the intensity of quality assurance (QA) and control (QC) activities. In practice, management often compromises between these two routes of action. Delaying the schedule creates the typical control loop in which the desired target is readjusted to match the actual system's output. Conversely, increasing the work rate creates the second typical control loop in which there is an attempt to readjust the actual system's output towards the desired target. In order to help the manager in implementing effectively this control mechanism, the traditional approach includes a wide collection of tools and techniques that support monitoring and re-planning, as discussed in section 2.2. A SD project model captures within its feedback structure these control mechanisms.

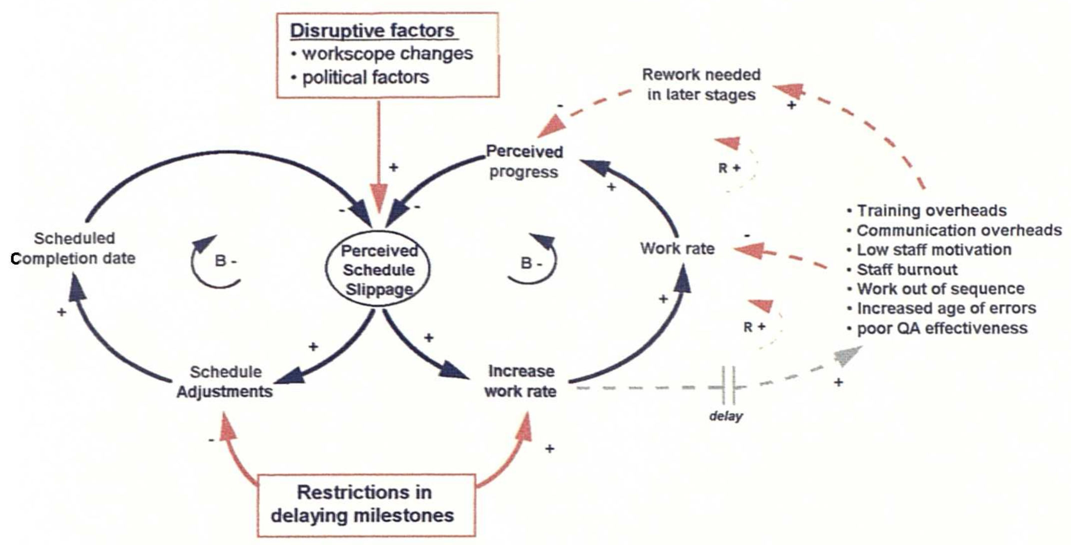


Figure 2.7 – The generic feedback perspective of the SD approach to Project Management

However, in reality there are several indirect and often subjective factors that tend to disrupt the performance of this mechanism in two possible ways: (1) countering directly the desired effects (like training and communication overheads reducing the actual overall productivity), or (2) by creating the need for an enormous amount of rework in the later stages, and for which contingency was not made. In both cases, these factors and effects generate degenerative loops which in the long-term will reinforce the initial problem (the so-called "vicious circles"). The feedback structure of a SD project model also captures explicitly these reinforcing loops, which in reality are the responsible for over-runs.

The high level feedback structure of a project as captured by a SD model is shown in figure 2.7: hiring more staff has the indirect effect of increasing communication and training overheads, which will counter the desired increase in the work rate. These long-term impacts are of course difficult to quantify and hence are not considered explicitly in the traditional models. Another typical effect is the increase on error generation, which then escape throughout the life-cycle to the final testing phase. Typically, this effect results from decisions intended to increase the work rates, like excessive schedule pressure and the consequent staff fatigue or the skipping of QA activities. When these errors are discovered, the enormous amount of rework required results in even worse over-runs. Political factors also create pressure towards optimistic progress monitoring, preventing the detection of problems in the early stages. By capturing both control and degenerative feedback loops, the System Dynamics perspective aims at providing a pragmatic and more realistic view of how projects behave in reality.

In order to incorporate the project feedback structure, a SD model normally integrates the two process levels of management and of engineering. The engineering processes consists of the physical activities responsible for the development of the output product, while the management process incorporates the monitoring and planning functions responsible to keep the work on target. Figure 2.8 provides a representation of this integrated perspective. The modelling of the engineering process is based on the three main assumptions: (1) the quality of the work developed first time is not perfect and hence errors are eventually generated, (2) these errors tend to remain unperceived during a certain period of time, and hence represent a certain amount of undiscovered rework, (3) when errors are detected this rework is discovered joining the pool of work remaining in the project. These three assumptions are the basis of the *rework cycle* proposed by Cooper (1993), as discussed previously. The impact of these assumptions on the management process is that the perceived progress, incorporating both "work done" and "undiscovered rework" (thought to have no errors), differs from the real progress *by excess*. The gap is represented by the *monitoring ramps*, a concept also proposed by Cooper (1993), as shown in figure 2.8: in a situation of perfect monitoring, the perceived and real progress would be exactly the same (dashed line), however when the perceived progress reaches 50% the real progress might fall in a lower range of say 20% to 30% (grey area read in the vertical axis). Finally, when the perceived completion reaches 90%, further progress seems to be extremely slow because real progress is actually being achieved in the rework of errors (i.e. the "90% syndrome"). The structure of a SD project model therefore captures how project control is implemented in the real world, where decisions have to be planned on the basis of imperfect progress monitoring information.

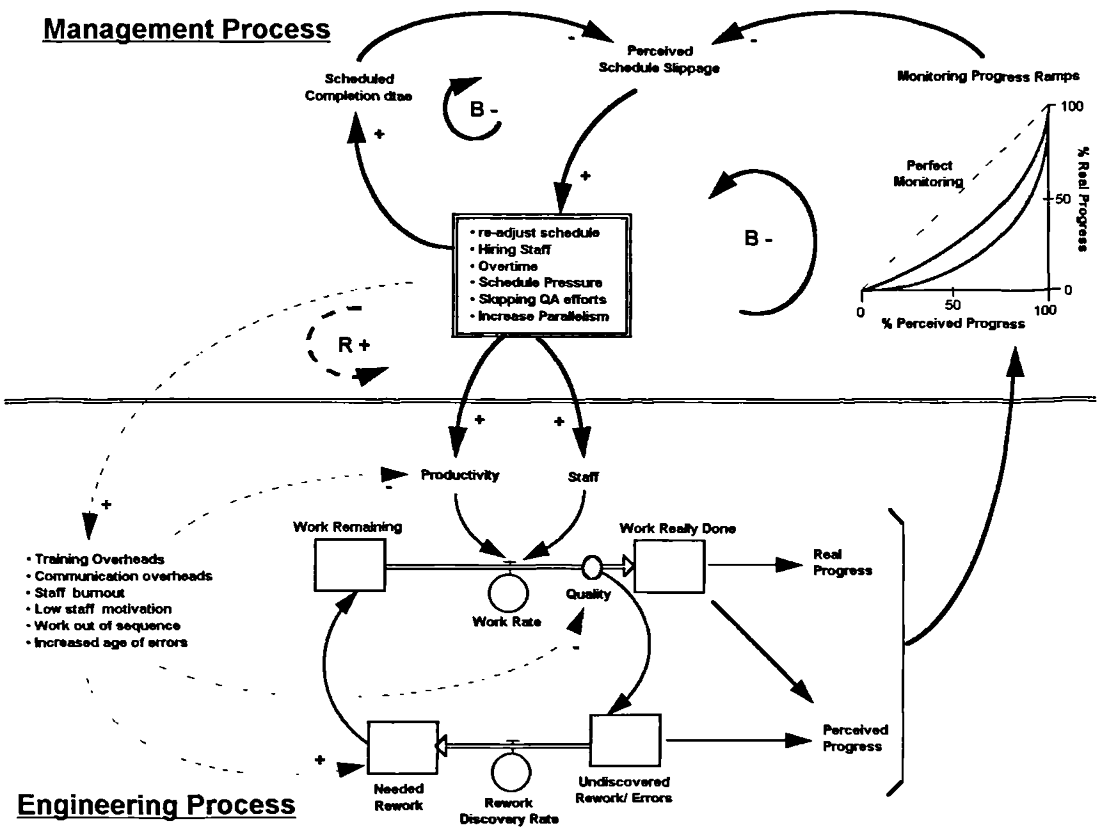


Figure 2.8 – The integrated perspective of management and engineering in a SD project model

Finally, another interesting feature of the SD approach is the perspective of project estimating. In the traditional approach, cost and schedule estimates are seen an "inevitable" result of accumulating the expected outcomes of the individual project tasks: given the amount work to be accomplished, the technical dependencies amongst the project tasks, and the resource availability and allocation, the project duration and cost emerge as an "arithmetical" consequence. On the other hand, in the SD perspective the initial estimates made for the project, which are the basis for continuous planning and control, have a significant impact on the project outcome and hence they are seen not just as a consequence but also as a *cause* of the project outcome: different estimates create different projects (Abdel-Hamid 1990). It follows that historical databases of past projects which are used to derive early estimates contain *raw* historical data which may reflect poor management practices. Therefore, fitting to this data, usually seen as "normal", may in the long term result as an obstacle to organisational learning and to the improvement of project management practices; in short, it encourages the "self-fulfilling" persistence of bad results. This phenomenon is represented in the ID of figure 2.9: the use of defective estimates leads to an over-estimated cost and hence to project over-staffing. The consequence is the creation of wasteful processes within the project plans to fill the available resources (the so called "Parkinson law", see Abdel-Hamid 1990). At the end of the project, the poor productivity levels and the unnecessary project "work size", both reflecting excessive overheads, are updated into historical databases which are used for estimating in future projects. A SD project model considers explicitly the initial estimates as one of the causes of the project outcome, and hence it can be used to test their appropriateness and filter-out "inflated" estimates.

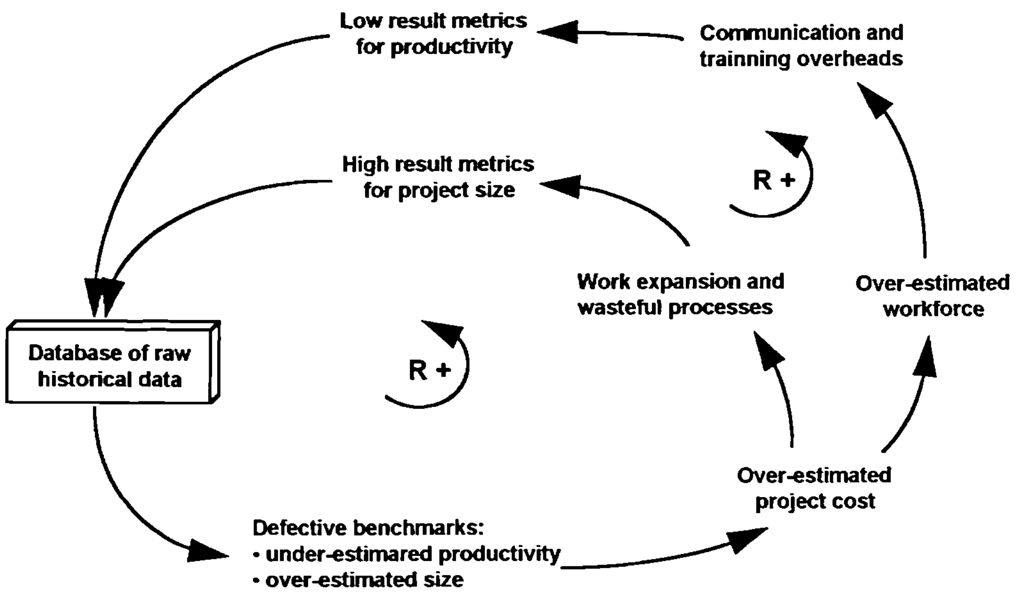

Low result metrics for productivity
Communication and trainning overheads
High result metrics for project size
R +
Work expansion and wasteful processes
Over-estimated workforce
Database of raw historical data
R +
Over-estimated project cost
Defective benchmarks:
• under-estimared productivity
• over-estimated size

Figure 2.9 – The self-fulfilling "vicious circle" of poor project estimating

In summary, the SD approach clearly moves away from the traditional view of project management, assuming a distinctive and more pragmatic perspective. A SD model aims at capturing the several features of the real world that make the task of project management a complex one: in the first place, the monitoring information about the project status, upon which control and planning are implemented, is considered to differ from the "real" reality; secondly, a SD project model considers that the project is exposed to unpredictable exogenous disturbances, which affect progress and restrict the scope for control decisions; finally, the *full* impact of control actions goes beyond the expected direct effects:

corrective decisions, intended to keep the project on target, generate secondary effects which ripple throughout the project, reinforcing the initial problems. The holistic perspective of a SD model integrates the two project processes of engineering and management, capturing the full impacts of changes in the project. This more realistic vision offered by System Dynamics provides an alternative method to address these types of problems to which the traditional approach has been failing to provide effective solutions, as discussed in section 2.2.

A comparative analysis

Both the system dynamics and the traditional approaches examine some common basic issues of Project Management, but from very different perspectives. The summary discussion below analyses these main differences, under the following perspectives: (i) the view of the project management process, (ii) modelling the project work, (iii) modelling the underlying influences, (iv) estimating the project outcome, (v) the managerial needs addressed, and (vi) the accessibility of the tools.

<u>The view of the project management process</u>

The ideal application of the traditional project management methodology and the System Dynamics approach, both consider project management as a dynamic process of planning, implementation, and monitoring, as illustrated in figure 2.6. Planning is concerned with the specification of the actions that have to be performed in order to implement the project according to the objectives. Monitoring is the process of assessing the project status and generates information for corrective actions in planning. According to this view, the project is continuously being assessed and re-planned as the work is being undertaken.

In traditional project management, tools and techniques such as the Work Breakdown Structure (WBS), PERT/CPM networks, and cost schedules, are employed within this dynamic process. They are dominated by the project WBS and the network plans (PERT/CPM based), based on the act of decomposing the project system into its constituent elements. Typically, a project plan includes in great detail:

- a definition of the work into tasks and sub-tasks;
- a work schedule specifying the timings for each work package;
- a resource schedule specifying the allocation of human and material resources among the project tasks;
- cost schedules that specify the capital requirements and support the estimation of budgets.

The assessment of the project status is based on the comparison of the current state of the work against the project plan. The corrective information generated to support re-planning specifies in detail the deviations. These may include schedule and cost over-runs of specific tasks and of the whole project.

In contrast, the primary objective of a System Dynamics model is to capture the major feedback "forces" responsible for the project behaviour, with less concern about the detailed project components. The project management process is put into a wider context, which includes the many soft factors often external to the project work but critical to its outcome. There is a strong focus on human factors as these are considered to dominate the feedback structures. This motivates the explicit consideration of a human resource management process, as shown in figure 2.10 below.

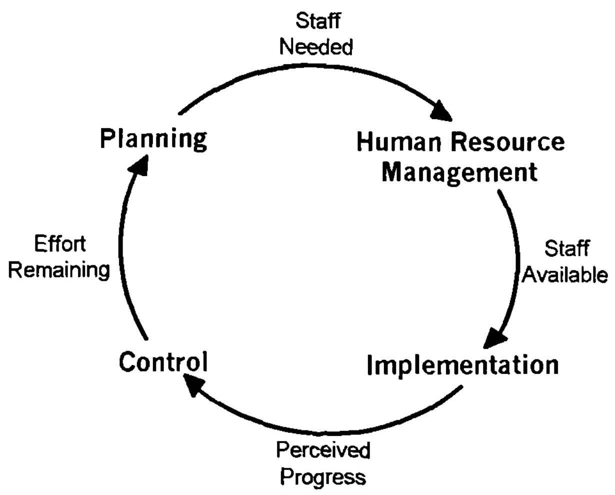


Figure 2.10 – The System Dynamics view of the project management process

The issues addressed in each of the four main project areas of activity can be summarised as follows:

- *planning*: examines the trade-off between delaying the project completion date and hiring more staff. The models explicitly incorporate managerial policies of staff hiring/firing and of schedule adjustment. These are often implemented through the definition of critical soft variables such as "willingness to change workforce" (Abdel-Hamid 1989; Keloharju 1989). The main output from these analyses is guidance in the allocation of additional staff to minimise schedule over-runs.
- *human resource management*: although traditionally this is part of the planning process, in a System Dynamics model it is considered separately and addresses several issues related with hiring more staff to the project. It usually includes factors such as workforce training, workforce experience level, workforce assimilation time, and communication overheads. This process is responsible for the generation of the actual level of staff working on the project.
- *implementation*: focuses on the problems associated with the generation of errors that remain unperceived. Cooper (1993a; 1994) addresses this problem through the definition of the *rework cycle* concept, discussed above. This process may address more complex rework problems such as customer delays in providing information and equipment, design changes, process changes imposed by the customer, and actual implementation of QA policies. This process is responsible for the generation of the actual work perceived accomplished.
- *control*: addresses the issues related with monitoring the project status. The difference between the perceived and the real project status is considered explicitly as a way of addressing the problems of the "90% syndrome". Managerial perceptions of productivity, quality, work completed, project size, and others, provide an estimation of the effort remaining which is used to plan project re-scheduling and staff allocation.

As expected, System Dynamics project models assume higher level view of the project management process focusing on human factors and managerial policies. They have an inherent flexibility which enables them to incorporate a wide range of influences specific to particular applications. The models used in the traditional approach focus on the project work structure and are more specialised, assuming a detailed view of the individual parts of the project system. These techniques are more rigid, enforcing a particular view of the project; this can ease their implementation but at the expense of some reality: while ensuring rigorous monitoring of the project past, their view of the future is focused on a "planned success". In contrast, System Dynamics simulation models provide a laboratory to test several different scenarios for the project, delivering a clearer and perhaps more realistic view of the possible futures.

Modelling the project work

One of the most important differences between traditional and System Dynamics approach is their approach to modelling project work. Although both assume that project implementation is based on the process of performing work through the employment of resources, they differ in the level of detail in which the work is considered and in the range of factors they address explicitly.

The traditional tools, such as Gantt charts and PERT/CPM, view the project work as the sum of a set of work packages or tasks, each scheduled according to their precedence relationships and resource requirements and availability.

In the Systems Dynamics approach the project work is modelled at a higher level and a holistic view is adopted. In general, the project work is represented by a continuous flow of units of work that flow from the initial state "to be done" to the final state "done", as the staff allocated to the project perform the work. There is no detailed consideration of what work is done when, and by whom.

Different SD models might consider this view at different levels of detail, by decomposing this flow of work into several phases or stages, according to the life-cycle of the specific project (Cooper 1980; Abdel-Hamid and Madnick 1991; Ford 1994, Rodrigues and Williams 1996). As an example, such a comprehensive model could be used to analyse whether allowing early design milestones to slip would have a beneficial impact on the overall project. Extra effort expended in improving the quality of the design might result in some initial project delay but this could be outweighed by savings in rework in the later development stages. The comprehensive model would enable a quantitative and rigorous assessment of the internal quality-time trade-offs. Nonetheless, the decomposition of this model is still far from a detailed consideration of the work responsibilities within the project, as in the traditional work breakdown structure and responsibility matrix. Instead the System Dynamics approach requires the input of an initial estimate, perhaps based on an approximate, high level, work breakdown analysis. With this estimate as a basis, a wide range of factors such as rework, changes in workscope, quality, productivity, and motivation, may then be built into the model.

Modelling the underlying influences

The power of the System Dynamics approach lies in its ability to incorporate the more subjective factors which can have an important influence on the whole project. Factors such as changes in workscope, quality, productivity, and motivation may be included and represented explicitly within causal feedback loops in the model. The System Dynamics model offers a language, using symbols and the concepts of feedback loops, to express these factors in a rigorous though qualitative manner and also the opportunity to incorporate quantitative approximations of their effects. The traditional approach focuses on the direct causes of the project outcome in great detail. A Systems Dynamics model does not provide such a detailed breakdown, but it can include the indirect causes that result from the feedback "forces", which are often responsible for over-run and overspend.

A traditional project network analysis can contain approximations to the effects of these underlying influences, either by use of simple models of tasks' duration including factors such as productivity or by employing more sophisticated network facilities, such as Q-GERT (Pritsker 1977, Nicholas 1990). However, the traditional tools can only achieve this by increased detail and complexity with the attendant danger of obscuring a project's important underlying truth. The System Dynamics approach is based on the premise that these underlying influences are the key to project management and deserve a much greater emphasis.

Estimating the project outcome

The fact that both approaches provide estimates for the project outcome, in particular for the final cost and duration, raises a potential conflict. Traditional models focus on a detailed view of the project work and thereby on evaluating possible alternatives they only assess the direct impacts on cost and time, while other higher order effects can be very important (Weil 1993). System Dynamics models focus on the feedback processes and assume a holistic view of the project. On evaluating possible alternatives they consider a wide range of subjective and disruptive factors, but by ignoring the detailed_logic of the work structure, as represented by a network, they may overlook important operational issues. This suggests that both the operational detail of the traditional approach and the systemic view incorporating the feedback processes are crucial for the generation of accurate estimations; a combined operational and System Dynamics model may therefore offer a useful approach to improving project estimates.

The effectiveness of the estimates produced by the traditional models depends on the validity of the many underlying assumptions, typically drawn from individual personal experience, such as the assumption of a particular productivity level for the staff. The assumptions provide a mechanism for handling subjective issues that are difficult to quantify, but they are often implicit and too readily taken for granted. The weakness of this more classic operational research approach is that the assumptions are not always applicable and can result in a model divorced from reality. This is particularly true when the analysis targets a complex social system such as a project.

A System Dynamics project model is often validated by comparison with past projects. As in any modelling exercise, even a perfect reproduction of past behaviour cannot guarantee the accurate forecasting of the behaviour of a new project. Projects are characterised by their uniqueness and particular caution is needed when extrapolating past experience, whatever the modelling methodology. While more evidence of the validity of System Dynamics models would be desirable, experiences in project management indicate that their holistic approach is valuable, avoiding a narrow, detailed view of those aspects which happen to be more readily quantified.

The managerial needs addressed

It is very important that the comparison of the traditional and System Dynamics approaches is performed within the context of requirements of project managers. As a basis for a comparison of the appropriateness of the two approaches in satisfying managerial requirements, the following issues can be considered: the nature of the managerial needs, the factors explicitly considered, the basic managerial decisions evaluated, the impacts of uncertain events addressed, and the project estimations provided. Table 2.4 and table 2.5 provide a brief summary of this analysis. While the traditional approach encapsulates a recognised set of project management tools, the System Dynamics approach is less well defined. In this comparison the System Dynamics assumes a compilation of all the models of table 2.2.

Nature of the Managerial Needs	**Traditional Approach**	**System Dynamics Approach**
Specification of the work (WBS)	Yes	No
Assignment of responsibilities to the work within the organisation	Yes	No
Work Scheduling	Yes (detailed)	No or high level (life-cycle phases/stages)
Resources management / scheduling	Yes	Yes - high level
Cost estimation / budgeting	Yes	Yes
Project control / monitoring	Yes	Yes - high level
Evaluate the impacts of major decisions	Yes (not effectively)	Yes
Evaluate the impacts of uncertain events	Yes (not effectively)	Yes
Post Mortem diagnosis	No (not practical)	Yes

Table 2.4 – The nature of the managerial needs addressed by the traditional and the System Dynamics approaches

Table 2.4 indicates that many of the basic managerial needs are addressed in both approaches. However, it is important to note that the level of detail of the analysis is different: traditional models suggest decisions focused on operational issues, while System Dynamics models focus on the strategic issues providing more general directions. They have the potential to consider a highly aggregated view of the project work structure and their focus on the causal feedback loops driving the project behaviour enables a high level model to be built without much detailed information about the past; this makes System Dynamics particularly useful in the diagnosis of historical cases, such as in supporting dispute resolutions (Cooper 1980; Williams et all 1995a; 1995b).

	Traditional Approach	**System Dynamics Approach**
Factors considered explicitly	- Logic of the work structure - Cost of resources - Indirect costs - Constraints on resources availability - Work resources requirements	- Quality of work performance - Staff Productivity - Staff Experience Level, Learning, and Training - Schedule pressure on the staff - Rework generation and discovery time - Mismatch of *perceptions* and *reality* - Staff motivation - Client and project team relationship
Managerial decisions	- Cost-time trade-off: crashing activities - Changes in the schedule of activities - Scheduling resources among activities - Changes in the logic of the project work structure	- Hiring Staff *vs.* delaying the project completion date - Introduction of new technologies - Effort on quality assurance - Effort on rework discovery time - Cost-time trade-off: hiring staff - Multiproject scheduling - Multiproject staff allocation - Managerial turnover/succession - Estimation of schedule and cost - Changes in the schedule of the project life-cycle phases/stages
Uncertain events	- Delays in the completion of activities - Constraints in the schedule of activities - Resource constraints - Uncertainty in the duration of the activities (simulation)	- Changes in the project workscope - Changes in quality and productivity levels - Customer/Vendor delays in delivering information - Constraints in the staff levels
Major Estimations	- Project duration - Project cost - Resource allocation	- Project duration - Project Cost - Staff allocation

Table 2.5 – Comparison of some important characteristics of the traditional and the System Dynamics approaches

Table 2.5 emphasises the ability of System Dynamics' models to consider a wide range of subjective factors that are often ignored in traditional operational models, or are only addressed by simplistic assumptions. The managerial decisions which they aim to support are complex and the possible use of the quantitative models of the traditional approach typically requires excessive effort; similar difficulties are experienced when traditional quantitative techniques are used to examine the effect of uncertainty on a project. While both approaches provide project estimations for cost and schedule, System Dynamics models assume a more aggregated view of the work, and a strategic perspective of management decisions.

<u>Accessibility</u>

System dynamics software is now both powerful and attractive to the non-specialist with packages such as iThink, Powersim and Vensim, offering sophisticated analyses combined with a effective graphical interface. However, whereas many people have sufficient understanding of the traditional tools to develop a project network, the System Dynamics approach does require the assistance of specialist skills. The additional costs associated with employing such skills may well limit the application of Systems Dynamics to large projects. However, some of the benefits can be accessed by the careful use of standard System Dynamics models, appropriately tuned to the particular project. Many of the lessons are generally applicable and some of the benefits may be obtained by considering checklists of problems and advice drawn up from past System Dynamics studies. However, such a course must be regarded as second best since seeing the evolution of the analysis and being involved in the model building process can provide a very effective education for management.

The distinctive contribution of System Dynamics

From the above discussion it is clear that System Dynamics offers a distinctive modelling perspective to project management. Both as a modelling method and as a practical tool, System Dynamics adds something new to the current Project Management Body of Knowledge (PMBOK 1996). In some aspects, the SD models provide an alternative way of analysing management problems within a project, like estimating the impacts of resource availability on the final schedule. Where a SD model can be an alternative, it is important to consider what are the strengths and weaknesses against the traditional models. In other aspects the SD method provides a new and unique type of analysis, impossible to implement with the traditional tools. An example by excellence is to assess the performance of management control policies: given an apparently reasonable plan, what is the best way of controlling its implementation so that the impacts of unexpected problems can be minimised?

As it will be seen later in this research, there are many other types of valuable project analysis unique to the System Dynamics approach. In terms of modelling, three main distinctive novelties can be identified at this stage:

(1) explicating the project outcome as the result of combined set of feedback loops or forces not captured in the traditional tools;
(2) incorporating explicitly soft variables and intangibles;
(3) incorporating explicitly management control policies, simulating their implementation, and assessing their performance.

Table 2.6 summarises the main differences between the traditional and System Dynamics approaches, and identifies the distinctive methodological contributions of the later.

In respect to the practical application of SD models within real project management environments, the preliminary conclusions drawn from this comparative study lead to the following scenario:

(1) SD contributions unique or above the traditional models:
    - causal analysis of project behaviour;
    - "flight simulators" used to train project managers;
    - effective support to litigation cases of dispute resolution;
    - assessment of high-level strategic decisions, control policies and risks;

(2) SD contributions which are alternatives to traditional models:
    - pre-project bidding, early estimating and preliminary risk analysis;
    - estimating and risk analysis while the project is underway.

Aspect	Traditional approach	System Dynamics	SD Contribution
Focus	Project work structure *The problem*	Project feedback structure *The situation*	*Holistic* perspective of Project Management explicated by the role of the feedback phenomenon
Level of detail	Highly decomposed based on top-down decomposition	Aggregated based on bottom-up aggregation	Clear vision of the whole project system, identifying the major long-term project influences and intra-forces
Factors considered explicitly	The readily quantified, directly related with the project work	The human and subjective, often intangible, which have a relevant role within the feedback processes	Identification of managerial assumptions otherwise implicit, and explicit incorporation of human issues
The key factor	The logic of the work structure and its resource requirements	The feedback processes and the role of management policies and staff behaviour	Assessment of how management policies and human behaviour impact on the project outcome
Project work	A set of distinct tasks interrelated through technological and resource dependencies	A continuous flow processed sequentially through overlapping activities, with the possibility of iteration	A dynamic view of the physical work processes considering explicitly the overlapping and iteration of activities
Nature of results	A one-step prediction of a "planned success"	Integrated simulation of reality: continuous interaction between work accomplishment and managerial control, often resulting in a "failed plan"	Explicit representation of the project behaviour eliciting its underlying dynamics
Main aim	To support detailed operational planning and progress monitoring	To support strategic policy analysis, identifying the major project trends	A "management laboratory" where the full impacts of high level decisions and risks can be assessed quickly

Table 2.6 – A summary of the differences and distinctive contributions of the System Dynamics approach to project management

Apart from offering a new and powerful type of modelling tool, System Dynamics also provides other secondary contributions, in particular helping communication within the project team. The advanced state of System Dynamics software now allows the analyst to develop models in the presence of management, encouraging the project team to participate in the modelling process. This direct involvement results in improved communication with the analyst and better understanding amongst the team: management can appreciate the effects of the rework or hiring feedback loops and relate the model directly to their own experiences. The graphical, quantitative output can be valuable but viewing the evolution of the model on the screen helps develop a deeper insight into the fundamental dynamics of the project. This process can also improve communication within the project team: the system dynamics approach encourages people to make their mental models explicit (Senge 1990), and thus share their understanding of the project. Often different people will have different mental models, reflecting their specific responsibilities and interests in the project. The object of the system dynamics analysis is to combine these individual models into one, which closer reflects the real world. It is important to emphasise that there is not a single "correct" mental model; each is personal view of reality and differences are inevitable, given the different priorities of the various players in the project. The objective, not always easily attained, is an open discussion with the different views of the project being expressed rather than an atmosphere in which no one is willing to state their understanding for fear of embarrassment.

System Dynamics: a different philosophy?

Any model of a project, whether it be a network, work breakdown structure or System Dynamics model, strives to deliver two types of information:

- an unbiased picture of reality, including best estimates of the project's duration, cost and resource requirements. The picture might not be attractive but it is the truth, including the many imperfections of the real project.
- a vision of the ideal but attainable project, including credible but ambitious targets for project control. This vision includes the concept of separate, well defined tasks all linking with a resolute logic to deliver the project's objectives.

In general, the traditional techniques are well suited to providing the second type of information. The rigid structure of the techniques requires management to break the project down into distinct elements with well defined interfaces. This might be a relatively artificial structure but if a project can be persuaded to adopt this form, management is much easier. As well as encouraging a disciplined project, the traditional techniques can provide motivating schedule and financial targets, though they can only be achieved if all goes strictly according to plan.

By comparison, the System Dynamics model emphasises the inadequacies of the real world. It can provide more accurate, unbiased estimates and also help management appreciate all that might go wrong. Such information is very valuable, helping senior project management understand a project's problems and in comparing alternative project strategies. While Systems dynamics has the laudable aim of providing a more realistic project model, sometimes a more visionary model is useful. Analogies may be drawn with the application of just-in-time methods in production processes: traditional stock control methods accept working practices and plan accordingly, while just-in-time offers the vision of a smooth running factory with no work-in-progress. Such a vision may not be quite attainable but it forces management to confront problems and strive for improvement. Similarly the simple, disciplined logic of the traditional project models represents a desirable aim, though it can never be achieved unless the discrepancies between projects' actual and planned performance are investigated, the problems identified and solved so that the next phase of the project, or the next project, might be closer to the ideal.

2.4 Conclusions and way ahead: an integrated approach?

The traditional view of project management has produced an undue focus on the project work (Turner 1993). There is a need to expand this view into a wider context reflecting the importance of the relationships between the individual elements of the project system and their effect on management performance. So far, these strategic issues have been handled implicitly, encapsulating personal experience in crude rules of thumb; the holistic approach offered by System Dynamics has emerged as an attempt for a more systematic approach.

System Dynamics analysis offers a distinctly different view of a project, with a key general output being a better understanding of the important underlying influences that take place within a project management system. These influences are interpreted under a "feedback-loop" perspective, and the overall analysis leads to the identification of counter-intuitive management practices, providing a basis for policy improvement. Although System Dynamics may also provide support in those areas where traditional tools and techniques are better established, it seems clear that its overall contribution is complementary, rather than a complete alternative: while the traditional models have not been able to cope with the complexity of the strategic issues (Cooper 1993a), the holistic perspective of System Dynamics, reinforced by the continuous nature of the simulation models, imposes a level of aggregation with no immediate translation into the more detailed world of operational actions.

There is now evidence that System Dynamics can provide valuable support at the strategic level; however, good project management practices will always require operational models to provide the detail necessary for the effective implementation of strategic decisions. It seems clear that both approaches are not incompatible and individually they look incomplete. The obvious corollary from this analysis is that if System Dynamics is to play a core role in the future of project management, the approach needs to be embedded within the traditional project management framework, eventually leading to a single, practical, integrated project management methodology. Various routes may be adopted to incorporating the two approaches:

- more sophisticated network based models can be developed, including the feedback processes and detailed mechanisms for modelling the tasks' duration and costs, in order to reflect the project underlying influences; such operational models should also be able to incorporate, more or less formally, quantitative data from higher-level System Dynamics models;
- more detailed System Dynamics models can be developed considering explicitly the project WBS, distinguishing its major tasks, phases, or stages as distinct activities, and employing different types of project work, resources and management processes;
- at a more informal level, the main lessons from System Dynamics studies could

be assimilated in a set of rules for use in estimating tasks' duration, costs, rework, risks and several uncertainties.

The first two routes depart from a different perspective but towards the common objective of a comprehensive project management methodology, incorporating all the best features of the System Dynamics and traditional models. A key issue of such an integrated approach, is to retain the dominant characteristic of a good System Dynamics study: its clarity and emphasis on the key issues. Allowing SD models to incorporate more detail can easily lead to a situation where the pertinence of a factor (i.e. its relevance) quickly gives place to its evidence (i.e. is it "really" there?). The resultant model complexity may not be justifiable, it can obscure the usefulness of the holistic vantage-point, or might even jeopardise the viability of the modelling process.

The third route fits within a new modern approach to strategic modelling where (simulation) models are used in a simple form to support strategic thinking, team reasoning, and create learning environments (Morecroft and Sterman 1994). This could involve experiments with Systems Dynamics models of a variety of projects in order to identify archetypal behaviours (Senge et al 1994), which would then be summarised as a set of rules; these rules could then be included in existing methodologies in different ways, like for estimating tasks' parameters (Bowers 1994). This offers a comparatively simple approach to propagating the benefits of Systems Dynamics, though at the expense of some pertinence to the particular project.

Whatever the route to follow, it is also desirable to maintain a continuous review and synthesis of System Dynamics developments and applications. It is important that valuable experiences can be shared among all those organisations involved in the use of this rather recent project management tool.

Furthermore, developing more complex network or System Dynamics project models on its own may not prove a fruitful effort, unless a well defined framework is available where the combined use of both type of models can be effectively articulated. It is argued in this research that in order to develop and apply both models within an integrated approach it is first necessary to establish and validate the foundations of such methodology. These general principles must be exercised, tested, and reviewed within a real world project environment, and only then further refined into a formal, System Dynamics-based, integrated methodology. This achievement is the main aim of this research. The route followed and the results achieved are described in the following chapters.

3. Proposed research methodology

3.1 Overview

This chapter provides a brief discussion about the methodological approach adopted to conduct the present research. A rationale is also provided.

In order to design an appropriate research approach, various issues must be taken into account. Of particular relevance are the aims of the research, the nature of the problem being addressed, and the system within which "a theory" is to be tested or developed. As it will be seen, the research approach adopted was predominantly phenomenological, or "social constructionist". The main reason for this was that the problem being addressed was in great part of social nature. Therefore, the aim of the research was to develop a theory based on fieldwork experience, as opposed to test an up-front theory conceived logically.

The next section 3.2 presents the research question, hypothesis, aim and objectives. The next section 3.3 provides a brief overview of some of the key methodological issues of management research, and relates them to the way in which the present research was conducted. Section 3.4 summarises the research approach adopted and explains why it was appropriate. Finally, section 3.5 describes a high-level work plan, which was the basis for the actual implementation of the research.

3.2 The research question, hypothesis, aim and objectives

3.2.1 Overview of the research background

The previous chapter 2 presented evidence that project failure is a current problem in project management. While projects became increasingly important for companies' survival and success, many projects fail to achieve their targets and others are cancelled prior to completion. Basic evidence was also provided about the general awareness that failure is rooted on factors which are not addressed by the traditional approach to project management.

The traditional approach has placed great emphasis on the operational issues of work scheduling and resource allocation. While this is an important operational area of management concern, most failures are of strategic, social and human nature. Managing people, their influences, expectations and general behaviour, is therefore crucial. Managing the whole, by taking effective high-level long-term decisions is essential. The traditional project management process and techniques were not designed to address these managerial needs. Problems with human factors and strategic decisions eventually have an impact on the operational issues. However, in most cases, effective solutions cannot be found at this level. The traditional approach needs to develop processes and techniques to diagnose problems and devise solutions at the higher strategic level. This requires a more holistic and systemic approach to project management, where the whole is considered as much more than the simple sum of the parts, and where interactions among the project elements are the drivers of project behaviour.

Based on an exhaustive review of a number of practical past applications, chapter 2 also provided evidence that System Dynamics modelling gathers the requisites to support this systemic approach, and thus to address the main causes of project failure. Past applications indicate that System Dynamics has the potential to provide distinctive benefits and thereby its own contribution to the project management process. While SD also addresses some issues common to the traditional approach, it appears to be much more valuable and suitable as a complement rather than as a complete alternative to the PERT/CPM based techniques.

However, as also discussed in chapter 2, System Dynamics is not well known within the project management community. And there is even less knowledge of how to apply it in practice. If project management is to benefit from the distinctive contribution of System Dynamics, knowledge needs to be developed and shared, about how to use this technique, integrated within the traditional project management process. As also discussed in chapter 2, integration emerges logically as the most beneficial route. Project management deals with complex problems, where various tools and techniques need to be used together in an articulated manner. Different tools and techniques address different aspects of problems. An effective articulation is crucial so that effective global solutions can be developed. The use of different tools and techniques in an informal manner can provide some benefits, but it does not ensure consistency between the recommendations provided – not only it is difficult to find global solutions in this way, as it may lead to problems (e.g. different tools providing different forecasts and conflicting recommendations). It is equally logical the most effective way of articulating different tools and techniques is through the establishment of formal processes (which generally comprise rules, conditions, and interfaces, to ensure consistency and support the exchange of data and information). It can be objected that formality can also bring lack of flexibility and restraining conditions, which are often difficult to respect in practice. The perspective assumed in this research about formal processes is that they constitute an ideal vision which, if properly designed, can be relaxed to accommodate the imperfections of the real world. The reverse is not true. An informal framework can be the basis for the development of formal processes, but discovery and creation is required. An informal framework is not ready to support the practical implementation of formal processes to integrate different tools and techniques.

On the other hand, it can be objected that, under a pure logical perspective, if the use of System Dynamics in a formally integrated manner has never been implemented before, then one cannot logically assert that this will be beneficial. One premise of this research is that SD has already proven beneficial when applied in an isolated manner from the project management process. As just argued, another premise is that formal integration is most likely to sustain those benefits and provide extra ones, due to possible synergies – past experiences in various fields have been showing that integration of tools, techniques and technology has brought this type of results. But it is still logically fair to wonder about "negative synergies", in which case integrating SD would overall prove dis-beneficial. While this was not expected to be the more likely outcome of this research, nor its core motivation, this possibility was no excluded. Fieldwork was therefore carried out under the spirit of developing a theory to integrate the use of System Dynamics within the project management process, while finding evidence that such theory would provide beneficial results. These results would consist in two main conclusions: used in this way, SD would help to solve problems to which the traditional approach would not provide effective support, and SD would not attenuate or eliminate the benefits of the traditional project management process.

Overall, the formal integration of System Dynamics within the project management process therefore appeared, at this initial stage, as the most effective way of providing a means for project managers to take advantage of its distinctive benefits. However, as also discussed in chapter 2, there had been no attempt so far to integrate the use of System Dynamics within the traditional project management framework, in a formal structured manner. All past applications of SD had been implemented in total isolation from the processes and techniques of the traditional approach. Those who carried out these applications were experts in System Dynamics interested in exploring its potential benefits in the field of project management, rather than practising project managers interested in solving real problems, where they would have perceived System Dynamics to be the appropriate technique to devise solutions. These past applications were fundamental initial steps in the use of System Dynamics, but none of them developed structured knowledge about how to use it as an integral part of the overall project management process.

As discussed in chapter 2, the integration of System Dynamics within the traditional framework can be considered at different levels of formality. Alternative routes can therefore be followed. It was argued that it is first necessary to establish a process framework capable of articulating the use System Dynamics with the traditional tools and techniques, within the existing project management process.

3.2.2 The research question and hypothesis

The present research question therefore focuses on the establishment of an integrated process framework capable of improving project performance: how can this be done? How to apply System Dynamics within the traditional project management process in order to address the strategic issues? What are the more valuable distinctive roles of SD? How to link formally a SD model with the other models produced by the traditional techniques (e.g. WBS, PERT/CPM plan)? What information and data must be exchanged between them, and how? How to accomplish all this in a form that provides added benefits to the project management process?

It is important to note that these questions raise issues of process and technical nature. In terms of process, the use of SD may bring changes to the logical flow of managerial actions considered in the traditional project management framework. Regarding the technical issues, a SD model needs to be linked and exchange data with the models produced by the traditional tools and techniques.

Therefore, two fundamental questions can be posed: what are the process changes, and thus what is the new logical flow for the project management process? How are information and data links established between the SD and the traditional models? This is summarised in the proposed research question:

Summary research question:
How can the use of System Dynamics be formally integrated within the traditional project management approach, becoming part of the project management process, exchanging information and data with the traditional PERT/CPM based models, and thereby provide additional distinctive benefits?

A formal nature for the integration of SD is proposed at this stage on the basis that formality implies rigour, lack of ambiguity and thereby potential for better results. This is particularly important because SD leads to the development quantitative simulation models, which have many inputs and outputs in common with the also quantitative PERT/CPM models. Furthermore, a formal integration can be relaxed into an informal integration, if that becomes required in practice. However, the reverse is not true.

One way to answer a research question is to formulate the correspondent hypothesis, and then test it. This strategy is typical of a positivist approach to management research (Easterby-Smith et al 1991) – as already mentioned, the approach adopted was predominantly phenomenological. An hypothesis aimed at answering this question would be supported by an up-front theory, which would explain how SD could be used. This is inappropriate for the present research, because the theory was actually developed through fieldwork, according to the phenomenological approach – the reasons for this choice will be discussed in the following sections. Nevertheless, it is the author's opinion that posing a research hypothesis in a "positivist way" may bring clarify to the overall purpose of the research. Such an hypothesis would be a "yes" to the research question:

Research hypothesis:

The use of System Dynamics in project management can be formally integrated within the traditional project management process, through the establishment of formal links with the PERT/CPM based models which support the exchange of information and data. The new resultant project management process has potential to provide management with an aid to enhance project performance, through the complementary and distinctive contribution of System Dynamics models.

For obvious reasons to be discussed in the following sections, this hypothesis could not be tested statistically, based on large-scale sampling. As it will be seen, apart from problems of feasibility, conceptually this would not be the right approach as well.

On the other hand, if this hypothesis is to be confirmed throughout fieldwork, then it is essential to establish a clear aim and objectives for such research project.

3.2.2 The research aim and objectives

Aim and objectives are herein differentiated in the sense that an aim is more general and far-reaching, while objectives are more specific and operational. The research aim should provide a basis to specify the work scope, and the objectives should support the development of an appropriate work plan. The aim and objectives also raise the need for an appropriate research approach.

The aim of the research focused on answering the research question. This was achieved through an extensive practical fieldwork. This strategy required that the use of System Dynamics was first conceptualised, and then tested, refined and formalised into a generic formal methodology. The research aim is therefore as follows:

Research aim:
To conceptualise the integrated use of System Dynamics modelling within the traditional project management process, formally linked with the PERT/CPM based models. To test this concept in practice, identify possible distinctive benefits, and, if any, further develop a formal methodology capable of guiding implementation in other projects.

In order to achieve this aim, specific objectives were identified for the research. The conceptualisation stage preceded the testing and development stage during fieldwork. It was decided that this conceptualisation would consider less detail than what would be required to implement the idea in practice. As it will be seen, this highlights an important phenomenological aspect of this research: the theory was to be developed based on understanding gained from the in-depth fieldwork study. It was acknowledged that there were many questions and issues that could not be anticipated at this early conceptualisation stage. The outcome from conceptualisation would therefore consist only of a high-level framework. Developing this up-front conceptual framework was the first main objective set out for this research.

The second objective was to test this framework in practice, using a real project as fieldwork. As results and conclusions would emerge, this framework would need to be readjusted and refined into more detail. The third objective was to explore throughout the field work alternative ways of applying System Dynamics, and identify the more effective ones. Another objective was then to refine the conceptual framework into the necessary detail, so that a formal methodology could be developed – this would include the "technical invention" of formally linking a SD project model with the traditional PERT/CPM models. This methodology would comprise a practical "step-by-step" process, which project managers could use to apply System Dynamics to their projects in an effective way. Although this methodology was developed throughout the fieldwork, the final complete specification would take place after this practical experience. A final objective was to gather evidence that, used in this way, System Dynamics would provide its own distinctive benefits to the project management process. Therefore, the developed methodology would have potential to improve project performance.

The overall objectives of the research were defined as follows:

Research objectives:

- *develop a conceptual high-level framework (to apply System Dynamics within the traditional project management process, in an integrated manner with the PERT/CPM based models);*
- *test this conceptual framework in practice (through implementation in a real project);*
- *throughout this practical testing, explore alternative ways of applying System Dynamics and identify the more effective ones so that overall benefits are achieved;*
- *throughout this practical testing, if benefits are identified, readjust and refine the conceptual framework into more detail (in particular the formal links between the SD model and the PERT/CPM models), so that a formal methodology can be developed. After this practical fieldwork, develop a final complete specification of the methodology;*
- *gather evidence that this methodology has potential to provide distinctive benefits, thereby leading to performance improvements in project management.*

The successful achievement of these proposed research aim and objectives required that an appropriate research methodology was adopted. As already mentioned, this approach was predominantly of phenomenological nature, and it was based on an extensive fieldwork carried out in a real project. The following section provides an overview of some important methodological aspects of management research, and relates these to the research methodology adopted.

3.3 Methodological approaches to management research

3.3.1 Overview

Management research can be implemented in different ways and thus alternative approaches can be followed. It is important to note that there is no "correct" generic approach that can ensure, per se, the validity of the research. Instead, the approach adopted must respond to the various demands and conditions, within which the research is to be conducted. For each specific case, the appropriate approach depends on various factors. Of particular relevance is the nature of the research question (and problem) being addressed. This section provides a brief overview of the methodological aspects considered in the design of the research approach adopted in this work.

3.3.2 The perspective of management research

Easterby-Smith et al (1991) argue that management research is not only about achieving a better understanding of management, but also about producing understanding for managers so they can produce better decisions. The research conducted in this work was aligned with this concept. The integration of System Dynamics within the project management process, as an attempt to address the strategic human causes of project failure, is aimed at both understanding how these factors may be handled effectively, and deploying a practical methodology which project managers can use in the field to improve project performance.

As compared to the research activity in other classical sciences, management research is characterised by a shift of emphasis from counting events on the basis of quantitative methods, to understanding the nature of those events based on qualitative methods (Easterby-Smith et al 1991) – i.e. the focus is on the "why" and not on "what". The present research involved the development and use of quantitative modelling techniques for project management. Furthermore, the methodology developed is intended to be formal. However, the nature of the research itself focused on a qualitative and deep understanding of how System Dynamics could be effectively used to improve project management performance. The human activity of "management" itself can be considered under different perspectives. This will influence the approaches taken to conduct research in this area (Easterby-Smith et al 1991). In the present research, management as a human activity was considered in the context of planning and controlling a project towards certain objectives. This implies a structured process of human actions and decisions, according to the classical view of management: planning, organising, co-ordinating and controlling. It was also considered that project management is centred on a decision-making process, a view aligned with the perspective of the decision-theory, about management research (Easterby-Smith et al 1991). Project management is not a theoretical formal science. On the other hand, it is a practical science primarily concerned with helping project managers to solve real problems. These problems take place within time-pressured environments, where social human issues and negotiating skills are permanently at work. In this way, project management was also considered in this work as a social construct.

There are various aspects that make management research particularly challenging. Easterby-Smith et al (1991) identify some important ones: (1) management is based on multiple disciplines, (2) required fieldwork is restrained by managers' personal interests, and (3) resultant actions and practical consequences of the research are vital. These authors further argue that a cross-disciplinary approach to management research is more likely to produce practical results. This was also the perspective herein adopted. The fieldwork was carried out under various constraints, which were accounted for in the conclusions drawn from this experience. The present research also involved action and practical consequences, in the form of intervention during this fieldwork. The methodology developed is also meant to be of practical consequences in the future.

3.3.3 Forms of management research

There are three main possible forms of management research: pure, applied and "action research" (Easterby-Smith et al 1991). Pure research is characterised by theoretical developments, which may or not have practical application. This can consist in the discovery of a new idea about reality, the invention of a new technique, or in the reflection about the application of an existing theory in a different novel scenario. Applied research is intended to lead to a solution for a specific problem. It must focus on the causes of the practical results achieved, and it requires that valid evidence is provided. Action research focuses on continuous change and in-depth understanding through fieldwork: the research leads to changes in the field, and these changes in turn affect the research. This form of management research requires a collaborative approach, involving the researcher and the researched closely. It often takes the form of a "one-off" event that cannot be repeated in the same way, but which is appropriate to test a certain "technology" (Eden and Huxham 1996).

As it will be discussed later in chapter 5, the form of research adopted in this work was dominated mainly by action research. It involved an extensive fieldwork, where an initial concept was progressively refined through the development of deeper understanding. However, the initial stage of conceptualisation took the form of pure research, through creative and logical invention. As argued by Easterby-Smith et al (1991), chances of a doctoral research work are improved if both pure and applied elements are incorporated into the work. These authors also argue that starting from the analysis of others work, instead of using extensive data collection, can be in some cases a good strategy. To an extent, this was implemented through the analysis of the past applications of System Dynamics to project management, as discussed in chapter 2.

It is also important to note that action research may have some drawbacks. For example, difficulties in finding an end-point to the fieldwork, and in finding the generic relevance of the work when writing it up (Easterby-Smith et al 1991). The former difficulty was experimented in part, but the overall extent of the fieldwork was worth and most valuable for the purpose of the research. The latter difficulty was not encountered, probably because the starting point was a well defined question, which mixed a pure and applied research perspectives.

3.3.4 Research design: positivism vs. phenomenology

Easterby-Smith et al (1991) suggest is a good strategy to design a research project at the light of the two extreme philosophies of management research: positivism and phenomenology (also referred to as "social constructionism"). As already mentioned, the research here presented was primarily designed under a phenomenological perspective. This was largely imposed by the social nature of the research question. This was an interesting experience, since the background of the researcher was dominated by the positivist perspective.

Table 3.1 below provides an overview and compares the key aspects of positivism and phenomenology (Easterby-Smith et al 1991). The fundamental difference (1) is that positivism considers reality as external and independent from human desires or interpretations. This is the perspective underlying the classical approach to scientific knowledge. Research therefore is about discovering elements and understanding the "inevitable" reality, and about developing techniques which will always produce an expected result if applied under same conditions. Phenomenology on the other hand, focuses on the human dimension of reality. It tries to understand in-depth the human perception of reality, which depends on meanings, emotions and opinions. Reality is therefore socially constructed and is primarily subjective.

Stemming from this fundamental difference, other important aspects differ as shown in table 3.1. The positivist approach is particularly appropriate to study "physical" systems, which are dominated by the inevitable fundamental laws of nature. However, while this approach has it roots in these classical sciences, it has been applied to social systems as well. Phenomenology is rooted on a human perspective of reality and emerged in the study of social systems, where the dominant element of reality is human behaviour.

In the social sciences, positivism is implemented by starting with an objective question under the form of an hypothesis. This hypothesis is then tested based on objective quantitative analysis. The question is objective in the sense that an answer can be provided by means of measurements and data analysis.

Quantitative methods are employed over large samples taken from reality, in order to prove or disprove the hypothesis. For example, if a researcher wanted to know whether the use of a certain planning technique improves project performance, then a large sample of past projects would be considered, wherein some projects would have used the technique. The performance of the all projects would be known and objectively measured (e.g. percent deviation from targets). Statistical tests would be implemented to find out whether the use of the planning technique had a relevant impact on this measured performance. In practice, a positivist approach tends to take the form of a sequence made of hypothesis followed experimental tests. Hypothesis are formulated a priori as potential explanatory theories. This approach is also based on the principle that the truth about reality can be split into simple elements, and thus progressively discovered through elementary truths, which added-up from the final truth.

Key aspects		**Positivism**	**Phenomenology**
Basic beliefs	(1)	The world is external and objective	The world is socially constructed and subjective
	(2)	Observer is independent	Observer is part of what is observed
	(3)	Science is value-free	Science is drive by human interests
Researcher should	(4)	Focus on facts	Focus on meanings
	(5)	Look for causality and fundamental laws	Try to understand what is happening
	(6)	Reduce phenomenon to simplest elements	Look at the totality of each situation
	(7)	Formulate hypothesis and then test them	Develop ideas through induction from data
Preferred methods	(8)	Operationalising concepts so that they can be measured	Using multiple methods to establish different views of a phenomena
	(9)	Taking large samples	Small samples investigated in-depth and over-time

Table 3.1 – Comparison of key aspects of positivism and phenomenology

In contrast, phenomenology does not tend to start with a well defined hypothetical up-front theory. Instead, it tends to start with a general concept, idea, phenomenon or problem, which is not well understood. Developing understanding about this issue is perceived as useful in light of human interests. The research then focuses on studying the problem through an in-depth involvement in a few situations where it occurs. Throughout this fieldwork experience, the researcher will develop the required understanding to produce an explanatory theory, which can be useful to handle similar problems in the future. Ideally, this developed theory is aimed at being "generalisable". However, it is important to note the few in-depth experiences cannot prove this in a purely objective fashion. A typical example of this approach is the development of "best practices" within certain industries, like software development. The researcher studies in-depth one or a few successful companies, in order to try understanding what are their key practices that lead to success. From this focused experience, the researcher develops recommendations for other companies to follow. In phenomenology, data can also be used but the focus is not on proving or disproving an hypothesis. Instead, data is used to help developing a deep understanding about why things happen in a certain way. Phenomenology is also based on the idea that understanding about the problem requires a view of the whole, and this cannot be split into elementary sub-understandings (an intuitive analogy is to consider that the colour of an object cannot be derived by studying the colour of the elementary molecules which form that object; colour is a human perception about the whole).

There are various advantages and disadvantages of positivism and phenomenology. It is appropriate to consider these as a basis for a choice. Easterby-Smith et al (1991) summarise their strengths and weaknesses as shown in table 3.2. In addition to these strengths and weaknesses, there can be fundamental restrictions and conditions which may impose a choice. The most important ones considered in the present research were as follows:

(1) *availability of large samples* – in order to apply the quantitative statistical methods of positivism, large samples are required. In the present wok this would consist of a large sample of past projects. In some SD had been used in certain ways, and in others where it had not. As discussed in chapter 2, there were simply no applications of SD integrated within the traditional approach. Without a sample, this type of quantitative analysis could not be implemented;

(2) *feasibility of an up-front theory* – positivism requires that an up-front theory is proposed as an hypothesis. The appropriate use of SD in an integrated

manner within the traditional approach, requires that social human issues are addressed. SD is a decision-support tool, which needs the commitment from people and affects people. Since the use of SD in this way had never been done before, there was a poor understanding about how to handle these social human issues. Therefore, an meaningful theory could not be developed up-front;

(3) *access to fieldwork* – the implementation of one or more fieldwork experiments requires access to real projects and openness by companies implementing them. This is often difficult to achieve;

(4) *time required by fieldwork* – the number of field work experiments is restrained by the time required for each case. In phenomenology, fieldwork should be carried out in-depth and over a long period of time. Projects can take several years to complete. Within the scope of this research, it would not be feasible to study in-depth many projects;

findings may depend on human judgement – when the "validity" of the theory being developed depends on human judgement (e.g. has a certain technique provided useful information for decision-making), this same "validity" can hardly be proved or disproved based on data. As mentioned in table 3.2, the conclusions could be artificial. Qualitative methods of phenomenology are more appropriate.

	Positivism	**Phenomenology**
Strengths	Wide coverage of situations Fast and economical Aggregate analysis can be of relevance for policy decisions	Look at change processes over-time Addresses people's meanings Flexible in adjusting to new ideas as they emerge Strong at developing theories
Weaknesses	The analysis can be inflexible and artificial Do not provide understanding of processes and meanings Not oriented to generate theories, but to test them Do not support inference of changes to be applicable in the future	Data collection can be time and effort consuming May be hard to interpret data Hard to control progress and establish end-points Policy makers may give low credibility to conclusions and their generalisation

Table 3.2 – Some strengths and weaknesses of positivism and phenomenology

Considering the weaknesses and strengths of table 3.2, together with this list of practical restrictions, it soon became clear that this research would follow mainly a phenomenological approach. The aim of the research would take advantage of the strengths of this approach, whereas the strengths of positivism were not particularly valuable. The weaknesses of both approaches were relevant, but the ones of positivism were far more restraining. The practical restrictions above suggested clearly that a predominantly positivist approach would be unfeasible.

As suggested by Easterby-Smith et al (1991), rarely a research work strictly adheres to one of these extreme philosophies and it is beneficial to mix elements of both. The following section summarises the research approach adopted, and presents the main reasons for this decision.

3.4 The methodological research approach adopted

3.4.1 Overview

Once the distinctive benefits of integrating System Dynamics within the traditional project management process were identified, the overall purpose of the research consisted in the development of a formal integrated methodology. It was intended that this methodology could be used by project managers in the future to improve project performance. The methodology would have to be practical, and thus address the managerial needs and conditions within which project managers carry out their wok.

In generic research terms, this methodology represented a theory to be developed. The research was intended to invent a new practical technique. This invention comprised aspects of social and technical nature. The social aspects referred to the way in which SD was to be applied and articulated with other elements within the traditional project management process. These aspects focused on organisational processes and how they would change.

The technical aspects referred to the formal integration of a SD project model with a PERT/CPM model. This included establishing a formal correspondence between the two types of models, and the exchange of data between them. However, this was not a purely technical issue. Formally linking the two models requires assumptions about meanings in the models while representing reality, and how they are used (e.g. data handling). Both types of models deliver a particular representation of reality, based on particular human perceptions. Most likely, different project managers produce different models. Therefore, linking the two models required that these human perceptions were taken into account – i.e. a valid integration required human judgement. As it will be described later in this research (chapter 7), assumptions were made and generic options were considered, so that the methodology itself would become generic. This way, it was possible to address this issue as an almost purely technical problem.

The predominant social aspects lead to the predominant use of phenomenological elements of management research. The technical aspects were handled by positivist elements.

There was also an initial stage of the research, where positivist elements were used. This consisted in identifying the research aim. The researcher looked for evidence of gaps in the project management discipline, and further studied a wide range of past applications of SD to project management. This led to the conclusion that SD could be a valuable addition to the project management discipline (Easterby-Smith et al (1991) refer to this approach, of initially studying others' work, as "armchair theorising"). During this stage, the researcher was not involved in judging the causes for project failure nor in the implementation of the past applications of SD. An initial conceptual framework was then developed to integrate the use of SD within the project management process. This framework was derived logically and thus a high-level up-front theory was proposed. These were all elements of positivism, motivated by the circumstances within which the research was being conducted at that stage (i.e. a purely academic environment). However, even at this initial stage phenomenological elements also emerged: the research topic was motivated by a personal interest in addressing a problem of social concern; and while the conceptual framework was being developed, it soon became evident that there was a lack of understanding about how to best integrate SD and therefore an in-depth fieldwork was required to develop a more comprehensive theory.

3.4.2 The research approach adopted

Overview

The research approach adopted mixed elements from positivism and phenomenology, but it was essentially phenomenological in the form of action research. An extensive fieldwork was carried out, which lasted for about 18 months of intensive involvement in a large software intensive project.

Easterby-Smith et al (1991) consider five main choices that need to be made, in order for the researcher to select the appropriate elements from positivism and phenomenology:

(1) the researcher is independent or involved;
(2) large samples or small samples;
(3) testing theories or generating theories
(4) experimental design or fieldwork
(5) verification or falsification

The first options are used in the positivist approach, and the alternative ones in the phenomenological (except for point 5 where both belong to positivism).

Table 3.3 summarises the research approach adopted and the elements used. Three main research areas are considered: identification of research aim, social aspects of the methodology to be developed, and its technical aspects.

Research areas	Philosophy adopted and elements used
Identification of useful research aim: • does it address an unsolved problem? • is it likely to be feasible?	***Positivist*** • Researcher was independent, in generating initial hypothesis (SD can be useful and should be integrated) • An exhaustive survey of past SD applications was carried out (large samples) • Development of initial conceptual integrated framework, with the desire to test it (testing theories) ***Phenomenology*** • Research topic driven by human interests (i.e. find ways to reduce project failure) • Awareness of lack of understanding and the desire to develop a more comprehensive theory based on fieldwork
Development of methodology. Social aspects: • devise a process to apply SD (including how to develop a model and how to use it) • assess usefulness of applying SD in alternative ways	***Phenomenology*** • Researcher was closely involved in the real world situation studied (became part of a project planning team) • One project was studied, and a small set of scenarios was considered to apply SD (i.e. small samples) • The methodology was developed based on, and throughout the fieldwork (generating theories) • One in-depth fieldwork was carried out
Development of methodology. Technical aspects: • formal integration of SD model with PERT/CPM models	***Positivism*** • A reasonable sample of scenarios was considered • Theories on how to link the models and exchange data were hypothesised *a priori*, and then tested • Experiments were designed to test these theories • Theories were tested mainly by attempts at falsification (i.e. trying to identify scenarios where they would not work)

Table 3.3 – Research approach adopted and elements used

Most of the reasons underlying these choices were progressively introduced throughout this chapter. However, it is important to clarify why this approach was appropriate to ensure that the research aim and objectives would be achieved.

Appropriateness

The aim and objectives of the research implied three main requirements:

(1) prove that it is possible to apply System Dynamics modelling in an integrated manner within the traditional project management process, by formally linking a SD model with the PERT/CPM models;

(2) develop a generic formal methodology to apply System Dynamics in practice in this way, based on an in-depth understanding about the managerial needs;
(3) find evidence that this integrated methodology provides distinctive benefits and improves project management performance, above the traditional approach.

In order to meet these requirements, the research approach proposed proved adequate because:

(a) one single project was able to generate the necessary number of scenarios to verify whether there would be ways to integrate SD, as in requirement (1) (note that there was no intention to prove that SD can always be integrated in this way, regardless of the project management process in place and way in which a PERT/CPM model is used);
(b) in order to be practical and useful, the methodology would have to be tailored to the human and social elements of projects – i.e. it must be "accepted" by the project team (requirement 2 above). An in-depth extensive fieldwork would be necessary to develop the required understanding about these complex social issues, and thereby "validate" the practicality of the methodology. Fieldwork through action research is most appropriate to address this need;
(c) for the methodology to provide benefits and improve project performance, individuals in the organisation must react to its implementation in a productive manner, and they must themselves perceive these benefits (requirement 3 above). The usefulness of the methodology is therefore a social construct, and not an independent truth. It was essential to gather qualitative evidence through a phenomenological approach.

The action research approach adopted ensured that the required fieldwork was carried out, as the means to gain the required understanding so that the theory (i.e. the integrated methodology) could be developed and validated. It was fundamental to understand why SD would work well in certain ways, and not in others. The close researcher involvement in the fieldwork was crucial, given the importance of human behaviour. For example, for interpreting correctly the codes of communication within the organisation, which can be crucial (Easterby-Smith et al 1991).

The use of elements of positivism in the initial stage of the research was appropriate because it was important to find evidence about the usefulness of SD, independently from the researcher's points of view or involvement in actual applications. The initial conceptual framework logically derived was also important as an input to the fieldwork. The further use of positivist elements in the technical aspects of the methodology was also appropriate, given the analytical nature of the "invention" (provided some assumptions were made regarding the meaning and use of the models). Throughout the remaining of the research, the elements used were phenomenological, for the three main reasons stated above.

As already mentioned, the research approach adopted was also designed based on the strengths and weaknesses of positivism and phenomenology. It was important to build upon the strengths and prevent weaknesses (see table 3.2). Practical restrictions were also considered. This can be summarised as follows:

(1) *strengths of phenomenology explored:*

- the aim was to develop a theory about which there was poor up-front understanding;
- the "validity" of the theory was primarily based on people's meanings;
- theory was about changing processes;

(2) *weaknesses of positivism prevented:*

- inappropriateness in supporting the development of theories;
- inflexibility in adapting to emerging ideas and changes;
- not focused on understanding the "why";
- not appropriate to infer future practices;

(3) *restrictions considered*

- unavailability of (large) samples of integrated applications of SD;
- not feasible to develop a comprehensive up-front theory;
- fieldwork required a considerable amount of time;
- ultimately, the research findings were dependent on human judgement.

Potential limitations

The main potential limitation of the approach adopted was the significance of a single fieldwork experience, regarding the generalisation of the theory. This would be crucial to propose a "generic" methodology.

In the first place, it is impossible to objectively demonstrate that a certain invention will produce the expected results in all possible scenarios. However, testing the invention in more than one setting at least provides some confidence about a certain degree of generalisation. In this particular case, it would be relevant to check whether the methodology would work in other projects. As it will be described in chapter 5, the fieldwork took place within a large-scale software intensive project. This project included various increments, each with its own development life-cycles and development teams. In this way, the long time-frame and dimension of the project provided a wide variety of different scenarios wherein SD was applied – some of the more relevant applications are described in detail in chapter 8.

Furthermore, the organisation (a defence company) followed a well structured project management process, very much aligned with the traditional approach. Therefore, a very specific and unique project management environment was not encountered, at least within the software industry. On the other hand, it can be found in many other organisations that implement the standard project management process in this industry. To some extent, the overall fieldwork experience can be compared with a set of separate experiences carried out in different projects (where the traditional project management framework was being adopted). This fact supports the generalisation of the methodology developed in this research.

Nevertheless, it is important to note that the fieldwork, having been carried out within a specific industry (as it had to be), may impose some constraints regarding generalisation. As it will be described in chapter 5, the project wherein the fieldwork took place was aimed at developing a complex combat system, which included software, hardware and other physical components (e.g. radar sub-systems, weapons). It was a typical "military system development project", well within the domain of this research (i.e. "design and realization" projects; see chapter 2). However, the work developed in-house was primarily software development, and it was within this environment that the integrated methodology proposed in this research was developed. Although some of the practical applications were not specific to the software development process (some of which will be described in chapter 8), it is fair to wonder whether the methodology is applicable to other industries, or whether a different methodology would have been developed if the fieldwork had taken place in such other industries. Of particular relevance are the more "discrete" type of projects, like engineering, naval and civil construction. The product development process in these industries has some important differences from the software field. For example, iteration is limited and often prohibitive, the product is more tangible, and the requirements, in general, are more stable. As it will be discussed in chapter 9, it is the author's view that further fieldwork in this type of industries would be most valuable to verify the applicability of the methodology proposed – it would not be feasible, however, to include such experiment within the scope of this initial work. It should also be noted that during the fieldwork carried out, the author has focused on the conceptualisation and testing of the methodology on generic project management issues, most of which are common to all "design and realization industries". In particular, the process logic and the analytical links between the SD and the PERT/CPM model, are not specific to the software industry. In addition, as discussed in chapter 2, some of the most successful past applications of SD to project management have taken place in other engineering type of industries. Interestingly, they do not differ much from the ones carried out in the software field. For example, it would be fair to expect that the structure of the SD model would probably be considerably different. However, some authors of these past applications even propose cross-industry generic SD models. The author has also been involved in applying SD to large-scale engineering projects (e.g. submarine development), and again many similarities to the SD application in the software field were encountered. Overall, it is the author's view that while further fieldwork in such type of projects is valuable, the extrapolation of the methodology here proposed to those fields will not require a major re-conceptualisation.

Validity

The overall validity of the research approach should be questioned in different ways, depending on its positivist or phenomenological nature (Easterby-Smith et al 1991). Regarding positivism, it is important to address the validity of the measurement instruments used, the independence of the measures achieved, and the independence of the results from the sample measured. In this research, it was clear that the conclusion of the initial review of past applications of SD would be the same regardless of the researcher. Regarding the technical issues of the methodology, control experiments were used to test the algorithms developed.

Regarding the phenomenological elements, it is important to address whether the researcher has gained full access to the knowledge and meanings of informants, whether the observations made are independent from the researcher, and whether the findings or theory developed can be generalised. Regarding access to knowledge and meanings, the researcher was involved in the project for the period of 18 months, during which co-lived and interacted with the members of the project team. Furthermore, he organisation did not raise obstacles regarding access to individuals and data (except that is should not cause any major disruption to the project). Informal interviews, working closely with particular individuals, and social interaction, all ensured that the views and opinions being passed onto the researcher were genuine, and in some cases could be "filtered".

3.4.3 Other possible approaches

The research approach adopted resulted in great part from the availability of fieldwork. BAeSEMA Ltd (UK) (now part of British Aerospace Systems) was interested in the use of System Dynamics, and there was a major project just starting. The researcher was given the opportunity to become involved in this project, which resulted in about 15 moths working on-site (this will be described in detail in chapter 5). Action research was perceived up-front as the more appropriate route to take, and became feasible.

In order to appreciate the options taken in the research approach adopted, it is probably interesting to speculate what could have been the alternative approaches, in case this fieldwork had not been available. Once the literature review was complete, and the novel idea of integrating the use of System Dynamics within the traditional project management framework was developed, what to do next with no fieldwork?

Since the underlying motivation for the research was perceived valid and useful, the main alternative route would have been the invention of a new technique through pure research. The technique itself could have been one of the two main elements of the methodology developed: the analytical links, or the underlying process.

In the first case, the theoretical invention would have focused on the analytical links established between a SD project model and a PERT/CPM model. These links could be fully automated requiring either no human input or very specific quantifiable input parameters. The technique could have been implemented, for example, through the development of a software tool, which would have the capability of transferring a project representation back and forth between the two models – if this had been "ordered" by a "client", this work would take the form of applied research. The validity of this invention would be based on tests using of fictitious project scenarios. The practical application of this invention to solve problems in the real world would not be a direct concern of the research – many of the PERT/CPM enhancements available in literature were developed in this "pure research" way . The research approach would have been predominantly positivist, where a theory had been developed and tested in an artificial academic environment.

In the second route, the technique to be invented would have been a process methodology on how to use a System Dynamics model in an integrated manner within the traditional project management process. This would take the form of a step-by-step logical process. It would be developed up-front based on logic, literature, and the researcher's own experience. In order to validate this technique (which would be an up-front theory), and because fieldwork was not available, it would be important to know the opinion of practitioners. This could be done through questionnaires, surveys and interviews. These would have to be in a large number. Again, in this scenario the research approach would also be mainly positivist, but this time perhaps with more phenomenological elements (e.g. the subjective opinion of practitioners).

The actual action research approach followed focused on the development of these two techniques in a single research project – i.e. the analytical links and the process methodology. However, there was a fundamental difference from the alternative route specified above. This difference resulted from the nature of phenomenological action research: the methodology developed was "grounded theory" (Easterby-Smith et al 1991). This is, it was developed through fieldwork and based on understanding gained from observing and living actual problems in the field. The main impacts were a more holistic and flexible methodology. It was less focused on the automated implementation of the technical issues; its scope became wider and focused on the whole project management process; its "rules" became flexible, leaving alternative paths opened to cope with human and social issues. It is the author's opinion that, overall, a more realistic and practical output was produced, reflecting a large amount of testing and feedback received form the real world. Again, this was a clear benefit from phenomenological action research.

3.4.4 The research methods employed

Overview

To implement a research approach, there are various methods of analysis that the researcher can use. These fall into two main categories: quantitative and qualitative. Quantitative methods focus on the collection and analysis of data, and are dominated by statistical techniques. Qualitative methods focus on structured ways of validly extracting useful information from individuals (Easterby-Smith et al 1991). An example of a statistical technique is regression analysis. An example of a qualitative technique are questionnaires, or the Delphi and nominal group techniques (Wright 1985, Kerzner 1998).

The phenomenological nature of the research approach adopted called for the use of qualitative techniques. However, as already mentioned, there were also aspects of the research where positivist elements were employed, and this called for the use of some quantitative methods.

The methods employed in this work are presented in table 3.4 (see appendix B). They are related to the main research needs encountered throughout the work. As expected, most of the methods are of qualitative nature.

Qualitative techniques

The validation of the process underlying the methodology was aimed at ensuring its practicality. For example, in order to be successfully implemented in the real world, it is important that a project team does not feel overwhelmed and restrained by an inflexible and effort-consuming process. Group presentations to the project team, followed by debate, were used for this purpose. As the process was being progressively designed, it was essential to expose it to criticism to the whole project team. Various questions were debated: would certain data be available when required? Would the PERT/CPM project plan be available when required? How regularly could certain information and data be updated? Would certain types of analysis be of any help at certain stages? The perspective of the project team was simultaneously one of user and "client" of the methodology. They would have to support its implementation, and would expect the benefits to outweigh the effort required. It was therefore essential that the process underlying the methodology was practical, dynamic, flexible and at the same time unambiguous, structured and rigorous. These group presentations were carried out regularly, one about every two months in average.

Informal individual interviews was the technique used more often. It was used to validate the methodology, the SD models developed, and to assess benefits. Being individual and informal interviews, it may appear as having nothing of structured to make it a research method. However, there were very good reasons to implement the interview in this way, and care was taken on how to conduct them. The project was an important commercial endeavour, and therefore management and staff were under considerable pressure to achieve high levels of productivity. The staff was mixed, including temporary staff, newly recruited staff, and experienced staff from the company. Because SD project models tend to adopt a pragmatic perspective about reality (e.g. that the perceived progress is generally above the actual progress), the interviews were often focused on issues related with productivity levels, defect rates, scope changes, QA efficiency, and others alike. These are obviously sensitive issues. It was therefore important that interviews were individual in order to prevent influence from others (which could stem from hierarchical issues and lack of familiarity). It was also important to carry out these interviews informally to prevent fears that recorded statements would be used for unknown purposes or exposed to other individuals. Initial attempts at using formal questionnaires were not successful, whereas informal interviews were found to be very effective. This was supported by the close involvement of the researcher within the project team. While the interviews were informal, some important issues were addressed carefully: clear questions, personal statements were prevented, and "leading" questions were avoided.

Informal questionnaires were used to identify the relevant factors to include in the SD model. Interestingly, this was the type of information that management and staff would feel free to provide in a "recorded fashion": "tell me about what you think is important that should be addressed by the model?". Individuals are generally keen on talking about how things really work in the field and what is relevant. Nevertheless, the questionnaires were informal, as they were not used for further statistical analysis (e.g. counting the no. of persons that answered in a certain way).

Observation of management and staff behaviour was a crucial method. It was essential "to read" the behaviour changes of management and staff, and relate them to the research underway. System Dynamics was something new, totally unknown to most of the individuals. There was support to this research project, but there were also concerns about its success and about the conclusions from the SD analyses (e.g. political issues). For example, most individuals would feel intimidated to suggest that the process being followed was no adequate to apply SD. Furthermore, managers could take into account the conclusions from the SD analyses, but in the early stages they would feel reluctant to accept that they were changing their decisions because of this (i.e. they were supposed to know what was best for the project anyway!). The more relevant behavioural aspects observed were changes regarding explicit support to the research project, and regarding actual management decisions.

Demo sessions with the SD model were also carried out. The validation of the SD model involved running simulations regarding different project scenarios. Was the model reproducing the expected behaviour, and for the right reasons? In order to answer this question, the qualitative opinion of managers about the behaviour and explanations produced by the model has proven valuable.

Finally, influence diagrams were used in a "cognitive mapping fashion" to support the validation of the structure of the SD model. Are all the relevant factors being considered? One way to answer this question is whether all relevant feedback dynamics are being captured. High-level influence diagrams, resembling cognitive maps, were used in some occasions to support this analysis (e.g. as reported in Rodrigues and Williams 1998). As far as these diagrams were kept simple and tidy, this experience has shown that they were an effective means to identify factors which were missing in the model structure.

Quantitative methods

Some quantitative methods were also used to support the validity of the research work underway. They were use for two main purposes: validating the calibration of the SD models and the technical elements of the methodology, more specifically the analytical links established between the two models.

The validation of the SD model was important, because otherwise the analyses produced would not provide useful recommendations. The calibration was validated based on the typical confidence tests used in System Dynamics modelling (Forrester and Senge 1980; Barlas 1996). Because of its importance, this issue became a subject of the research itself, and a general validation framework was developed as part of the proposed methodology – this is described in detail in chapter 6. The quantitative method used as part of these confidence tests was the statistical analysis of "goodness-of-fit" (Sterman 1984). Basically, this consists in producing statistics which indicate how closely the SD model is reproducing a certain project behaviour. Examples of the application of this method are presented in chapter 8.

The validation of the model calibration was also based on extensive data collection and analysis. This took the form of a metrics program, where collected raw data is processed to generate useful parameters about the project. For example, the staff productivity measured in lines of code per person-day, and the average effort required to detect errors measured in person-day per error. Some of these parameters were inputs to the SD model, while others represented final results that SD model would have to reproduce. Data was collected in two different ways: by inspecting the database of the project management information system (which included progress reports and versions of the project plan), and through informal interviews.

The validation of the analytical links between the two models consisted in ensuring that, once the data was transferred from one model to the other using these links, the two models would represent the same project reality. The statistical analysis of "goodness-of-fit" was used for this purpose. The project behaviour produce by the SD model, and the behaviour implied in the PERT/CPM model were compared by this analysis. This lead to the concept of "data-consistency links" , which is explained in detail in chapter 7. This was not the only way used to validate the links, but it was an important quantitative method used.

Dissemination

There was another very important "method" used to "validate" the research, which is not identified in table 3.4. This was the regular dissemination to expert audiences of the developments underway, and results achieved. This took the form of presentations at conferences, and publications in refereed scientific journals (e.g. Rodrigues and Bowers 1996a, 1996b; Rodrigues and Williams 1997, Rodrigues and Williams 1998, Rodrigues 1999). In this way, the research work was exposed to the scrutiny and critique of experts in the fields of System Dynamics and Project Management. Among others, the conferences included the annual International System Dynamics Conferences and the UK Operational Research Conferences. The scientific journals included the System Dynamics Review, the Journal of the Operational Research Society and the International Journal of Project Management, also among others. Very useful feedback was received by both the referees of these journals and the audiences in the conferences. The work was readjusted and improved accordingly.

3.5 Initial work plan

Once the key decisions had been made regarding the research approach, an initial high-level work plan was developed. This plan specified clear and objective targets for the research, and a general route to be followed.

The initial stage of the research had been the gathering of evidence about the potential distinctive usefulness of applying System Dynamics in an integrated manner within the traditional approach to project management. The following stages planned for the research were:

(1) develop a conceptual framework, specifying:
   - (i) a high-level structured process, describing how a SD model was to be used and its distinctive roles;
   - (ii) the basic principles of formal integration of a SD project model with a PERT/CPM model;

(2) carry out the fieldwork:
   - (i) using the conceptual framework as starting point to apply System Dynamics;
   - (ii) progressively refining and readjusting the framework as appropriate, based on feedback from tentative applications;
   - (iii) developing new concepts and ideas to be incorporated in the methodology (in particular the analytical links);

(3) compile the results and conclusions from the fieldwork. If the use of SD would prove successful:
  (i) gather the evidence about the benefits;
  (ii) produce a final version of the formal integrated methodology.

The research was implemented according to this plan. The next chapter 4 presents the conceptual framework developed prior to the implementation of the fieldwork. This framework includes a specification of the conceptual links between a SD project model and a PERT/CPM model, and the identification of the potential roles of the SD model within the integrated process. Chapter 5 provides a description of the action research fieldwork carried out (herein referred to as the "KDCOM case-study"), and presents the main conclusions. Chapters 6 and 7 provide a detailed description of the integrated methodology developed, which is referred to as SYDPIM (System Dynamics-based Project-management Integrated Methodology). Chapter 6 presents the model development method, and chapter 7 presents the underlying process and the analytical links of the methodology. Chapter 8 summarises the main evidence gathered regarding the benefits and successful implementation of the methodology.

4. A conceptual integrated framework

4.1 Structure of this chapter

In the previous chapter, some of the more relevant past applications of System Dynamics to project management have been reviewed. Some of these developments have been undertaken at recognised organisations like NASA (Abdel-Hamid and Madnick 1991, Lin and Levy 1989), M.I.T (Ford 1994) and PA Consulting (Cooper 1993). At the same time, research in this area has been progressing at several academic institutions. It was concluded that System Dynamics offers a valuable complementary contribution to the traditional approach, and therefore the previous chapter suggested that the most promising way ahead is towards the development of a single integrated project management methodology. The first step in this direction is to develop an initial conceptual framework, outlining the general principles of such methodology.

In this chapter, the implications of the past applications of SD to project management for an integrated approach are discussed. In particular, it is relevant to investigate to what extent these applications have been implemented within the context of the traditional project management approach. The general scope, requirements, and objectives of a conceptual integrated framework are identified, and a rationale is further developed. The final framework is then proposed and described in more detail. The directions for future work are discussed.

4.2 Current scenario

While the past applications of SD discussed in the previous chapter provide good evidence about the promising contribution of System Dynamics, this alternative approach is not well known within the project management community. Some of the major real life applications have been undertaken on a consulting basis, in which the SD analysis is fully implemented by the service provider (Cooper 1980, Pugh-Roberts Associates 1993, Williams et al 1995). Most of these cases relate to the post mortem analysis of very specific problems within a particular project, typically within the context of dispute resolution. On the other hand, while there has been an increasing use of SD models as tools to support strategic analysis and learning within management environments (Morecroft 1994), few of these cases seem to apply to real project management environments. The little reference to the System Dynamics approach in the project management literature suggests a poor awareness of its distinctive roles and benefits. This scenario poses a major obstacle for System Dynamics to become of widespread use within project organisations.

Regarding the past applications of SD within real project environments, while some authors claim a better performance of SD models against the traditional tools in the common areas of application (e.g. cost and schedule forecasting) (Cooper 1993), other authors prefer to suggest that SD models should be seen as a "different type of tool to serve a different purpose" (Lin 1993, Williams et al 1995). On the other hand, as discussed in chapter 2, the improvement of traditional PERT/CPM network based models has been moving towards an increasing level of complexity and detail. Such efforts have not been directed towards incorporating the soft strategic factors and feedback processes, which are the focus of the System Dynamics approach. The practicability of advanced PERT/CPM based models has benefited from the development of user-friendly software. However, their static perspective focused on the detailed logic of the project work, still fails to deliver the dynamic feedback perspective offered by System Dynamics. In the real world, projects seldom follow the simple linear route suggested by the logic of the project network.

Although the importance of a combined use of SD with the traditional techniques has been suggested elsewhere (Rodrigues 1994, Williams et al 1995), no specific ways of linking them have been proposed so far. Within this scenario, the use of SD models is restricted to an extra but separate tool, capable of providing valuable support to strategic issues. Well calibrated models can be used to generate quantitative estimates (Cooper 1996). But in this type of application, no explicit relationship with the estimates produced by the traditional models is considered. There is no process available for a combined application of both types of models. In this scenario, the use of SD models cannot move beyond the role of providing *informal* strategic guidance to the operational management level. As an example, a SD analysis could recommend staff not to be hired in the later stages of a project. However, such a recommendation would not specify how this policy would relate to the individual budgets of the specific tasks in the WBS. In estimating, a SD model is therefore restricted to produce quantitative input to the project management process (e.g. overall project duration and cost), where it can be used either as an alternative to the estimates produced by traditional models or, at best, as some kind of input to the traditional models but where a direct "translation" is not available. As an example, a SD analysis could estimate an overall schedule slippage for the project, but without being able to specify the project tasks in the PERT/CPM network where the critical delays would occur.

According to this current scenario, the use of System Dynamics within the traditional project management framework is characterised by a "gap" between the SD analysis at the strategic level, and the use of the traditional network models at the operational level. This is illustrated in figure 4.1 below.

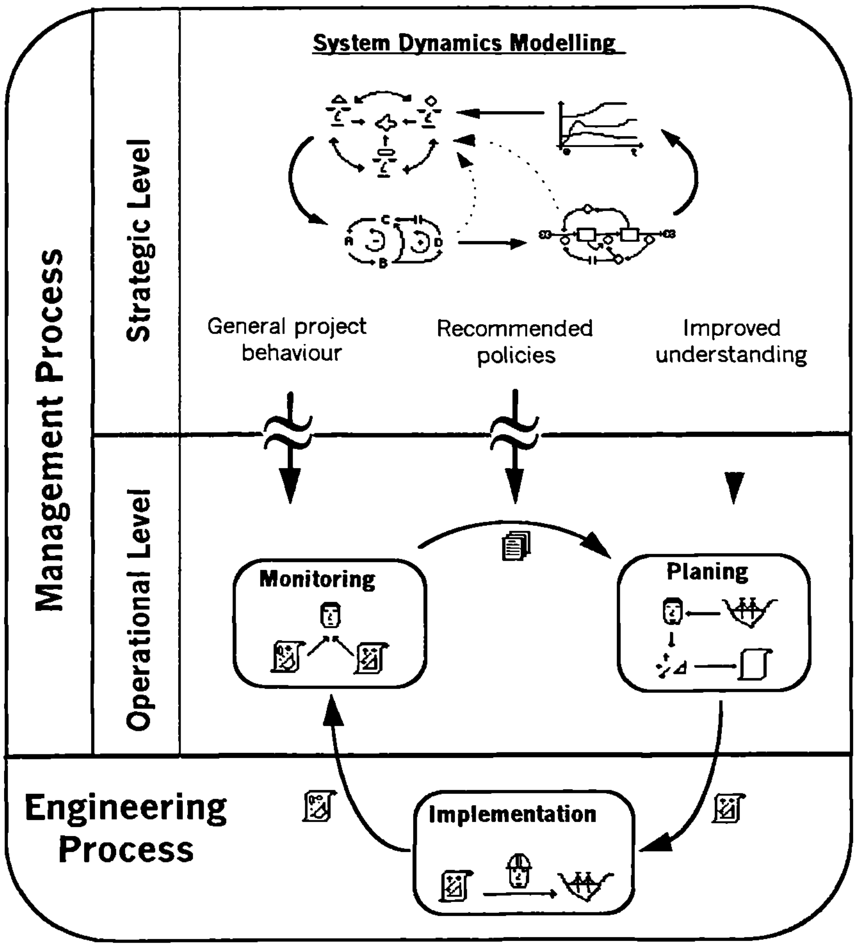


Figure 4.1 – Current application of System Dynamics in Project Management

While the use of SD models in this way may provide a useful contribution to the project management process, it is argued in this research that the most promising benefits are of synergistic nature and thus can only be achieved through a more rigorous and comprehensive integration with the traditional models. This should be based on the establishment of quantitative links between both types of models. The next section discusses the aims and requirements of an integrated conceptual framework to be proposed in this chapter.

4.3 Scope, general objectives and requirements

The overall aim of the conceptual framework proposed in this chapter is to establish a basis for the development of a "formal bridge" between the strategic and the operational project management levels, through the use of SD models. The relevance of this stems from two main aspects of reality: (1) strategic decisions have a crucial impact on project performance, and (2) their practical effectiveness depends on a successful translation into operating actions. The concept of "bridge" is here considered as any formal mechanism capable of providing a quantitative translation of managerial information between the strategic and the operational project management levels. This information includes any general insight, high-level policy, or detailed aspect of the project, which can be represented quantitatively in both the SD model and in the PERT/CPM based models.

It is assumed in this research that the project management process refers to the management of a single on-going project. This research also focuses on the role of quantitative SD simulation models. Therefore, the conceptual framework here proposed will refer to the application of these models to support the management of an on-going project.

The proposed aim of a formal integrated methodology is to provide a rigorous mechanism, which can be employed by any project organisation in order to develop and apply SD models to support the management of an on-going project, in an integrated manner within the traditional framework. This dual emphasis on model development and model application widens the scope of the methodology to include some critical aspects of System Dynamics (both in practical and conceptual terms), like model validation and model generalisation (these issues will be discussed later in this research). At this stage, the purpose of the conceptual framework here proposed is to provide a general basis upon which the practicality of an integrated approach can be explored. As discussed in chapter 3, this was done through extensive fieldwork. This framework will identify the general principles and mechanisms for the integrated use of SD models. A preliminary rationale is developed in the next section.

4.4 A rationale

4.4.1 System Dynamics Project Model: definition, concept and requirements

A critical issue for the scope of this research is the concept of a "System Dynamics Project Model". Some basic characteristics must be identified for a SD model to be considered as a "project model", suitable for the purpose of the integrated framework. Although this may involve a subjective judgement, a rigorous definition is required, as it will determine the sub-domain of SD models which can be considered as applicable within the framework. Because a project model is aimed at providing some type of support to the management process of an-going project, such definition will also shape the scope of the potential roles of a SD model.

In the first place, the aim of the integrated framework is to place the use of a SD project model within the generic project management process. Therefore, any specialised SD development aimed at addressing a particular and unique concern in a project, most likely will not be suitable for the purpose of the integrated framework. A SD project model must be "sufficiently generic" in order to fit into the general requirements of the project management process. Based on the review of the past applications presented in the previous chapter, some insights can be drawn into what should be the requirements of such SD project model.

The main generic aim of a SD model is to assess the performance of various managerial policies, in controlling the outcome of a certain "physical process". These policies are incorporated into the model as "decision rules". They monitor how the physical process is behaving over time, and further generate reactive actions, intended to keep the outcome within a desired targets. Under this perspective, the very basic requirement of any SD model is to simulate the two main processes of any managed system: the underlying mechanisms of the "physical system", and the managerial decision-making process aimed at controlling such system. The same principles apply to a SD project model: the "physical process" consists of the overall project organisation and material resources which are organised to accomplish the project work; the managerial decision-making process consists of the management functions of progress monitoring and re-planning, which aim at controlling the project towards the objectives. The basic requirement of a SD project model is therefore: "to incorporate and simulate the project work being accomplished, progress being monitored, and re-planning decisions being generated and implemented".

A typical and interesting issue addressed by SD models in modelling managerial decision-making processes, is the explicit mismatch between human perceptions and "real" reality. In most management systems, monitoring is not perfect. Therefore, management perceptions about reality differ from the real status of the system. While these misperceptions are in part caused by our own natural limitations, they tend to be aggravated by political issues. As an example, the design errors of a software system are difficult to detect early in the development life-cycle. However, the high number of errors which often escape to the later stages is typically aggravated by management reluctance in accepting early rework delays. This gap between management perceptions and "real" reality is a major issue in project management. The explicit consideration of this phenomenon is one of the novel features of SD project models. Where relevant, a SD project model should therefore model explicitly this mismatch.

Another major novelty of SD project models is the explicit incorporation of human factors and other intangible aspects of the project status. Typical examples are staff motivation, staff experience level, effects of schedule pressure, undetected errors and associated undiscovered rework. In reality, these subjective intangible factors have proven crucial for project performance and hence, where relevant, they must be considered in a SD project model.

Another major problem in management systems which SD studies usually attempt to analyse is the influence of "uncontrollable" factors. These factors can emerge at any time, and because they are uncertain, they are usually considered as risks. They are beyond direct and full managerial control. These risks can be external or internal to the system. As an example, a Client introducing requirements changes throughout the life-cycle of a project is an external risk; the staff achieving low productivity levels in the learning curve can be considered as an internal risk. Very often, projects run out of course due to risks. A good SD model supports the assessment of how well managerial policies cope with the impacts of such risks. Therefore, a SD project model must consider explicitly the presence of risks as uncontrollable factors, either as exogenous events or as intrinsic to the project system.

In summary, five main requirements are here proposed for the definition of a SD project model:

(1) it incorporates and simulates the product development process through which the project work, which is directly related to product design and realization, is implemented by the several human and material resources allocated to the project. The structure of this process depends primarily on the system development life-cycle adopted, and will be henceforth referred to as *engineering process*;

(2) it incorporates and simulates the decision-making processes through which management tries to control the project towards the desired objectives. This includes monitoring and assessing the work progress within the engineering process, detection of deviations from the targets, and generation and of reactive control decisions. The structure of this process depends on the management style, procedures and policies adopted by the project management team. This process will be henceforth referred to as the *management process*;

(3) where relevant, the model considers explicitly the mismatch between the human managerial perceptions about the project status, and the "real" status. It is essential that the model does not consider a control process using information about the project status to which in reality managers do not have access. Instead, the model must reflect a "less perfect" management process, as it happens in a real project;

(4) where relevant, the model must incorporate the presence of uncontrollable factors in the project, which will represent risks. These can be exogenous forces external to the project system (e.g. Client changes, delays from sub-contractors in delivering materials), or internal forces which reflect intrinsic properties of the project system (e.g. unexpected technical difficulties in product development);

(5) the model must consider and incorporate explicitly all the relevant human factors and intangible information about the project status, regardless of their subjective nature and of inherent difficulties in measurement and quantification.

In this research, a SD project model must comply with these requirements (more or less comprehensively), so that it is considered as suitable to be used within the proposed integrated framework and final methodology. The requirements of any model are necessarily related to its purpose. A SD project model as considered in this research is primarily aimed at representing a project "micro-world" and thereby it simulates the full process of project implementation and management, throughout the life-cycle.

4.4.2 Conceptual links

A first issue to address when integrating any two pieces of knowledge, is to identify the conceptual links that can be established between them. As previously discussed, System Dynamics models and the traditional PERT/CPM network models have a different perspective of the project. However, they also have some common features and hence relationships can be established.

In the traditional approach, several models are used in combination to represent and analyse the project. As described in chapter 2, at the core of these models are the WBS, the OBS, and the PERT/CPM based networks. PERT/CPM models are directly related with the first two, integrating elements of cost control, schedules, resource allocation, and work dependencies. By integrating all these elements within the traditional approach, network models constitute perhaps the most complete representation of a project. For the purpose of establishing links between the traditional models and the SD models, the term "network model" will often be used hereafter as referring to the traditional models.

Both SD and network models incorporate several elements of the project system and establish relationships among them. The conceptual differences between SD models and network models can be grouped into three categories:

(1) the range of elements of the project system considered explicitly in the model;
(2) the specific elements to which the model gives special emphasis to explain the project outcome;
(3) the type of relationships established in the model among these elements, also used to explain the project outcome.

Typically, the traditional models do not consider explicitly the influence of human factors (so called *soft* factors), nor the role of managerial control policies. Instead, they focus on the individual elements of the project work structure, explaining the project outcome as the linear accumulation of the individual results achieved in each of the WBS tasks. On the other hand, System Dynamics project models do not consider explicitly the details of the work and organisation breakdown structures, nor the several precedence dependencies between the work tasks. Instead, they focus on the role of human factors, with particular emphasis on the effectiveness of the managerial control policies. They explain the project outcome as the non-linear result of the continuous feedback interactions which take place among the project sub-elements, and in particular between the two processes of engineering and management.

However, despite the differences in the range of elements considered and the type of relationships established, both models have a common object of analysis: the project system and its outcome. The structure of both models is therefore directly related with the structure of the project system being modelled (i.e. sub-elements and their inter-relationships). Therefore, there can be a relationship between the structure of a SD model and the structure of a network model, as different representations of a common structured reality. Two projects with different structures would be represented by different operational network models. In the same way, these structural differences would also be reflected in the structure of the two different SD models. Although probably in less detail than in a network model, the structure of a SD project model must conform with the specific characteristics of the project being modelled. Therefore, the first conceptual link that can be identified between a SD project model and a network model is a "structural link": some level of formality can be established between the structure of both types of models when representing a same project.

Apart from providing a particular representation of the project, both types of models also incorporate quantitative data about the project system, and also produce quantitative data about the project outcome. This input and output data consist of quantitative descriptions about certain aspects of the project. Although at different levels of detail, both models incorporate input data regarding work schedules, resource allocation, and budgets. Although some of this data might not be the same (because the two models focus on different aspects of the project), there might be implicit relationships, which also impose a certain level of consistency between the data in the two models. As an example, if a SD model is set to recreate a project as described by a network plan, the productivity parameters assumed and considered explicitly in the SD model must be consistent with the budgets, schedules, and resource allocation considered in the network model. On the other hand, the staff profile planned for the project might be a common input to both models. The conclusion is that the input data considered in the two models must conform with a certain level of formal consistency: the common data must be the same in both models, and the remaining data must respect implicit relationships. The same rationale for data consistency can be applied to the output produced by the both models: the common results must be the same (e.g. final schedule and cost), and the remaining data must respect implicit relationships. Finally, there can also be relationships of consistency between the input data of one model with the output data of the other model. The degree of consistency between the data used and produced by both models establishes the second type of conceptual link: a "data link". Some level of formality can be established between the quantitative input and output data of the two types of models.

The integrated application of SD models and network models can now be discussed on the basis of these two categories of conceptual links: structure and data. Depending on the level of detail with which these links can be established, the integration of SD models within the traditional project management framework can be considered at different levels of formality. At the more informal level, the exchange of information between both models would be purely informal, without requiring any formal consistency between the structure and the data of both models (although there could still be some consistency, this would not be a requirement). Within this scenario, the SD models would be used to provide informal strategic guidance to the traditional project management process (as discussed in section

4.2). At the highest level of formality, several analytical links of data and structure would be established between both models. At this level, a formal verification of consistency between the SD model and the network model would be required, eventually based on and supported by automated procedures. Structural changes in the network plan would have to be checked and translated into changes in the structure and calibration of the SD model. The overall process of formal consistency checking and maintenance would be based on the analytical links established between the models. This can be complex in terms of detailed conceptualisation and implementation, and may have important impacts on the practical functionality of the traditional project management framework. A more detailed study regarding this "implementability" is required.

Based on this rationale, it is suggested that the formal integration of SD models within the traditional project management framework must be centred around the two conceptual links that can be established between the two types of models: structure and data. The level of formality of this integration depends on the detail of the specific links established. In this research, the conceptual implementation of formal links will be explored and further tested against practical implications through fieldwork.

4.4.3 Potential roles

Introducing a new element into a well established mechanism, in an attempt to improve performance, raises the question of what are the roles of this new element within the existing process. As discussed in the previous sub-section, SD models have common features with network models, regarding the way in which they represent a project and regarding their input and output data about that project. In chapter 2, the general range of applications of SD models was discussed and the distinctive contribution of the System Dynamics approach was identified. It is now necessary to identify the potential roles of SD models within the practical scope of the conceptual framework here proposed, as defined in section 4.3: to support the management of an on-going project.

The project management and implementation process can be described as a continuous control cycle of (1) estimating and forecasting likely results, (2) planning work schedules and resource allocation, (3) work accomplishment, and (4) monitoring progress against the plans. In the traditional approach, a wide collection of tools and techniques are used within this cycle to support the management functions (see chapter 2). Network models support cost and schedule estimating, risk analysis, re-planning alternatives, progress monitoring, and the assessment of project performance. The review of past SD applications discussed in chapter 2 has shown that SD models can also provide valuable support to these management functions. It has also been concluded that the SD analysis is usually undertaken at a higher level of aggregation, and hence it is more appropriate to provide support at the strategic management level. It is important at this stage to discuss whether the usefulness of SD models is specific to those past applications reviewed in chapter 2, or whether it can be considered as generic.

The range of analysis that a SD project model can provide depends in great part on the particular characteristics of the specific model: the range of factors and processes considered, and the level of detail. The required characteristics of a SD project model to be applicable within the proposed framework were discussed in sub-section 4.4.1. Based on these requirements, it is possible to identify its potential generic roles within the project management process.

The first requirement considered for a SD project model is to incorporate and simulate the engineering process of product development. The fact that the model simulates work being accomplished indicates that the project work structure is considered explicitly in the model, although at a certain level of aggregation. At the highest level, the whole project work would be considered as a single task to be accomplished within a single schedule and budget, and perhaps by a single project team – except for the PMMS (Pugh-Roberts Associates 1993), most models were developed in this way. A more complex model may well consider the decomposition of the product development process (and hence of the project work), into a set of sequential or parallel sub-tasks. Whatever the level of possible dis-aggregation considered, a SD project model simulates the implementation of a project's operational plan. Therefore, in supporting the planning function a SD project model can be used as a "test laboratory", where the performance of alternative work plans can be assessed within several scenarios, prior to implementation.

The second requirement considered for a SD project model is to incorporate and simulate the management decision-making processes. This indicates that, somehow, the model incorporates the characteristics of the decision-making policies and general control framework adopted by the organisation. In particular, it must incorporate explicitly the way in which work progress is monitored, and how control decisions are generated. Therefore, and still as a "test laboratory", the model can also be used in planning to assess the performance of the managerial policies used by management to control the implementation of a project's operational plan – the majority of the project models developed in the past focus on this second application, which is generically termed in System Dynamics as "policy analysis".

Applied as tool to assess the performance of the project work plan prior to implementation, and to assess the performance of the managerial control policies, the use of a SD project model can be integrated within the planning function of the project management process.

However, the use of a SD project model is not restricted to investigate future scenarios: it can also be used to "inspect" the project past. Here, the model is first set to simulate the observed past behaviour, and is further used to analyse "what-would-have-happened-if" type of scenarios. For a SD model to simulate the observed past, it must reproduce with some accuracy:

(i) how the project work was accomplished over time;
(ii) how problems occurred;
(iii) how these problems were eventually perceived by managers;
(iv) what corrective actions were generated;
(v) how these corrective actions were implemented.

The possibility of using a SD model to investigate past scenarios as a "diagnosis tool", provides three main distinctive benefits:

(a) it helps to identify the causes of observed problems;
(b) it provides extra information about the current project status (e.g. undiscovered rework);
(c) through retrospective "what-if" analysis, it provides the opportunity to verify whether different and eventually better results could have been achieved.

Applied in this way, the use of a SD project model can be integrated within the monitoring function of the project management process.

Apart from the two main requirements of a SD project model discussed above, three other characteristics were identified in sub-section 4.4.1. First, the model considers explicitly the mismatch between the managerial perception of progress and the "real" project status. This can improve the role of the model within the monitoring function as a diagnosis tool, by uncovering this "gap" about the project status and thereby improving management awareness about the likely real progress. In this same way, in planning the model can also forecast this gap and improve awareness about possible future progress. It was also identified that a project model must incorporate the presence of uncontrollable factors. This characteristic gives the model the possibility of assessing the impacts of risk factors both on past and future results. The behaviour of a project is often partially explained by the influence of what managers cannot control. Finally, it was identified that the model should consider explicitly all the relevant human factors and intangible information about the project status. This gives the model the capability of uncovering extra status information, like the amount of undiscovered rework, and in particular to characterise the human "soft" aspects of the project, like "staff fatigue" and "staff experience level". This widens the model's scope to explain the project past and future behaviour, and hence enhances its roles in both monitoring and planning.

In summary, two main distinctive potential roles can be identified for the use of a SD project model within the traditional project management process: within the monitoring function the model can be used as a "diagnosis tool" to investigate the project past, helping to uncover problems and identify its causes, while supporting process and policy improvement; within the planning function, the model can be used as a "test laboratory" to investigate future scenarios, where the performance of several planning alternatives and control policies can be assessed. These two general roles stem from the model's capability of simulating both past and future project scenarios, thereby providing a safe environment where managers can carry out experimentation. In order to accomplish these roles, a SD project model must incorporate explicitly the engineering process of product development, and the management control process of progress monitoring and re-planning. The use of SD models in this way implies that the information provided by the SD model should be incorporated into the management control process, wherein the traditional network models are in use.

Table 4.1 summarises the generic roles of a SD project model within the project management process, and how these relate with the model's characteristics defined in sub-section 4.4.1. The integrated use of a SD project model to support the management process of an on-going project can now be conceived as the model being continuously updated within the project control cycle, to reproduce the past and possible future scenarios, while providing support to the planning and monitoring management functions. This is illustrated in figure 4.2 below.

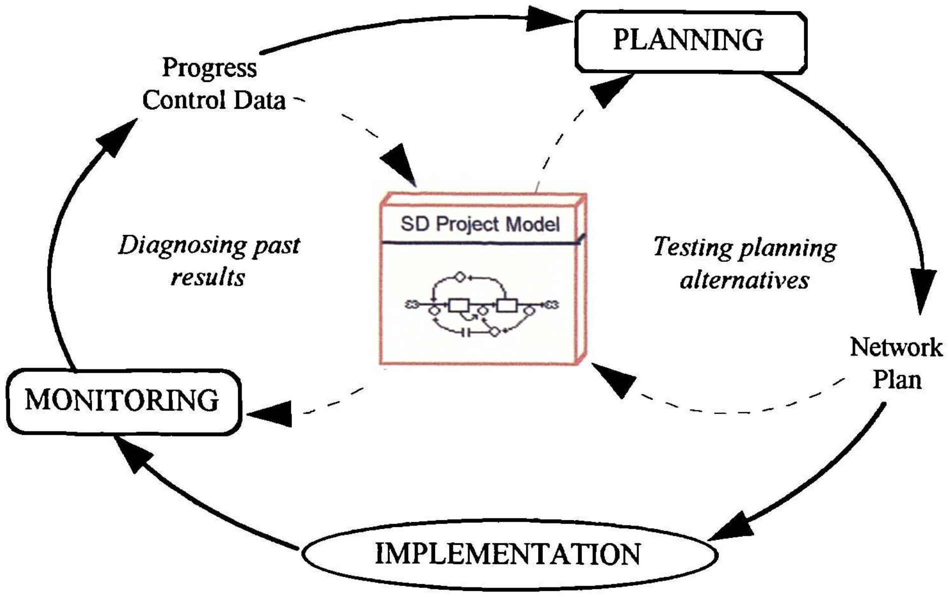


Figure 4.2 – The two general roles of a SD project model within the traditional project management process

Model characteristics	Model roles within the project management process	
	Monitoring: Diagnosis of past behaviour	*Planning*: Investigate future scenarios
1. Controllable factors incorporated		
1.1 engineering process: work plan for the product development process	a) identify causes for deviations related with the development process and work planning b) improve the development processes and work planning	a) assess alternative work plans and development process structures b) forecast work progress and completion date
1.2 management process: control policies of progress monitoring and re-planning	c) assess work progress d) identify causes for deviations related with the control policies e) improve management policies	c) assess alternative management policies
2. Uncontrollable factors incorporated		
2.1 exogenous to the project	f) identify causes for deviations external to the project	d) risk analysis: impacts of external disturbances
2.2 endogenous to the project	g) identify causes for deviations internal to the project	e) risk analysis: impacts of wrong estimates or unrealistic assumptions
3. Considers intangibles explicitly	h) uncover information about the project status, improving awareness of current progress	f) forecast extra information about the project status, improved awareness of possible future progress
Overall generic roles	a) identify causes for deviations, explicating past behaviour b) assess progress c) improve development process, work planning and managerial policies d) improve awareness of progress (uncover intangible information)	a) assess alternatives for development process, work planning, and control policies b) risk analysis: uncontrollable factors c) forecast project outcome d) improve awareness of future progress (estimate intangible information)

Table 4.1 – The generic roles of a SD project model related with the model's characteristics

4.4.4 Level of detail

Overview

A critical issue in the practical application of any modelling technique is the level of detail considered when representing the system structure in a model. This is particularly relevant in quantitative approaches, since the high level of formality imposes rigorous considerations. The level of detail of a model is here meant to be the degree of decomposition considered, to represent the system structure, its elements and interrelationships, and the resulting processes. The higher the level of detail, the "further down" the system is dis-aggregated into sub-components, which are then translated into an abstract representation in the model. The level of detail of considered in a model should relate primarily to its purpose: the model's structure should incorporate explicitly all the factors considered relevant to those aspects of the system behaviour which are of concern. However, achieving "the appropriate" level of detail is a complex and subjective issue, as it must also conform with many practical restrictions of model implementation, like model validity, quantification of subjective factors, and effort required among others.

On the other hand, the usefulness of a SD project model will probably depend on the level of detail considered in its structure. It is common sense to accept that the more the detail considered (assuming model "validity" is preserved), the wider the range of alternative scenarios the model can analyse. However, special attention must be given to both practical and conceptual implications. This is a complex issue which will be analysed in chapter 6 in more depth. At this stage, it is relevant to discuss the general impacts of the level of detail, in the integrated use of a SD project model within the project management process.

Level of detail in network models

Like any model of a project, the traditional network models can be developed at different levels of detail. In practice, it is a common procedure to maintain several networks updated throughout the life-cycle, representing the project at different levels of detail. The difference between each network plan usually relates to the level of decomposition taken from the WBS, where the several project sub-tasks are identified. While high level networks are useful to maintain a summarised view of the project milestones, the critical areas of activity and their interdependencies, the PERT/CPM critical path analysis is usually performed using the more detailed network. It is at this level of complexity that the whole project outcome is difficult to derive. Hence the need for the analytical aid of a detailed network model. On the other hand, it is also at this detailed level that the decisions about schedules, budgets, and resource allocation, relate directly to operating actions. Consequently, it is the detailed network plan that dictates the real pace of work accomplishment within the project. The underlying principle of the network analysis is based on an approach where the project is decomposed into its constituent elements, until these are perceived to be sufficiently simple so that schedules and budgets can be estimated with confidence. In this way, although a network model can be used to represent the project at any level of detail, it is at the lower detailed level that its analytical support is perceived to be useful and hence where these models are mainly used in practice. For this reason, when discussing the integrated use of SD models with the network models, it will be assumed that the latter are implemented at the detailed operational level. This issue will be revised in chapter 7.

Level of detail in SD models

The problem of the level of detail considered in a SD project model can be divided into two main related areas of analysis:

(i) the aggregation of the project work structure (represented in the traditional approach by the WBS), within the engineering process of product development, and

(ii) the detail of the managerial decision-making mechanism, within the management process.

With no attempt to provide a formal definition for the concepts of "high level" and "detailed" project models, the terms will be here used informally as referring to the two categories of models, which differ considerably in the level of breakdown of the project work within their structure. A high level model can be thought of one that considers the project as single task, which must be accomplished within a single schedule and budget, and which is managed by a single management function (e.g.

see Richardson 1980, Abdel-Hamid and Madnick 1991). On the other hand, a detailed model can be considered as one which breaks down the project into a set of interrelated sub-phases (or tasks), each possibly having its own internal management process, and eventually sharing higher-level management processes (e.g. Pugh-Roberts Associates 1993, Williams et al 1995). The terms "high level SD model" and "detailed SD model" will be henceforth used as referring to any specific model which falls into one of these two categories.

Level of detail in past SD developments

As reviewed in chapter 2, most of the SD project models developed in past applications are aimed at supporting high-level policy analysis. As a consequence, high levels of aggregation were considered in these models to capture the project WBS. In most cases, the project is considered as a single task that has to be accomplished within a single schedule and budget, and wherein some basic activities take place (typically product development, rework discovery, and rework accomplishment). While other more detailed models consider a breakdown of the project into sub-tasks, typically according to the phases of the life-cycle, they incorporate only the detail required to allow the explicitly analysis of the relevant high level management policies. In some cases, the models consider the management process as the simple trade-off between the two policies of hiring staff and delaying the final project schedule.

With the practical application of SD focused on policy analysis, there has been little emphasis on using SD project models to support detailed work scheduling and resource allocation, the major domain of the traditional network models. On the other hand, while more detailed models have already been used (e.g. Cooper 1980), recent developments have been moving towards an increasing model detail and complexity (e.g. Williams et al 1995, Lin 1993, Pugh-Roberts Associates 1993). This trend probably reflects a natural evolving process, where investing in more detail follows the gain of confidence from more simple experiences. However, a more detailed model is not necessarily more useful: practical experiences show that there are situations where simpler models are likely to perform better (Alonso 1968). Different models can be considered for a common problem at different levels of detail, each providing its own benefits. Likewise, in System Dynamics both high-level and detailed project models have their own strengths. A combined use could be the best way ahead.

Implications for policy analysis

The aggregation of the work tasks in the product development process restricts the range of analysis regarding work scheduling and resource allocation. As an example, a SD model that considers the whole project as a single task cannot support the analysis of intermediate schedule delays, nor the re-allocation of resources among the several project sub-tasks. On the other hand, the level of detailed considered within the management process restricts the range of managerial policies that can be tested and assessed. As an example, a model that does not consider the possibility of increasing parallelism within the project, cannot be used to test the impacts of this decision. Furthermore, the level of detail considered in these two main areas of the model is strongly inter-related, since the aggregation of the project work structure restricts the scope of scheduling and resource allocation policies. In general, the higher the aggregation, the narrower the scope for policy analysis.

Implications for data collection

The main practical implication of a high level model is the data required in order to develop, validate and use the model. A high level perspective implies more aggregation of the system structure, and hence the data required is also more aggregated. Typically, a high level model benefits from structural simplicity, hence requiring less data. On the other hand, aggregated data is usually more likely to be collected within a management system, and tends to be more "robust" in terms of accuracy. Given these implications in terms of data, a high level model is usually easier to develop, validate, use and maintain. Within the context of a SD project model, these benefits are relevant since projects are implemented within highly time-pressured environments. The practical use of a SD model must respond effectively to management demands for quick and reliable information.

Implications for the scope of the analyses

Another important characteristic of a high-level SD model is a wider scope in terms of system boundaries. Although the areas of system activity are represented in a highly aggregated manner, the boundaries of the model tend to be enlarged. The analysis is focused on the general interactions between the system's sub-components, where "almost everything matters" – this endogenous perspective is the basis of any System Dynamics study. In this way, a high level SD project model has therefore the potential benefit of covering the full project life-cycle, being able to provide long-term project analysis. As an example, it is possible to assess the full impacts of risks during the early stages in the life-cycle. A high level model can also works as "a map", preserving a clear view of the whole project. A typical problem with detailed models is that managers may lose the sense of the "big picture". As a tool for policy analysis, the main purpose of a high level SD project model is to provide strategic recommendations and directions, which can be used to guide detailed planning at the operational level. Finally, the model incorporates management decision roles at the strategic level, and thereby it encourages high-level policies to be made explicit, preventing these to remain "hidden" in managers often defective mental models.

In summary, a high-level SD project model usually requires less effort to be developed and applied in practice, benefiting from both structural and data simplicity. The scope of the model can easily cover the full project life-cycle, providing long-term analysis. As a "map" it preserves a clear view of the whole project. In supporting the management process of an on-going project, the model can be used as tool for policy analysis at the strategic level, allowing managers to share their mental models and test the consequences of their "beliefs".

Detailed SD models

In more recent applications of SD to project management there has been tendency to explore the benefits of more complex models. This increase in complexity is primarily reflected in the level of detail considered to represent the project work structure. Typically, there is a breakdown into a set of main phases or stages, where different types of activities take place continuously. The most complex application of this type is perhaps the PMMS tool (Pugh-Roberts Associates 1993), which decomposes large programs into several main phases. According to the specific characteristics of the work within each phase, different types of generic sub-models can be used, all having the "rework cycle" at the core of their structure (Cooper 1980). These sub-models are then combined into a single integrated project model. In the practical cases reported using the PMMS, despite the breakdown of a project into sub-phases, the individual sub-models still assume a strategic, high-level perspective of the project areas they represent. It is argued in this research that the possibility of moving the application of SD project models "further down" in the level of detail, as in the PMMS, should be explored.

In principle, as far as a continuous view of the processes remains "valid", it is possible to consider in a SD model the breakdown of a project into a network of sub-tasks, each being modelled by a specialised sub-model. In this way, the resultant project model would incorporate explicitly some form of work dependencies, as PERT/CPM models do. The model could then be used to analyse work scheduling and resource allocation in more detail. The quantitative results produced by this model could then be more easily translated into the operational work plan. However, in practice several restrictions impose that less detail can be considered than in a detailed PERT/CPM network. In large projects, detailed logical networks often incorporate thousands of tasks. It is easy to anticipate that having such a large number of SD sub-models linked together, forming a large complex model, would not be feasible due to conceptual or even technological restrictions. It is here argued that a detailed SD project model should lay somewhere between the higher strategic level and the operational level where network models are in use.

This issue of how "further down" a SD project model can go in the level of project decomposition is important, and it will be discussed later in more depth. At this stage, it is here argued that overall detailed models can be developed. The implications for an integrated use with the traditional network models is now discussed.

Implications of SD detailed models for the integrated framework

Developing detailed models raises some critical issues of practical implementation. Considering more detail usually results in a larger amount of data being required to calibrate and use the SD model. In particular, it may require detailed estimates for the long term project future. In most situations, the accuracy of an estimate reduces considerably as it moves towards the future. The obvious consequence is a shorter time range into the future, within which the detailed SD model can be used with confidence. On the other hand, the type of data required by a detailed model is less aggregate. This type of data is usually more costly to collect and less likely to be available. As a consequence, while a detailed SD model can cover the full project past, in terms of future it may only cover the incoming phases to which a detailed operational plan is available. This is also consistent with the current use of network planning tools: a high-level network is used throughout the life-cycle to represent the whole project, while detailed networks representing the future are only available within a short time horizon.

Like in any modelling technique, the balance between detail and time horizon must be handled according to management needs, and thus the purpose of the model. It is here suggested that a detailed SD project model can be used to help managers translating strategic insights into operational decisions. This model would cover the full project past, but only the early incoming project phases. Some distinctive benefits can be identified for this type of application:

- the information produced by the model (in particular the numerical data), is closer to the operational level, thus enabling managers to understand better how high level strategic decisions relate to actual operational actions;
- since the model incorporates more detailed information about the project, it can be argued that there is potential for the estimates produced to be more accurate;
- while decomposing a project into detailed sub-components, these still hold enough complexity to "suffer" from systemic effects. A detailed model would allow managers to analyse these effects individually within each project sub-area. Therefore, while providing a more detailed view of the project sub-areas, the general benefits of the holistic perspective are preserved.

Summary and conclusions about the level of detail

Regarding the level of detail to be considered in the SD models, while most of the SD project models developed in the past focused on high level policy analysis, there has been a recent move towards the use of more detailed models. High level models provide several benefits, and in particular require less effort to be developed and applied in practice. They preserve a clear aggregated view of the whole project, and have the inherent flexibility of providing quick "what-if" analysis of various project scenarios. As a tool for "policy analysis", their distinctive contribution to the project management process stands at the strategic level, where important high level decisions are undertaken prior to the development of a detailed operational plan.

On the other hand, a detailed project model holds the conceptual attractiveness of incorporating more aspects of the real system, thus producing more detailed output. However, System Dynamics is based on a holistic and continuous perspective of systems, and hence special attention is required to assess the appropriateness of such a level of detail. The development and maintenance of this type of models can be expected to be more expensive, and their practical application is more demanding in terms of data requirements. Calibration for the project future is likely to raise problems in terms of data availability, and hence the scope of the model should be restricted to a short term future. On the other hand, the numerical output produced and the qualitative insights gained are closer to the operational level, and hence a more direct translation into operating decisions is possible. This is a critical for the integrated use of a SD model with the operational network models, as it helps in the definition of quantitative links between both types of models.

Any SD project model will always assume a higher level of aggregation than a detailed PET/CPM network. Nevertheless, perhaps a detailed SD model can work as an interface between the strategic and the operational management levels. While such model would focus on the main project sub-areas, it still provides the distinctive SD perspective, where internal interactions and resulting feedback processes are the key to explain the project outcome.

In conclusion, both high-level and detailed SD project models can provide a distinctive contribution to the project management process. The level of detail assumed in a SD model reflects a specific balance between detail, and time-horizon. Both types of analysis are important within the management process: a long-term aggregated view, combined with a shorter term detailed view.

Within the context of an integrated framework, a combined application of both types of SD models with the traditional tools is here suggested as an appropriate approach. From a high level SD model, where managers "look far" into the future, followed by a more detailed SD model where managers "look down" to convert this image into a more detailed view of the short term future, and finally to a network model where specific operating tasks are scheduled and budgeted, all the three models can provide a distinctive contribution to support and improve project control. Such a combined use of these three models could be formally integrated through the definition of analytical links. This integrated view, crossing from the strategic level down to the operational level, where the models become more detailed and of narrower scope, is illustrated in figure 4.3 below.

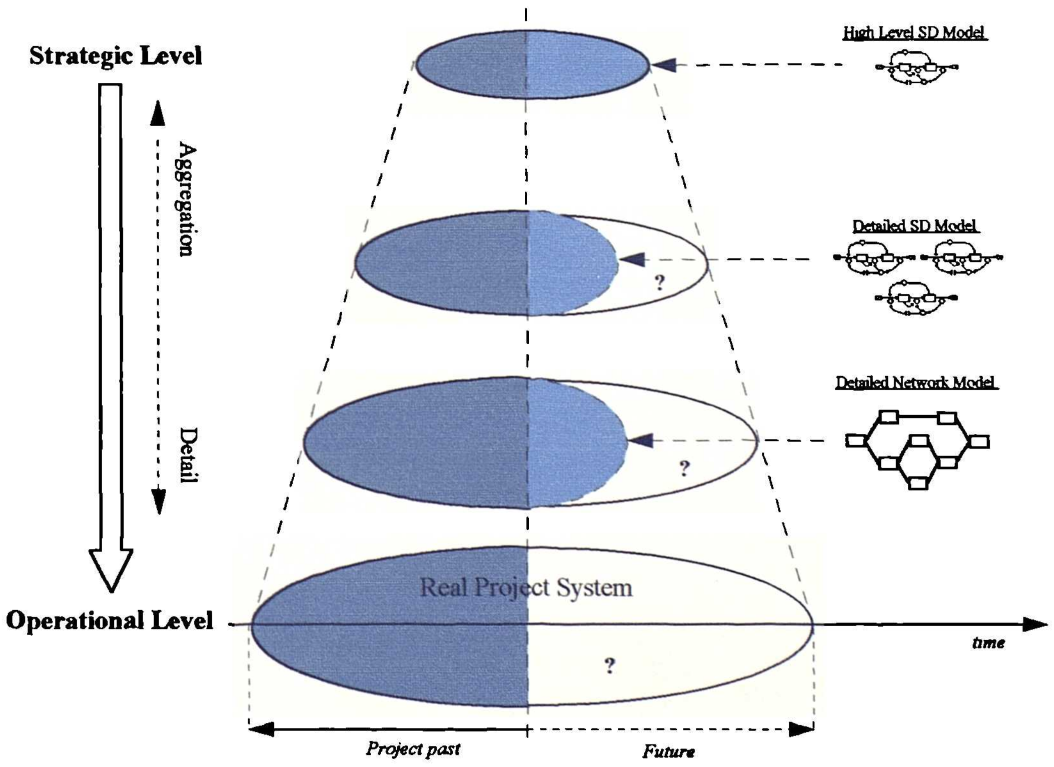


Figure 4.3 – Overview of the use of SD models and network models at different levels of detail

4.4.5 Summary of rationale

The rationale discussed above was aimed at identifying the main conceptual foundations upon which the integrated use of SD models will be considered. The review and analysis of the major past applications of System Dynamics to project management provided the basis for this rationale.

The following main issues were addressed:

(1) SD models and network models provide distinctive representations of a project, incorporating common information and producing estimates of similar nature. What kinds of relationships can therefore be established between the two types of models?

(2) SD models have been used to estimate the general outcome of a project under different managerial and risk scenarios. Their "what-if" functionality has been used to investigate both the future and project past. Based on this application, what can be the potential roles of SD models within the project management process? The answer to these questions inevitably led to a critical issue in SD modelling: the possible and appropriate level of detail of the model. If different levels of detail can be considered for a SD project model (as observed in past applications), what are the implications for an integrated use with the network models?

At this stage, an initial clarification was attempted, having led to the following conclusions:

(i) both SD project models and network models can represent a same project, incorporating and producing numerical data regarding the plans and the actual project outcome. Their integrated application can be based upon two types of conceptual links that can be established: structural and data links. Depending on the level of detail of these links, SD models can be incorporated within the traditional approach at different levels of formality. Ideally, several analytical links could be established between both types of models, with their joint application being supported by a formal "consistency checking" and "data transfer" mechanism;

(ii) two main roles can be identified for the use of a SD project model within the project management process: (1) in planning, the model can be used as a "test laboratory" to investigate future scenarios, and (2) in monitoring, the model can be used as a "diagnosis tool" to analyse the project past. In the first application the model can be used to test the plan's robustness to risks and the effectiveness of managerial policies. In the second application, the model can be used to uncover information about the project status and to identify the causes of past behaviour. The integrated use of a SD project model to support the management process of a on-going project, is based on the model being continuously updated to reproduce both components of the project outcome: past results, and planned future;

(iii) different levels of detail can be considered in a SD project model. The level of detail assumed reflects a specific balance between detail and time-horizon: the further ahead the model "looks" into the future, the more aggregated it is the view of the project. It is argued that two main different levels of detail can be considered for a project model, each providing a distinctive contribution to the management process. A high level strategic model can be used as tool for "policy analysis", providing quick assessment of long term impacts of complex decisions and risks. A more detailed complex model, focusing on the shorter term aspects of the main project sub-areas, can be used as an interface between the strategic and the operational network level. The combined use of all these three models can be based on the definition of formal structural and data links.

With these conceptual remarks as the core rationale for an integrated approach, the next section proposes a structured integrated framework.

4.5 A conceptual integrated framework

4.5.1 Overview

Having identified the conceptual links between SD models and network models, discussed the distinctive roles of the SD models within the traditional project management process, and having concluded that two different types of SD models can be used, the next step is to establish a well structured mechanism so that the integrated use of SD models can be explored in practice. The framework here proposed is "conceptual" in the sense that it aims at establishing some general principles, without imposing strict rules to the overall integrated process. As proposed by the research approach being adopted, a case-study will be used as fieldwork to refine this initial conceptual model.

4.5.2 The view of the project implementation process

As the background for the integrated framework, the project implementation process is considered as being comprised of two parallel interrelated sub-processes: engineering and management. The engineering process consists of all the activities necessary to develop the product, which are usually interrelated according to a structured development process. The management process consists of all the decision-making functions that aim at directing the project outcome towards the desired targets.

Within the management process, decisions are undertaken at two different levels: (1) at the strategic level managers assume a high-level perspective, focusing on complex problems that can affect the whole project. The long-term objectives are revised, and managers decide about general directions for the work in the field. This is usually supported by the specification of a high-level plan, which comprises the project's major tasks and dependencies, schedule milestones, budgets, resources, and capital requirements; (2) at the operational level, managers focus on the detailed short-term problems of implementation. At this level, a detailed network plan is used to direct the work within the engineering process of product development.

Two main control functions are considered within the management process: planning and monitoring. In planning, actual progress is compared against the plans, and where deviations occur corrective actions are generated. In monitoring, progress information is collected from the engineering process and is compiled into an appropriate form to be used in planning.

Overall, it is assumed that the management process consists of a periodic control cycle, within which the management functions of monitoring and planning alternate with technical implementation in the engineering process.

4.5.3 Use of SD models

At the strategic management level it is proposed the use of a high level SD strategic model (SDSM). This model covers the full project life-cycle, perhaps discriminating only the major milestones, according to the project high-level plan.

At the operational level, it is proposed the use of a detailed SD operational model (SDOM). This model comprises several sub-models, and assumes a breakdown of the project into a set of individual but inter-related sub-tasks, in consistency with the WBS.

In this way, both models have structural links with the traditional WBS, the systems development life-cycle, and with the network plans. The models are used within the management process to support the two basic functions of planning and monitoring. As mentioned in the previous sub-section, their general role in planning is to test alternative implementation scenarios (i.e. work scheduling, resource allocation, and control policies), while in monitoring they are used to diagnose the project past (i.e. refine progress information, and identify causes of past results).

While the SDSM "looks far" in little detail, to cover the whole project and analyse the major trends, the SDOM model "looks down" in more detail to focus on the individual project sub-tasks, supporting more detailed planning and monitoring.

Figure 4.4 illustrates the proposed integrated framework, where the SD models are used within the project management process at both strategic and operational levels, in the way described above.

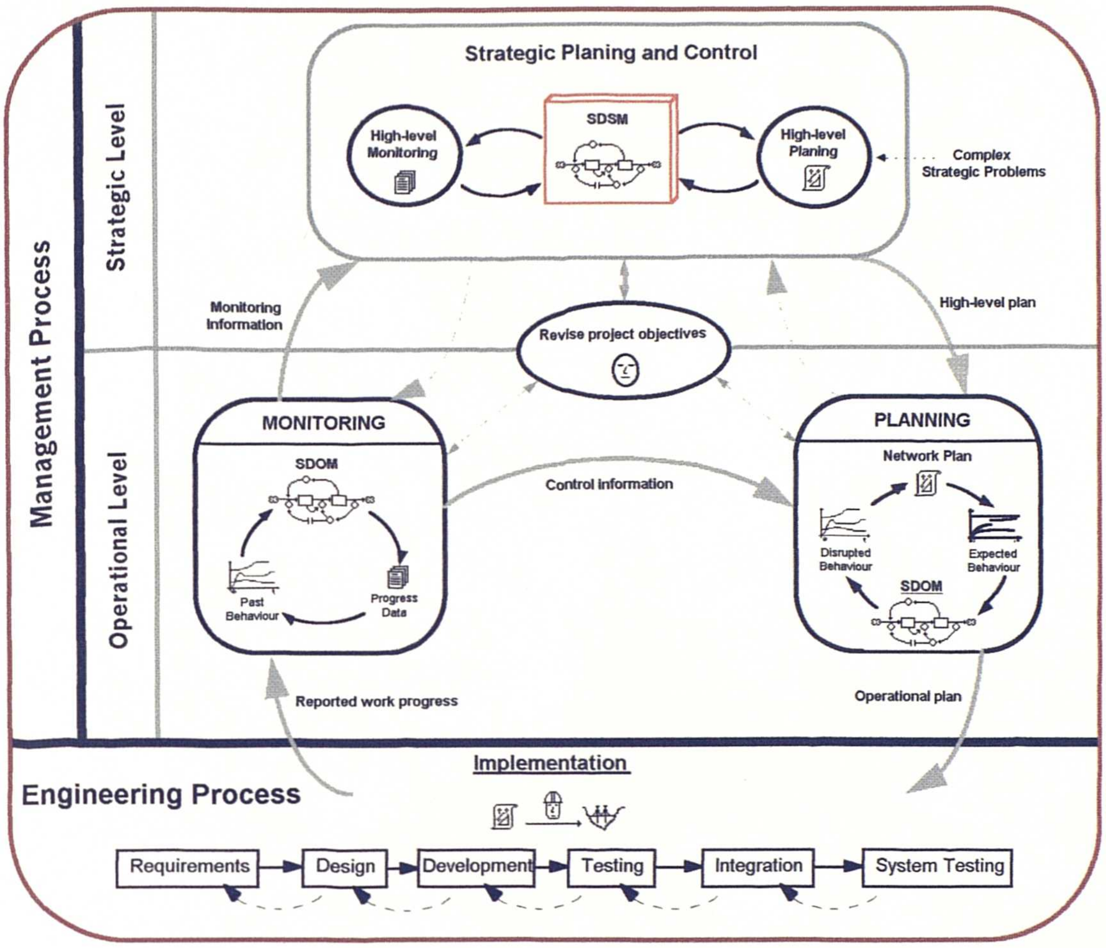


Figure 4.4 – Overview of the proposed project management integrated framework

4.5.4 The project behaviour and the links with the traditional models

The use of the SD models in this way is implemented through the establishment of data links with the traditional models. These links are centred around the concept of project behaviour.

A project is a complex system, incorporating many physical and human elements. A project being implemented changes its status continuously over time. This change of status can be represented by the behaviour patterns (i.e. changes over time), exhibited by some of its dynamic characteristics (e.g. staff level, current budget, work accomplished). This set of patterns forms the project behaviour. These dynamic characteristics of a project may fall in two main categories: management, and engineering. Management characteristics include managers' perceptions of progress (e.g. cost at completion, earned value), and re-planning decisions (e.g. schedule adjustments, re-allocation of resources). The engineering characteristics refer to the physical process of product development: amount of work accomplished, rework discovered, quality of work accomplishment, and others.

A PERT/CPM network plan also assumes changes in the project status over time. Hence, it portrays implicitly the image of a project evolving successfully towards constant targets of cost and schedule. This "successful" behaviour is hereafter referred to as "planned behaviour", and establishes the data links between a network and the SD model, within the planning function: both models must produce explicitly, or portray implicitly, the same expected project outcome, as described by the planned behaviour. Consequently, both models must also incorporate the same common inputs that characterise the project plan (e.g. staff profile, work schedules).

As the project evolves, progress data is collected in the monitoring function. This data captures changes in the project status and hence it portrays the project past behaviour, where changes in the targets may occur (e.g. schedule delays, over-expenditure, use of extra staff). This observed behaviour establishes data links between the monitoring tools and the SD models: both monitoring data and the SD model must produce explicitly, or portray implicitly, the actual project outcome as described by the past behaviour. Consequently, the inputs used in the SD model must also match their "real" occurrences in the project, as derived from the monitoring data (e.g. staff productivity).

The terms "produce explicitly" and "portray implicitly" used above reflect an important difference between the SD model and the traditional models regarding the concept of project behaviour. While the main output produced by a SD model consists explicitly of a set of behaviour patterns plotted over-time, the traditional models give little or no emphasis to this type of description. However, regardless of the use of a SD model, a project being implemented always exhibits a dynamic behaviour. Therefore, a network plan implicitly portrays an expected project behaviour. Several patterns can be derived from a network plan, and from progress data collected in the monitoring function. In the traditional control framework, cost charts are used to represent the evolution of costs over time (e.g. BCWP, ACWP, and BCWS). Other types performance analysis charts are also used to represent some dynamic changes in the project (e.g. plotting schedule performance against cost performance; Nicholas .1990). However, in general, most of the dynamic changes occurred in a project are not considered explicitly in the traditional approach. For this reason, the term "portray implicitly" is here used as referring to the traditional models, whereas the term "produces explicitly" is better applicable to the SD models.

4.5.5 The roles of the SD models

According to the conceptual framework proposed, both SD models, the SDSM and the SDOM, are used to support the planning and monitoring functions. The strategic analysis using the SDSM will naturally precede the detailed analysis at the operational level using the SDOM (with several iterations among them being possible). In practice, the interaction between the use of both models is important. However, for the sake of simplicity their individual application in planning and monitoring is explained first.

In planning, the SD model is used as a "test laboratory" to assess the performance of the current plan within different scenarios. While a network plan portrays the image of a project evolving successfully towards desired constant targets, the SD model provides a means to explore and analyse possible scenarios wherein these targets may "slip by". In this way, the model is first set up to reproduce the expect behaviour as portrayed by the network plan, and disturbances are then introduced and tested. The model provides the explicit description of the possible "unsteady" outcomes. Planning alternatives can be experimented, with several readjustments being carried out in the SD model to reduce the plan's sensitivity to risks. The more effective planning changes suggested by the SD analysis are selected by managers (as being the more realistic and "robust"). They are translated into the detailed network plan and implementation may then take place. The added value of the SD analysis in planning in this way is to provide a quick and realistic way of assessing risks, anticipating possible scenarios for disruption, and testing the impacts of major decisions prior to their translation into the detailed plans.

In monitoring, the model is used as a "diagnosis laboratory" to investigate the project past behaviour, helping to identify the causes for possible deviations. Once progress monitoring data has been collected using the traditional tools, the SD model is set to reproduce the project past behaviour. Estimates of undiscovered rework can be produced, suggesting adjustments in the progress performance indexes, thereby avoiding "over-optimism". Through "what-if" analysis, the model can be used to investigate the causes for eventual deviations, and to test whether alternative planning and control policies could have provided better results.

Each time the model is used to diagnose new "segments" of past behaviour, it also provides new estimates for the future behaviour, which can differ from the current plan. This might anticipate that, given the past results, deviations from the current targets are now likely to occur somewhere in the future. As an example, after a certain milestone has been achieved on schedule, traditional monitoring procedures might suggest that both effort spent and work progress are well according to the plan. This analysis reveals the ideal performance of a steady progress, with no concern of readjusting the work plan. However, achieving these good results might have been at the expense of using of excessive schedule pressure. The consequence can be staff fatigue and low work quality, with a large amount of undiscovered rework moving onto the next stages. The impacts of this type of intangible factors are not the subject of explicit analysis within traditional models. The use of the SD model in this way provides a richer analysis, with the potential to anticipate problems before any observable progress data suggests so. This is clearly a more proactive approach to project control. Once the past behaviour has been investigated and progress has been assessed, control information is generated, and a new cycle starts in the planning function.

In summary, in the planning function the models are used to estimate the project outcome and help to identify realistic and "robust" re-planning decisions. In monitoring, the models are used to estimate undiscovered rework, and as a diagnosis tool they help to identify possible causes for observed deviations; through retrospective "what-if" analysis, managers can explore how the current processes and policies can be improved in the future. Both SD models (SDOM and SDSM) are used in this way, but at different levels of detail.

4.5.6 Interaction between the two SD models

In an ideal application, both SDSM and SDOM would be used in parallel. The SDSM captures less detail and hence demands less information. The estimates provided by this model should be regarded as general forecasts and not as detailed predictions. The purpose is to provide a strategic analysis of the whole project, particularly at the early stages, when a detailed plan is not available for the middle and later stages. The SDSM allows managers to keep a concise global view of the project outcome, throughout the life-cycle. Whenever major changes occur in the project, and re-planning is required, the SDSM provides a preliminary analysis of the long term impacts, before the detailed plan is readjusted. This strategic guidance may prove vital as it prevents expensive counter-productive efforts, which so often result from inappropriate strategic decisions.

The SDOM provides a more focused analysis at the operational level, where strategic decisions can be refined into more detail, prior to their translation into the network plan. This model demands more detailed information, and can only be used to cover those stages where a detailed plan is available (in practice, this model is not likely to cover the full project life-cycle, until the later stages are reached).

The interaction between the use of both SD models is important. While their level of detail is different, they represent and reproduce the behaviour of the same project. The analyses using the SDSM is likely to be simpler and quicker to implement. Therefore, it should precede the more detailed analysis using the SDOM. The same experiment implemented in both models may provide different conclusions and insights: planning decisions that appeared to be appropriate in the short term using the SDOM, might prove inadequate in the long term by using the SDSM. Therefore, both models should be used in combination so that the full impacts of decisions are assessed in both the short term and in the long term. On the other hand, aggregating the monitoring data used in the SDOM may provide valuable input to the SDSM. Overall, and like in the network approach, structural and data links can be established between the two SD models, to guide their joint application.

4.5.7 Summary

The conceptual framework here proposed outlines the general principles for the integrated use of SD models within the traditional project management process. These principles can be summarised as follows:

(i) the project implementation process comprises two inter-acting sub-processes: (a) engineering, where technical product development takes place, and (b) management, where project control is implemented at two different levels; at the strategic level, complex issues affecting the whole project are analysed and general directions are set up for the project; at the operational level detailed issues of implementation are analysed within the short term project milestones. Two main distinctive control functions are considered in the management process: progress monitoring, and planning. These management functions alternate periodically with the implementation of technical development in the engineering process;

(ii) two SD models can be considered: a high level model at the strategic level, covering the full project life-cycle, and a detailed model focusing on the individual project sub-phases. Both models are used in a similar way, and consistence with the WBS and network plan;

(iii) both SD models are used to support the two management functions of planning and monitoring: in planning several future scenarios can be tested and risks are assessed, while in monitoring the project past is diagnosed;

(iv) the integration of the SD models with the traditional tools within planning and monitoring is centred around the concept of project behaviour. A network plan portrays implicitly the image of a project with dynamic characteristics, evolving successfully towards stable targets. As the project progresses, monitoring data provides the description of its actual past behaviour. Both SD models and traditional models must portray or produce quantitatively the same behaviour patterns for the project;

(v) in the planning function, the models are used to estimate the project outcome and help to identify "robust" planning decisions. The added value of the SD analysis in planning is to provide a quicker and more realistic way of testing the impacts of risks and of major decisions, prior to their translation into the detailed network plan, by being able to anticipate possible disruptive scenarios;

(vi) In the monitoring function, estimates of undiscovered rework can be produced, avoiding "over-optimism". This analysis allows problems to be anticipated before observed progress data suggests so. Through retrospective "what-if" analysis, managers can also identify causes for deviations and explore how the current processes and policies can be improved in the future;

(vii) both models (SDOM and SDSM) are used in parallel, in the same way, but at different levels of detail. The SDSM provides strategic analysis of the whole project, particularly at the early stages, allowing managers to keep a concise global view of the project outcome, throughout the life-cycle. It provides preliminary analysis of the long term impacts of decisions, before detailed re-planning is carried out. The SDOM provides a more focused analysis at the operational level, and is not likely to cover the full project life-cycle until the later stages are reached. This model allows strategic decisions to be refined into more detail, prior to their final translation into the network plan;

(viii) the interaction between the use of both SD models can provide synergistic results. The same experiments implemented in both models may provide different conclusions and insights. Structural and data links can be established between both models, to guide and improve their joint application. Conclusions and insights derived from one model should be tested in the other. By comparing the results, a more in-depth understanding can be achieved.

The scope for the integrated use of SD models as described in the conceptual framework here proposed is not intended to cover all the potential applications of System Dynamics to project management. As mentioned in section 4.4, the framework is restricted to the use of project models which must conform with the proposed requirements. In the proposed framework, these models are applied as control tools to support the management of a single on-going project. The use of SD models in this way is not intended to replace any other tool, technique, or procedure, which can provide its own useful contribution to the project management process. Other possible applications of System Dynamics, like interactive gaming, or developing simple models to create learning environments in teams (Morecroft 1994), may also provide a complementary and distinctive contribution, outside the scope of the conceptual framework here proposed.

Finally, the practical implementation of this framework can be considered at different levels of formality, depending on the characteristics of the specific project organisation and environment. This includes the structuring of the management processes, the type of information used and made available by the management system, the formal tools and techniques already in use, the organisational structure, the management practices and even the organisational culture. All these aspects must be considered carefully, before deciding about the level of detail to implement the framework.

4.6 Future steps in this research

In this chapter, a rationale was developed for the integrated use of SD models within the traditional project management process. Based on this, a conceptual framework was further proposed. As discussed in chapter 3, the next step in this research is to explore the practical implementation of this framework and refine it towards the development of a formal methodology. As it will be described in chapter 5, for this purpose, an extensive fieldwork was conducted in a case-study project.

In practical terms, the framework itself constitutes an already complex model, since it comprises the use of SD project models within a real project management environment. In general, these environments are large, complex, and usually work under extreme conditions of time-pressure. The strategy followed in this research was to use a real project as a case-study where, in a first stage, the conditions to implement the framework would be created; in a second sage, its application would be attempted. Based on the practical results produced, the overall framework would then be refined and improved into a final formal methodology.

A key requirement to implement the framework is the availability of appropriate System Dynamics project models. In this chapter it was discussed that these models must conform with certain requirements so that they can be used in the way described by the framework. In chapter 2, the System Dynamics methodology was described as comprising the development of a quantitative simulation model, as a final important step. This model should be based on the previous qualitative steps of problem identification, and causal analysis using influence diagrams. The ideal of the proposed framework is that project models are available as early as possible in the project life-cycle, preferably prior to product realization has been initiated. This has important conceptual implications to whether a full project model can be developed and validated at that stage, or whether models developed for past projects can be generalised for application in future projects. Therefore, under a practical perspective, a structured method for the development and validation of project models is an important complement to the integrated framework proposed in this chapter. In chapter 6, it is argued that such method is not available and therefore a structured approach is proposed as part of this research. The issues of model re-use, model generalisation, model conceptualisation and validation are discussed in some detailed in chapter 6. This structured approach was developed and used within the fieldwork case-study, as the basis to develop the required SD project models.

5. The KDCOM Case-Study

5.1 Introduction

In the previous chapter a conceptual integrative framework was proposed, where the use of SD project models is embedded within the traditional project management process. This conceptual framework is based on a control process wherein the SD model complements the traditional models thereby enhancing the planning and monitoring functions of project control. The SD model is formally integrated with these models, in particular with the PERT/CPM logical networks. Underlying this integration is the establishment of analytical links between the SD model and the PERT/CPM network. A brief summary of this framework was previously presented in Rodrigues and Williams (1996), and was further detailed in Rodrigues (1997).

Once this framework was developed, the main aim of this research was to further refine it into a more detailed and formal methodology – hereafter referred to as SYDPIM (System Dynamics-based Project-management Integrated Methodology). The underlying logic of the conceptual framework was based on two main elements: the conventional project management control process, and the possible ways in which a SD project model can be used to support this process. The first element is extensively described in the literature (e.g. Nicholas 1990, Turner 1993, PMI 1996, Williams 1996, Kerzner 1997). Regarding the latter, in most of the past applications of SD to project management, the SD model has been used either divorced from a real project (e.g. Roberts 1964), or for post mortem analysis of a past project (e.g. Cooper 1980, Williams et all 1995), or even in the most impressive applications, totally divorced from the traditional approach. For example, Cooper (1993) claims the use of a SD project model to support an on-going project, but in replacement of traditional PERT/CPM tools which, he claims, are a complete failure. Lin et al (1995) also propose the use of the SD model to support on-going projects, but still without suggesting any specific framework to explain how the model is embedded within the traditional control process. In this way, since the conceptual framework proposed in the previous chapter considers the continuous support to an on-going project, the ways in which the SD model is to be used were, in great part, logically derived. Therefore, while this was supported by some evidence extracted from these past applications (e.g. the SD model can provide reliable estimates; Cooper

1980, Pugh-Roberts Associates 1993, Williams et al 1995), it was perceived that the testing of these ideas in the real world was needed.

The "testing" of the conceptual framework should be carried out before proceeding to refine it into a formal methodology. The best way to conduct this experimentation is to try implementing the framework in a real project. Ideally, various projects should be considered. However, given the relatively long time-frames in which development projects take place, as well as the considerable amount of effort required to carry out this type of experimentation in a real project, the scope of this research was restricted to conduct only one case-study project.

Under a research perspective, the two main aims expected from such a case-study were:

(1) to test the practicability of the underlying process logic and principles of the conceptual framework;

(2) to refine the framework into a more detailed formal methodology.

Since the conceptual framework consisted of a high-level outline, the case-study would therefore comprise a considerable amount of creative development work. In order to achieve these two aims, the approach followed was based on a continuous process of feedback between the experimental implementation of the framework and its readjustment and refinement. This process implied an iterative implementation of the framework, where previous steps could be repeated because corrections would be made, or because more detail would be added to the framework.

Such an iterative approach to a case-study project raises some critical issues. In the first place, it was important that the interference with the real project would not disturb progress, putting critical objectives at risk. Secondly, the practical uses of the SD model should be explicitly acknowledged as tentative, specially in the early stages of the case-study. Finally, it was important that some of the effort allocated to the case-study by the host organisation, would be compensated by actual contributions of the SD model to the project control process.

There were other critical issues that needed to be addressed carefully. Of particular importance was the involvement of the project staff in the case-study. The project was considerably compressed in terms of schedule and it also comprised some level of technological innovation. In addition, new software development tools were also being used. Within this environment, interviews needed to be performed and extensive data needed to be collected. The availability of updated PERT/CPM plans was also of crucial importance. Finally, the tentative applications of the SD model would have to address real management needs, and the recommendations produced by the model would have to be considered carefully.

As it will be described in more detail in the following sections, the overall strategy consisted in implementing and refining the conceptual framework in "parallel" to the existing project management framework, and wait for opportunities of actual application. These opportunities would emerge when particular problems needed to be solved and where the limitations of the traditional models would be explicitly acknowledged by management – some examples of this practical application will be described in chapter 8, after the detailed description of the refined SYDPIM methodology in chapter 7.

The possibility of *intervention* within the organisation gave this research project the shape of *action research* (Eden and Huxham 1996), moving beyond the classic concept of "case-study" within the standard management research methodology (Easterby-Smith et al 1991), according to which intervention does not take place. Action research is characterised by both, the classic academic scientific rigour as well as being an action oriented initiative in a matter that concerns the organisation and where the researcher works with the members of that organisation. Action research is particularly suited to carry out research initiatives aimed at testing, demonstrating or improving the effectiveness or practicality of a certain "technology". This was the primary aim of this case-study: to test a conceptual framework and improve it towards a formal methodology. While the term "case-study" will be hereafter used, it is assumed that an action research initiative is taking place.

Another important issue that became to play a major influence on this research was the need for a SD project model. As a consequence, in addition to testing and refining the conceptual framework, in the course of the case-study the author has also developed a structured method for model development. This method became an integrative part of the SYDPIM methodology, and will be described in the next chapter 6.

5.2 The KDCOM project

5.2.1 Overview

SYDPIM was improved and validated in a large-scale software intensive project, at BAeSEMA Ltd. (now part of British Aerospace Defence Systems, UK). The company was at that time an Anglo-French joint-venture, between British Aerospace (specialised in the development of a variety of defence systems and equipment) and Sema Group (a French company specialised in the development of large IT systems). The "way of life" of this company was to implement this type of large-scale development projects. BAeSEMA was certified with the ISO 9000 family of standards, and therefore there was a well structured project management framework in place. A specialised group for software process improvement (SPI) and metrics collection was also in place, ensuring the required discipline in the implementation of the software development process.

This particular project developed a *Command and Fire Control System* (CFCS) to be installed in a "Destroyer" of the South-Korean navy – this is hereafter referred to as "KDCOM project". The overall project was scheduled to last about 4 years, and comprised the development of several sub-systems, including hardware and software. The prime contractor was BAeSEMA, and there were several sub-contractors from Europe and Korea. BAeSEMA was directly involved in the development of the software component of the Command and Control System (the C2 SW System). This software system was developed through an incremental development process, comprising two major increments each delivering a main system build (SWB1 and SWB2). Some sub-components of this system were being developed from scratch, while others were being re-used from a similar existing system previously developed by the company.

Traditional planning and control techniques were being used (e.g. WBS, OBS, PERT/CPM networks, COCOMO model). Furthermore, the company was committed to exploring new approaches, such as Boehm's spiral model and other software estimating tools. KDCOM therefore provided an excellent case-study to explore the practicability of SYDPIM.

At the very beginning of the project some important risks had been identified by the project planning and productivity improvement teams. Some limitations of the PERT/CPM planning approach were acknowledged, where the use of the SD model appeared appropriate.

In the following sub-sections, a more detailed overview of the key elements of the KDCOM project is presented. This includes a description of the Client, of the product and of sub-contracting. The project management framework adopted in the KDCOM project was of crucial importance for the case-study. On the one hand, SYDPIM requires that some basic processes are in place, and on the other hand it also interferes with this same framework. An overview of the KDCOM project management framework is therefore presented. This is followed by a brief summary description of its main elements: the product development process, the organisational structure (OBS), the work breakdown structure (WBS), and the project plans. Finally, a summary of the relevant risks and critical issues is presented. For the sake of clarity and, for the purpose of this work, these are only summary descriptions. In addition, some elements had to be omitted for reasons of confidentiality.

5.2.2 The Client

For confidentiality reasons not much can be described about the Client. The more important issues were that this was the government of a far-eastern country, and hence cultural differences were likely to have an impact on effective communication. This was particularly relevant because a senior team from the Client was following the project closely, on-site, working in the same building as the project organisation. Effective communication with the Client is a critical project management issue that needs to be addressed carefully (Rodrigues and Williams 1998, Rodrigues 1999).

In addition, this was also a new Client to the company. Satisfying al the contractual agreements was therefore a major priority. Beyond the profit of this individual project, this was also an important strategic move of the company into a new emerging market.

5.2.3 The product

The main output product from this project was a Command and Fire Control System (CFCS), to be delivered and installed in a destroyer. As a secondary product, BAeSEMA also provided offset products related to transferring technology to the Client.

The CFCS was composed of various sub-systems. This included physical equipment and devices (e.g. radar and weapon systems), hardware and software. Three main sub-systems were considered:

- a Command and Control System (C2), composed by a software (C2 SW) and hardware (C2 HW) sub-systems;
- a Fire Control System (FCS), composed by a Dual Director Weapon Control System (DDWCS) and a Medium Range Radar (MW08);
- a Command System Databus (CSDB, hardware).

The development of many of these components was sub-contracted, with BAeSEMA playing the role of prime-contractor. BAeSEMA was developing the C2 system, based on a similar product already developed by the company. The KDCOM project organisation was developing in-house the software component of the C2 System (the C2 SW system). The case-study focused exclusively on this software development sub-project. Part of this software system was a totally new built, whereas other sub-components were being modified from an existing system. Therefore, this implied a "hybrid" type of development.

5.2.4 The sub-contracting

The development of many of the CFCS components was sub-contracted. The primary sub-contractors involved in this process with direct responsibility to BAeSEMA were:

- Ultra Electronics Command & Control Systems (Ultra);
- Racal Radar Defence Systems (Racal);

- Hollandse Signaalapparaten B.V. (Signaal);
- Samsung Electronics Corporation Limited (Samsung).

Figure 5.1 below shows the involvement of BAeSEMA and of these sub-contractors in the development of the various CFCS components.

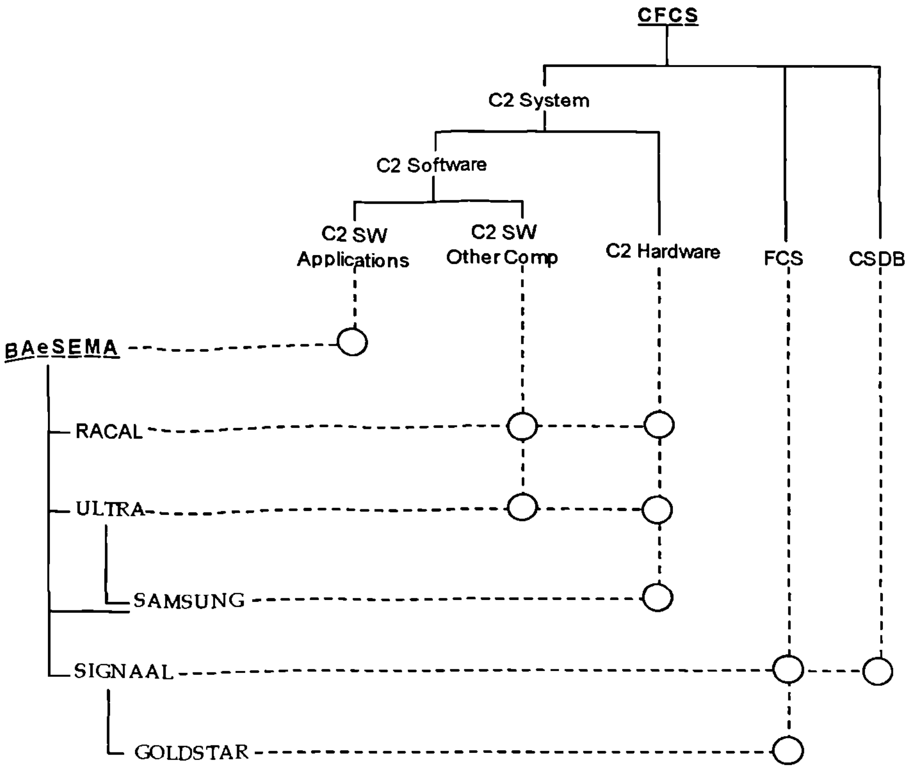


Figure 5.1 – BAeSEMA and sub-contractors involvement in product development

5.2.5 The project management framework

The project planning and control procedures and techniques in use were based on the traditional project management framework. A detailed work breakdown structure (WBS) was developed and continuously updated throughout the project. This was the basis for both work and cost planning and control. An OBS was also developed, specifying the organisational structure of the project team. The OBS was also updated as the project team evolved throughout the project life-cycle. Together with the OBS, the WBS was also the basis to assign responsibilities to the project work. Still based on the WBS, PERT/CPM logical networks were developed and used at different levels of detail. Together with Gantt charts, these networks were the primary scheduling technique and were the basis for critical path scheduling and risk analyses.

Progress monitoring and control was based on *earned value* calculation. This was used together with the other traditional performance indices (e.g. ACWP, BCWP, BCWS, SPI and CPI; Nicholas 1990). The control period for progress report and re-planning was the month. At the lower organisational level, some sub-teams would also report on a weekly basis.

All these traditional techniques and models were implemented in an integrated manner, through the use of *Micro Planner*, a well-known project management software tool.

The in-house software development project was based on the formal specification of a formal life-cycle process, comprised of stages and deliverables. This life-cycle process was updated throughout the project as required, and was followed by all members of the software team. This was an important element underlying the project management framework. In addition, software metrics were being collected as part of this life-cycle process. This included the tracking of defect and rework related metrics, which were stored in a central ORACLE database. This repository of software metrics was shared by the software team.

The product development process

The development of the C2 SW system followed an incremental development process, with two overlapping increments each delivering a main system build. This software process model was widely adopted throughout the company. In each increment, a generic life-cycle was followed. As part of process improvement, the future phases of this life-cycle could be re-structured, based on feedback from the past phases.

A formal document was available in the project information system, specifying this development life-cycle tailored for the C2 SW system. This document was used by the individual teams as the basis to plan and execute their work. This life-cycle specification was continuously reviewed and updated throughout the project, always subjected to approval from first-line managers. This was an extensive document and it is not the purposed of this overview to cover all of its contents in great detail. A summary is here presented.

The structure of the C2 SW development life-cycle reflected some important managerial strategies followed by the organisation. It was also the basis for the set up of the WBS and OBS for this project area.

The C2 SW life-cycle partitioned the development process into a series of stages, steps and tasks. The stages were considered at the management level, and were the basis for project planning and control. The steps were specified at the team leader level, and were used to monitor and control the work of the development teams. Finally, the tasks were specified at the working level, and provided a reference for the software developers to implement the work. For each stage, the life-cycle document provided a description of the objectives, an overview, and a list of the sub-steps within the stage. These steps were described using the ET^3VMX notation (Entry, Task, Tools, Training, Validation, Metrics, and eXit criteria), an extension of the ETVX model, a Quality Assurance technique originally developed by IBM. The main principles underlying the C2 SW development life-cycle were:

- the life-cycle should not describe in detail how the tasks should be accomplished. This was aimed at encouraging ownership and innovation among the various individual teams;
- although the life-cycle stages were described sequentially, they would be implemented in an overlapping fashion, depending on their inter-dependencies. The learning achieved during one stage could therefore feedback useful knowledge to on-going previous stages;
- one stage might have to be implemented more than once, each stage gaining value from the knowledge acquired in later stages or in its previous iterations.

The C2 SW life-cycle was developed based on a systems view of the product (CFCS). This view considered a system logical decomposition mapped to a physical re-composition. This view is shown in figure 5.2. Regarding logical decomposition, the C2 SW system was decomposed into a single software application component (developed by the KDCOM software team), and into other non-software application components (developed by the sub-contractors). The C2

SW application component was further decomposed into "groups", and these further into "processes" (not shown in figure 5.2). The groups were designed, coded and tested separately. Once tested, they were then integrated into physical partial builds of the system. Each build could be composed by one or more groups, and each group could be present in one or more builds. The implementation of the non-application software, which was developed by sub-contractors, also originated C2 SW partial builds. The integration of all the C2 SW partial builds led to the fully integrated C2 SW system build. This software would then be integrated with the other non-application product components, originating the final product (CFCS).

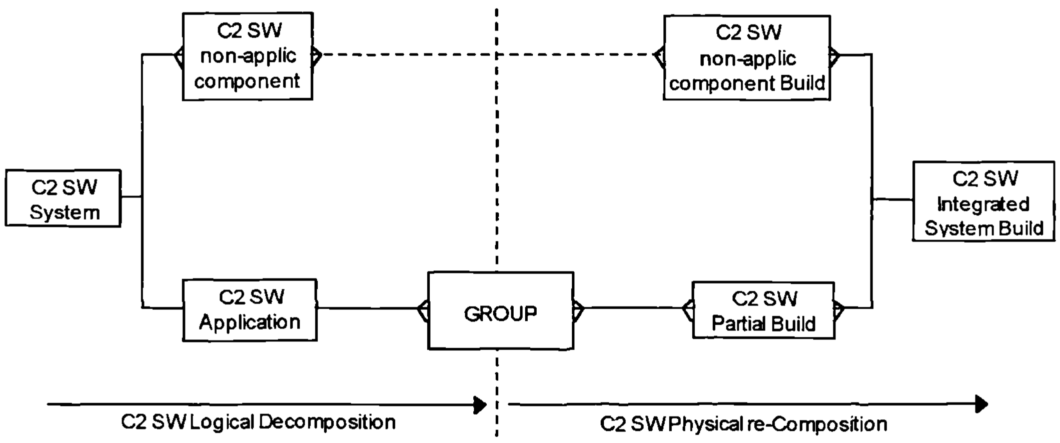

C2 SW System
C2 SW non-applic component
C2 SW Application
GROUP
C2 SW non-applic component Build
C2 SW Partial Build
C2 SW Integrated System Build
C2 SW Logical Decomposition
C2 SW Physical re-Composition

Figure 5.2 – C2 SW system logical decomposition and physical re-composition

The C2 SW development life-cycle was established according to this view of the C2 SW system. Therefore, the sequence of stages comprised the work required to implement both the system logical decomposition and its physical re-composition. These stages are as shown in table 5.1. The system logical decomposition into groups takes place between stages 0 and 3. Stage 0 corresponds to the requirements phase of the classic software development life-cycle. This stage comprises the definition of the CFCS system functional and non-functional requirements, as well as a high level design. This design decomposes and specifies the CFCS into its basic sub-systems.

Stage	Description
0	Requirements Baseline Definition and High Level System Design
1	High Level C2 SW Applications System Design
2	Group Requirements Clarification
3	Group Network Design
5	Unit Code and Verification
6	Group Build and Test
7	Integration Test
8	System Test
9	C2 Software Factory Acceptance Test (FAT)

Table 5.1 – The stages of the C2 SW system development life-cycle

Stages 1 to 3 correspond to the classic software design phase. Stage 1 comprises the decomposition of the C2 system into the C2 SW application component and the non-application components (software and hardware). The C2 SW application is then further decomposed into groups and their requirements are specified. This stage also includes the production of functional tests for the C2 SW application, according to the various functional areas. Stage 2 refers to a more detailed specification of the C2 SW application groups, with special emphasis on the new functional areas to which new code had to be developed (other areas were based on modifications of groups from an existing system). Stage 3 is the final design stage and consists in a detailed design of the groups, which are decomposed into processes (also called "units").

Stage 5 corresponds to the coding and testing of the individual units of each group. In stages 6 these units are integrated to form the application groups, and these groups are then tested. After all groups are built and tested, physical re-composition takes place during stage 7. Here different groups are integrated to form the various C2 SW application partial builds. The functionality of these builds is then tested and the necessary rework is accomplished. At the end of this stage 7, the C2 SW system is fully integrate into a single system build, with all of its functional areas tested. In stage 8, final system testing is performed. At the end of this stage, a clean system build is delivered by the software development team to the systems engineering team, for factory acceptance testing (stage 9).

The KDOCM project organisation (OBS)

BAeSEMA was a large company, permanently involved in project-related activities. The KDCOM project was itself also a large project which included a considerable amount of sub-contracted work. The complexity in terms of technology was also high: the full integration and testing of the system was perceived to be a novel complex task, critical for the project outcome. The KDCOM project organisation was in many ways a *pure project* type of organisation: it was set to be almost self-sufficient, incorporating all the personnel and resources necessary to cover the main project needs. It was fully dedicated to this project, as opposed to a *matrix* type of organisation (Nicholas 1990). An important secondary benefit of this *pure project* type of organisation was to demonstrate to the Client that a well structured organisation was ready to develop the product successfully, being capable of providing quick and effective responses to unexpected risks.

Figure 5.3 below provides a high level representation of the KDCOM OBS, with all the first-line and second-line managers for programme management and software management. Like in any other project, this initial OBS was likely to change over the course of the project, specially at the lower organisational levels.

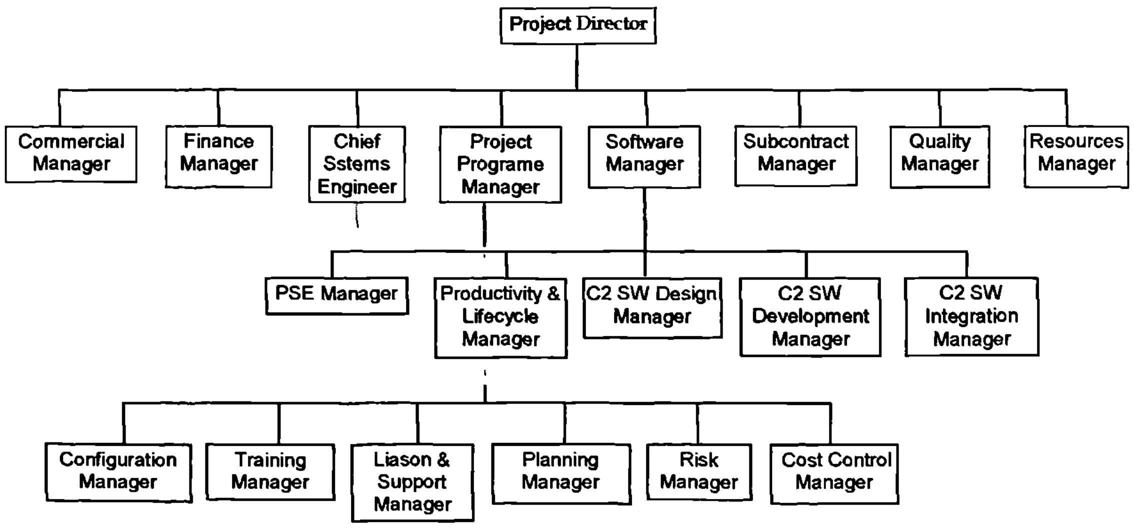


Figure 5.3 – High level representation of the KDCOM OBS

A project director had full authority over the whole project organisation. There was a "programme management" team which covered most of the co-ordination support. This team was focused on planning, cost control and risk management of the whole KDCOM project. There was also a "systems engineering" team responsible for system integration, trials and acceptance. Their aim was to ensure that the system would satisfy the requirements. The other major team was headed by the software manager, and comprised the whole management and development of the C2 SW sub-system. This included the design, development, integration and testing of this software system.

There was also a project management office (the PMO, not shown in figure 5.3 above), whose members belonged to the Client. In the PMO there was a "shadow" member corresponding to nearly every first line-manager of the KDCOM organisation.

As mentioned, this case-study focused on the C2 software development sub-project. Figure 5.4 shows in more detail the OBS of the C2 SW Management and Development area.

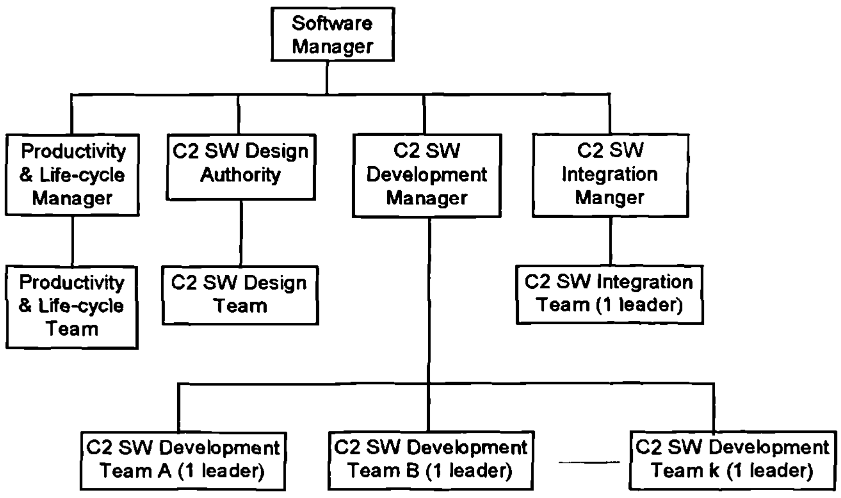

Software Manager
Productivity & Life-cycle Manager
C2 SW Design Authority
C2 SW Development Manager
C2 SW Integration Manger
Productivity & Life-cycle Team
C2 SW Design Team
C2 SW Integration Team (1 leader)
C2 SW Development Team A (1 leader)
C2 SW Development Team B (1 leader)
C2 SW Development Team k (1 leader)

Figure 5.4 – The OBS of C2 SW Management and Development

Below the software manager, there were four major areas of software development led by an individual manager: development of the C2 SW components, integration of the C2 SW components, software design, and the support area of productivity and life-cycle management (responsible for SPI and metrics collection). Below each of these managers there was a team with a single leader. The structure of sub-teams underneath the C2 SW manager reflects the product structure of the C2 SW system, which was composed by a set of individual applications running under a common infra-structure.

The KDCOM WBS

The aim of a work breakdown structure (WBS) is to specify the project scope in terms of elementary tasks (work packages). The whole project is decomposed into a hierarchy of tasks and sub-tasks, down to these work elementary packages. A WBS of a complex project can have various levels of breakdown and it can incorporate hundreds, or even thousands, of elementary tasks. Ideally, the terminal tasks at the bottom level of the WBS will be the basis to developed the PERT/CPM plan in the form of a logical network. The WBS also plays a fundamental role in cost estimation and control. Together with the OBS it is also the basis to assign responsibilities. The WBS is not a rigid structure. As the project scope changes, new tasks are added or removed; when required for control purposes, existing tasks are further decomposed into more detail.

The WBS is at the core of the conventional project management framework. In large complex projects, like the KDCOM project, developing and maintaining a WBS is imperative for an effective implementation of this framework.

The WBS developed for the KDCOM project was complex, comprising several thousands of tasks. This WBS was specified in a comprehensive document, where the tasks were coded and their scope was described. For confidentiality reasons and for the sake of clarity, a simplified overview of part of the KDCOM WBS is here provided, as shown in figure 5.5. The project work was decomposed into the following major areas of activity (or tasks): project management, systems engineering, C2 development, CSDB and FCS development, and system testing and acceptance. The first area refers only to management type of work, whereas the others refer to different types of product development activities. The further breakdown of "project management" is not presented here. This was a functionally-oriented breakdown, comprising: project control, sub-contract management, liaison with he Client, quality assurance, human resource management and training – these represent most of the key functional areas of project management. The "systems engineering" task comprised all the activities related with the development and maintenance of the whole system design. These activities were primarily aimed at ensuring that the product operational requirements were being effectively translated into functional requirements.

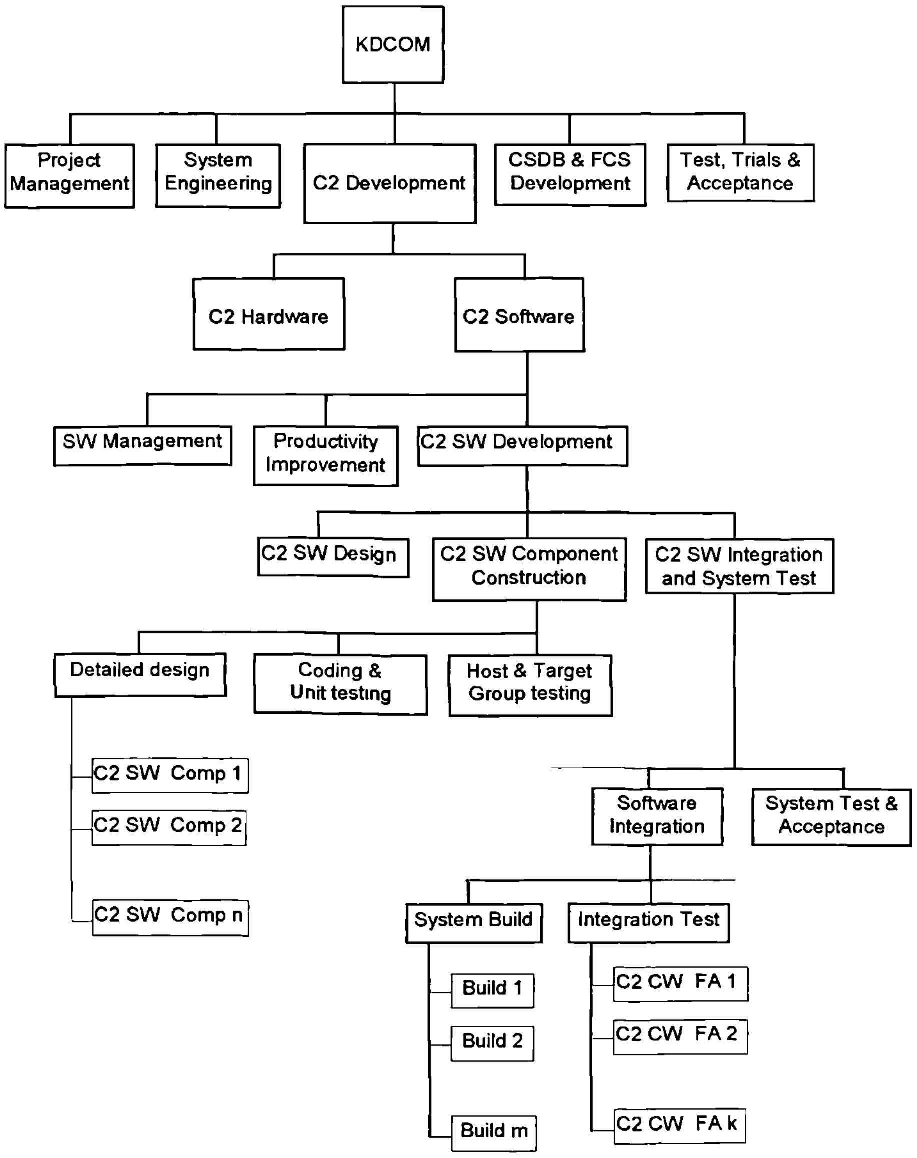


Figure 5.5 – Overview of the KDCOM WBS focused on C2 software development

As previously mentioned, the product components C2 hardware, FCS and CSDB were developed by sub-contractors. Although important project-wide issues were considered in the case-study (e.g. Client changes), the focus was on the development of the C2 software system (i.e. a software development sub-project).

The breakdown of the C2 software development task considers three major work areas: management, productivity improvement, and development. Software management comprised the managerial activities of project control internal to the C2 SW development sub-project, like risk analysis, training, and work scheduling and control. Productivity improvement comprised all the activities related with continuous SPI and metrics collection. This included the update and maintenance of the C2 SW development life-cycle. Finally, the "C2 SW development" work area comprised all the activities that implemented the development life-cycle: system design, component development and system integration and testing. The development of each component comprised the detailed design, coding and unit testing, and host and target group testing (according to the C2 SW life-cycle). The integration and final test of the C2 W system comprised the integration of the individual C2 SW components, and the final test and acceptance of the C2 SW system. The integration of the C2 SW components comprised two types of sub-tasks: system build, which brings together the C2 SW components into progressive builds (progressive integration was being adopted), and integration testing, which consists of integrating these builds according to functional areas (FA), and then test them against the system functional requirements. The full C2 SW system test and acceptance consists in performing system tests until a clean build is ready for final factory acceptance testing (FAT), at the eyes of the Client.

In the actual WBS of the KDCOM project, the terminal tasks show in figure 5.5 above were further decomposed down to elementary tasks, which were used as the basis to develop the PERT/CPM plan.

The project plan

In KDOCM, the planning strategy was to consider the project plan at four levels of increasing detail:

- *Level 1: milestones* – this level captured the major project milestones, providing an overview of the main project areas of activity;

- *Level 2: Client* – at this level the project plan was used as the formal interface to communicate and report progress to the Client;
- *Level 3: Critical Path* – this level considered a breakdown of the project work into to manageable elementary tasks. These were used to develop the various PERT/CPM networks. These networks had specific owners responsible for their construction and maintenance. This level was used to identify the critical path and therefore was the basis for project scheduling and risk analysis;
- *Level 4: teams* –the level 3 work packages were decomposed down into several sub-tasks, which were represented and scheduled using a Gantt Chart. This level of detail was used mainly in the C2 SW development area of KDCOM, as the basis to monitor and control the progress of the different software teams.

The actual critical path based planning was implemented at level 3. Here, several sub-networks were produced and kept updated by the planning team. At all levels the plan was continuously updated and readjusted, according to the traditional project control framework. For reasons of confidentiality, these plans cannot be presented here, including milestones, deliverables and budgets / costs. Some "disguised" planning data will be presented in the practical examples described in chapter 8.

As a summary of this plan, the whole KDCOM project was scheduled to last approximately 4 years. The development of the C2 SW was scheduled to last about 32 months and was planned to be developed in two overlapping builds, SWB1 and SWB2 (incremental development was adopted). SWB1 was scheduled to last approximately 14 months and SWB2 about 30 months. The majority of SWB2 development, in the second increment, would take place towards the end of the first increment.

5.2.6 Project risks and other critical issues

Various risks had been identified at the beginning of the project. Some of these risks were complex and of systemic nature (as most critical project risks in the real world). The limitations of the traditional PERT/CPM planning approach to assess these risks were explicitly acknowledged by management. These were therefore areas of opportunity to actively test the use of SYDPIM, and were therefore explored throughout the case-study.

First, there were human resource management issues regarding the build-up of the software team throughout the project life-cycle. The company had some of their best staff allocated to other projects and hence some aggressive recruiting had to be carried out. Secondly, knowledge transfer was part of the contract with the Client and so inexperienced staff from the Client organisation would have to be introduced in the project at a certain stage. In both cases, communication and training overheads could emerge and lead to quality and productivity problems.

Another risk had to do with Client behaviour, which was perceived of crucial importance for the effective management of the KDCOM project. Of particular relevance was the subjectivity involved in interpreting the several contractual agreements. This subjectivity could be easily exacerbated by cultural differences, often a major obstacle to effective communication. As a consequence, the threat of changes being introduced in the system requirements, particularly during the middle and later stages of the life-cycle, posed a major risk. Estimating and quantifying the final impacts of these changes on the major project milestones, costs, and overall product reliability was recognised by management to be a major difficulty, as were the inadequacies of traditional tools in capturing the secondary downstream effects of these changes.

Finally, the need for high quality designs, together with compressed schedules, was also perceived as a major risk. The integration phase was complex, incorporating some components with new code, others with slightly re-used code, as well as some heavily re-used components. The final delivery of the system to the Client was subjected to rigid milestones and high penalties for delays. If critical architectural problems emerged in the integration phase, it could be too late. It was therefore critical to ensure the required quality of the designs. A possible solution to this problem was to implement better design reviews, where the informal reviews would be replaced by the more thorough Fagan inspections technique (Fagan 1986). The idea was attractive but there were some potential complications: the Fagan technique is highly effort-consuming and with tight time-scales, would the benefits outweigh the costs? If so, how much more time should be given to design? What other adjustments should be made to the plan? It was acknowledged that the PERT/CPM networks in use would not support the required analysis effectively, in order to answer these critical questions. Among various limitations and difficulties, there were critical indirect impacts of systemic nature which could not be captured in PERT/CPM model.

Throughout the case-study, SYDPIM was actively used to assess these three risks. These practical applications will be described in chapter 8.

5.3 Implementation of the case-study

5.3.1 General strategy

The strategy adopted for this action case-study was to use the first increment of the of the C2 SW project as the basis to develop, validate, and calibrate a SD project model. In this increment, the use of this model would focus on *post mortem* analyses. This would allow for a more progressive introduction of the model in the organisation, without "challenging" or "criticising" the existing plans. Once confidence in the model was gained, it could then be used more pro-actively in the second increment, supporting on-going progress monitoring and re-planning. The case-study would only cover the early stages of this second increment.

Model validation during the first increment focused on both structural representation of the software development process being followed, and on accurate replication of observed behaviour. Within this increment, various sub-components of the C2 SW system were being developed in parallel. Where appropriate, and depending on the opportunities, the SD model would also be calibrated to replicate the development of these individual components. Additionally, informal interviews with managers and staff would provide the required feedback to improve confidence in the model.

One important aspect of model development was the level of complexity to be considered. In the first place, the aim of the case-study was not to develop a sophisticated and comprehensive SD model, specialised in the KDCOM project (although that could be a desirable outcome). Developing such a comprehensive model requires a considerable amount of time and effort (e.g. the models reported in Cooper 1980, and Williams et al 1995, required a fully dedicated team for the period of about 1 year). Furthermore, in order to test and improve the conceptual framework, a simple model would be sufficient. On the other hand, the company had shown interest in keeping the SD model after the case-study, in order to use it in the future, according to the SYDPIM framework. This required that a member of the planning team would get involved in the modelling process. In the end of the case-study, the final model would be handed over to this team member. Therefore, since the company was not familiarised with the SD modelling technique, a simple model was more appropriate for a gradual familiarisation.

The overall strategy for model development was therefore to develop a prototype single-phase model, as a generic structure for development tasks. This model could be easily tailored to simulate various parts of the development life-cycle, and at different levels of aggregation (e.g. design phase of specific component versus full life-cycle of a main system build). This prototype model would then be improved as appropriate. This would include incorporating more detail and even use it as a building-block to develop a more complex multi-phase model. Throughout the case-study, the model would therefore evolve in various versions.

In order to test and refine the conceptual framework, two ways of using the model were considered: in "parallel" to the existing control framework, therefore without interfering with decision-making; and by actively supporting this control framework, being used as the basis to take actual decisions. The latter case would be less frequent as it would require appropriate opportunities to emerge. Nevertheless, both situations would allow for the testing and refinement of the conceptual framework because the SD model would have to be used in an integrated manner with the PERT/CPM model. This way, the structure of both models would have to be "mapped" one another, and data would also have to be exchanged between them.

Testing and refining the conceptual framework without interfering with the course of the project, and further using the SD model actively when appropriate, was an important overall strategy for three main reasons: first, it was a condition imposed by the company that the case-study should not disrupt the course of the KDCOM project. Therefore, control decisions should not depend upon, and thereby wait, for the SD analyses. Secondly, since the model itself and the conceptual framework were being tested, this should be a tentative application hence not aimed at providing reliable results in the early stages. Finally, it was important for research purposes to be possible to step back from the highly time-pressured context of the project. This would provide the required time to analyse results, review and rework ideas, and conceptualise new SYDPIM elements. For example, the analytical links between the two modes were not available in the beginning of the case-study and needed to be developed throughout the project. This reinforces the "action research" nature of this case-study: a theory was supposed to emerge from the practical work which was being conducted with scientific rigour, and where intervention within the organisation would take place (Eden and Huxham 1996)

5.3.2 Revision of objectives

The research aims of the case-study were to test the validity of the conceptual framework and refine it into a more detailed formal methodology. The achievement of these aims required an appropriate work plan, adjusted to the reality of the KDCOM project. The first step was to developed a clear statement of objectives regarding practical implementation. This was based on a high-level analysis of the project and on early discussions with managers.

The very first issue raised was the availability of a SD project model. There were two main alternative approaches: to develop a brand new model, or re-use an existing model available in the literature. The second route had the benefit of requiring less time and effort. However, it had the disadvantage of presenting both the modelling process and the model itself to management in a "black-box" fashion. Whatever the route followed, it was important that management would develop a sense of ownership, confidence and understanding about the model. It was perceived that in a time-pressured environment, management was much more willing to provide input to the development of a new model, rather than studying an existing model. Furthermore, conceptually speaking, a new model was more likely to become a true requisite model (Phillips 1982), thereby representing "more validly" the specific issues of the project. It was therefore decided that a new model would be developed, eventually re-using existing generic structures and data available in the literature (e.g. Abdel-Hamid and Madnick 1990). In order to overcome the problem of the time and effort required for this modelling process, it was decided that a simple single-phase prototype model would be the starting point to apply the conceptual framework. This initial model version would be improved as needed for the refinement of SYDPIM.

The next issue was to decide about *how* to develop the model. Again, there were two alternative routes: implement a personal approach in a "black-box" fashion, based on the author's personal experience; or establish and follow a structured modelling process, visible to management. It was decided that the second route would be followed. There was a very important reason for this decision: like BAeSEMA, any other company willing to implement SYDPIM will need a SD project model. Since most companies will not have such model readily available, they will need to develop a new one. It was therefore clear that providing an accessible means for companies to carry out this modelling work was a critical factor for SYDPIM practicality. As a consequence, one of the new objectives proposed for the case-study was to develop an initial framework for model development. This framework would be used as the basis to develop the initial prototype model. It would also be improved and refined throughout the case-study, and it would become an integrative part of the SYDPIM final methodology.

The second objective was to develop an initial prototype single-phase model, capable of reproducing well the past behaviour observed during the first increment (SWB1). The model would be progressively calibrated to reproduce life-cycle stages, as soon as these were completed. Depending on data availability, this could focus on individual components or in the whole build. Although there was a metrics plan being implemented in KDCOM, this was not specialised to support the use of the SD model. Extra metrics would have to be collected, and not all teams would perform this task at the same time and in a timely fashion. The model structure and the data used for calibrations would be reviewed by management and development staff.

The next objective was to use the model according to the conceptual framework. During the first increment, this would focus on *post mortem* retrospective analyses, in parallel to the existing control framework. Whenever analyses requests would be explicitly issued by management, the SD model would be used actively to support decision-making – the practical applications reported in chapter 8 are examples of this situation.

The use of the model was aimed at testing and refining the conceptual framework. In practical terms this comprised two main objectives: the application of the model according to the process logic of the framework (e.g. roles of the model, stages of the management process where it should be used), thereby readjusting and refining this framework into more detail; and the conceptualisation and formal specification of analytical links between the two models. These links would consider both structural relationships and data exchange. The initial conceptual framework did not specify how these links were to be implemented. Throughout the case-study, these links would be progressively conceptualised, tested and specified formally.

The case-study was planned to cover SWB1, and after that, a few months of SWB2. The final objective was to hand-over the final project model to the company so that SYDPIM could be implemented pro-actively for the remaining of SWB2. This would imply the development of the required expertise within the company. A member of the planning team would follow the case-study closely for this purpose.

In summary, the objectives developed for the case-study regarding practical implementation were as follows:

(1) to establish an initial structured framework for model development. Throughout the case-study, try to refine this framework into a formal model which would become an integrative part of the SYDPIM methodology;
(2) develop an initial SD project model, capable of reproducing the past behaviour of the first increment (SWB1). Throughout the case-study, refine this model into further detail, as required by the continuous testing and improvement of SYDPIM;
(3) use the SD model throughout the first increment, according to SYDPIM, and in parallel to the existing project control framework. This would consist in *post mortem* retrospective analyses of past life-cycle stages. When requested by management, use the SD model to support the development of re-planning solutions to actual problems;
(4) as the SD model is used, revise and refine the conceptual framework into more detail. In particular, conceptualise and test analytical links between the two

models. In using the model, implement these links to exchange monitoring data and work plans between the models;

(5) develop expertise within the company so that the SD model can be handed-over to the planning team, for pro-active use throughout the second increment (SWB2).

At the beginning of the case-study, this statement of objectives added an important element to the scope of this research: to develop a structured formal method for model development.

The next step was to develop a high-level work plan to achieve these objectives. However, before that, it was important to analyse the underlying requirements and identify the critical issues to face ahead in the course this case-study.

5.3.3 Requirements and critical issues

There was a number of important requirements underlying the achievement of the proposed objectives. There were also some conditions imposed by the company, which could threat these requirements. Hence these conditions constituted critical issues for the case-study.

The main requirements of the case-study had to do with the existing project management framework and with the company support to the case-study. This included:

(1) free access to all information and data available in the project information system. It is important to note that the KDCOM was a project of military nature and hence most of this information was classified (at least as confidential);
(2) availability of data and metrics about the project. This implied the collection of extra metrics, not considered in the project control process in place;
(3) time and effort availability from management and development staff. This included metrics collection, model revisions, informal interviews and presentations of results;
(4) availability of updated PERT/CPM plans in a timely manner;
(5) opportunities to apply the SD model pro-actively.

On the other hand, the major conditions imposed by the company were:

(a) to cause no disruption to the normal course of the project;

(b) the project control process (i.e. monitoring and re-planning) should not depend on information available from the SD model, unless explicitly required by management;

(c) in conditions of delays and schedule pressure, full priority would be given to the KDCOM project work;

(d) while efforts would be made, extra metrics would only be collected by the development teams if there was extra time available.

From the combination of these requirements and conditions, the following critical issues were identified:

(i) it was important to identify and focus on those opportunities where the SD model would clearly provide a distinctive contribution to the management process;

(ii) when needed data was not available, alternative solutions would have to be devised. This included deriving results from interviews, extrapolation from existing data or from any other type of informal or empirical evidence. Also, apply the SD model to those system components where data was made available. In extreme cases, and if possible, collect own metrics;

(iii) it was critical to keep a focus on refining the conceptual framework and on establishing analytical links between the two models. The outcome of this research would depend on these developments. At certain stages, this would imply "stepping back" from the project, revise the state of the framework, and if necessary repeat the implementation of previous stages.

These critical issues were the basis to guide the actual implementation of the case-study.

5.3.4 Initial work plan and implementation

Following the initial high-level analysis of the KDCOM project, and the statement of objectives, an initial work plan was developed. It is not the purpose of this chapter to present a detailed description of this work plan, nor a chronological account of its implementation. In summary, the initial work plan comprised the following main stages:

(0) *detailed analysis of the C2 software development project* – this focused on the product, development process, organisation (OBS), WBS, sub-contractors and potential risks;

(1) *development of initial SD project model* – this included the following steps:

- (1.1) development of a structured framework for the modelling process;
- (1.2) development of a generic prototype model. This included validation reviews with managers and staff. The model development framework was refined as appropriate;
- (1.3) data collection to calibrate the model (input data) and to specify the past behaviour to be reproduced by the model (output data);
- (1.4) calibration of the model to reproduce this past behaviour.

(2) use of the SD model throughout the first increment, to support the continuous refinement of both conceptual integrated framework and model development framework. This required the continuous collection of the necessary metrics and data;

(3) hand-over of final model to the planning team.

The overall case-study lasted for about 18 months, working on-site at the KDCOM project, although there were some interruptions. An equivalent of about 15 full-time months were dedicated to the case-study.

Most of the effort was dedicated to stage (2), which was critical for the aims of this research project. During this stage, the SD model was calibrated for past behaviours, according to the results recorded in the PERT/CPM model. This calibration required data to be transferred from the PERT/CPM model to the SD model, on the basis of conceptualised analytical links. Based on the diagnosis of this past, the SD model would often provide a different view of the future than the PERT/COPM model. An improved readjusted plan was then developed in the SD model. The next step was to transfer this new plan back to the PERT/CPM model, thereby producing a readjusted logical network. Again, this would be based on the use of the analytical links. Not always the readjusted plan was actually updated in the KDCOM PERT/CPM networks – as already mentioned, this would depend on management explicit request. The important aspect of this exercise was that analytical links would have to be hypothesised, tested and rehearsed several times. Equally, the process logic underlying the conceptual framework was also rehearsed several times.

Stage (1) also consumed a considerable amount of effort. A generic prototype model was developed and calibrated to past life-cycle stages of various system components during the first increment. While the model was developed, the model development framework was also refined. This initial version of the model also evolved throughout the whole case-study.

5.3.5 Results achieved

Overall, the case-study was successfully implemented. The expected aims were achieved. The conceptual integrated framework was refined into sufficient detail, and a structured model development method was developed. Both of these elements were drafted by the end of the case-study, and were subjected to further re-structuring. Opportunities to actively apply the SD model emerged and were implemented according to the conceptual framework. At the end of the case-study, the SD model was handed-over to a member of the planning team for post-case-study usage.

However, the results achieved were not restricted to the actual developments carried out on-site, during the case-study. This practical experience has also proven of immense value for providing important insights, which were used as the basis for the author to carry out further conceptual developments, and to further refine and re-structure the overall SYDPIM methodology.

The final SYDPIM Model Development Method is presented in the following chapter 6. The SYDPIM Project Management Method is presented in chapter 7. Some of the practical applications of SYDPIM throughout the case-study are described in chapter 8.

5.4 Conclusions and discussion

Various conclusions can be derived from this case-study regarding the practical implementation of SYDPIM as a generic project management framework.

The main conclusion from this case-study is that *a SD project model can be actively used to support the control process an on-going project, embedded within the conventional project management framework, and formally integrated within a PERT/CPM model.* This conclusion is supported by the successful use of a SD project model in this way and according to SYDPIM, throughout a major phase of a real on-going project. This conclusion has three main implications: first, SYDPIM does not require a radical change to the existing project management framework in an organisation. This is because the changes required build upon the conventional framework. Secondly, it is possible to implement analytical links between a SD project model and a PERT/CPM model (the "theory" of these links will be presented in chapter 7). And finally, a SD model has a potential to provide reliable analyses and estimates, not only about the past of a project (i.e. *post mortem* analysis; e.g. Cooper 1980, Williams et al 1995), but also regarding the future outcome of a project (e.g. as reported in Pugh-Roberts Associates 1993).

Within the conventional framework, the SD model can perform distinctive roles. In particular, these include: assessing the systemic impacts of high-level complex risks (e.g. Client changes), uncovering information about the project status, diagnosis of project past, assessing the impacts of alternative control policies, and accelerating process improvement through repetitive "what-if" experimentation. The SD model can also support the traditional models by enhancing their roles (e.g. re-planning of the project future).

The establishment of analytical links between the two models allows for a closer and more rigorous integration of the SD and the PERT/CPM models, while taking this integration down to the quantitative level. This increase in rigour enhances the added value of the SD model as well as the validity of its recommendations. For example, through these links a project plan improved in the SD model can be transferred back to the PERT/CPM model. Otherwise, the translation of the SD recommendations would most likely be converted with "errors" to the PERT/CPM model, and would not be further "tested" at the operational level in this model.

The reliability of the SD analyses and estimates regarding the project future can be considered in an absolute or relative manner. The first situation is based on the assumption that the structure and the data in the SD model corresponds to specific elements observed in the project reality. In most cases, however, a full confidence will not be possible to achieve. Under a relative perspective, the SD model is first calibrated to reproduce a project plan. Alternative plans and risk scenarios are then tested based on this calibration. The outcomes are then compared as deviations to this initial scenario (i.e. calibration for the plan). While some practitioners claim having achieved full reliability (e.g. Pugh-Roberts Associates 1993, Cooper 1996), "relative reliability" reinforces the model's appropriateness to analyse the project future. This was observed in the present case-study. In both cases, model validation should be the focus of careful attention.

Another important conclusion is that in order to be implemented effectively, SYDPIM imposes some important requirements. First, an appropriate SD project model must be available. The development of this model can follow a generic structured process. The SYDPIM method developed throughout the case-study has proven effective in covering this need. SD expertise is also required to develop, maintain and use this model. Secondly, the use of the model, as described in SYDPIM, requires a considerable level of data availability. One obvious alternative to overcome the possible lack of data is through expert information gathering from management and staff. However, it was observed in this case-study that subjective opinions often differ considerably, and are dominated by biases of political nature. Specialised structured techniques can be used to help with this (Wright 1985). The maintenance and use of the SD model implies continuous re-calibrations. Overall this proves to be an effort-consuming task. This extra effort must therefore be explicitly acknowledged in the project budget. Overall, it was also clear that SYDPIM is more suited for structured management environments: it assumes that SD is applied within a well structured project management environment, wherein traditional operational techniques and tools are in use (a well defined WBS specifying the work scope, an OBS identifying the organisational elements, and a formal definition and maintenance of the life-cycle development process). In such mature environments, metrics collection and process improvement activities are an integrative part of the project management process. It can be argued that, more often than not, these "pre-conditions" are not met within many project organisations. Nevertheless, SYDPIM is a flexible framework and can be implemented at different levels of formality.

Finally, it is important to stress that the refinement of the SYDPIM framework into a formal methodology, and the conclusions here presented, are restricted to experimentation in one project only. Carrying out this case-study in a different project could have led to different or additional conclusions (this is particularly true if the project management framework was less structured). Ideally, many other case-studies would have been conducted and even repeated. However, in the domain of social sciences, such a positivist approach is not feasible and this particular research is no exception. This is here acknowledged explicitly as a limitation to this research. Nevertheless, it is a limitation that stems from practical restrictions to scientific experimentation in complex social systems. Eden and Huxham (1996) argue that this type of research falls into the "action research" methodology. It typically consists of "one-off" initiatives, hence not benefiting from "repeatability". Nevertheless, it can still be conducted in a rigorous scientific manner, and thereby produce valid theories as an output. The process through which SYDPIM was validated in the KDCOM case-study can be replicated in another future project, and can be explained to others – this is an essential requisite of valid action research (Eden and Huxham 1996).

6. The SYDPIM Model Development Method

6.1 Introduction

The next step within this research work was to test and refine the conceptual integrated framework in a real project. A case-study was carried out for this purpose. A SD project model was needed and therefore the first critical issue was to decide about how to make this model available. This issue has raised the critical question of how an organisation intending to implement the SYDPIM methodology would solve this problem. Most organisations in the market do not have a SD project model available for their projects, nor they are likely to have extensive expertise on SD project modelling. Furthermore, a review of past developments of SD project models revealed that there was no structured method of model development available in the literature. In these past developments personal approaches had been followed. With no SD project model available, and with no method also available to develop this model, organisations would find it difficult to implement SYDPIM. In addition, the SYDPIM methodology also requires a considerable level of validity from the SD project model.

A good quality model is required to implement the SYDPIM methodology. To the author's experience, the quality of the model is highly dependent on the quality of the development process adopted, especially if experienced modellers are not available. For these reasons, as part of this research the author has developed a structured method for the development process of a SD project model. This method is an integrative part of the SYDPIM methodology and was also refined and tested throughout the case-study. This chapter describes the SYDPIM Model Development Method.

The SYDPIM method is based on a review of the approaches followed in the past SD applications to project management (see chapter 2). While there are some interesting models available in the literature, the approaches followed to the model development process are personal and unstructured. They are primarily based on the high-level principles of SD modelling and are not specialised to the project management field.

The SYDPIM Model Development Method proposes a structure process comprised of main phases, stages and design elements. The method gives particular attention to model validation, proposing specialised validation phase and a continuous validation activity throughout the development process.

It should be stressed that this method is of crucial importance regarding the practical feasibility of SYDPIM. On the one hand, an organisation willing to implement the methodology needs a project model tailored for their specific projects. On the other hand, the validity of the SD model to be used affects the whole benefits of the methodology, as actual decisions will be based on the results produced by this model.

6.2 Structure of this chapter

The next section 6.3 provides an overview discussion regarding the conditions that motivated the SYDPIM Model Development Method. There were various possible alternatives to have a SD project model available to carry out the case-study of this research. Considering the implications for the validity of the case-study, as well as the practicability of the SYDPIM methodology, the development and implementation of a structured method was perceived to be required.

The following section 6.4 provides a rationale in favour of a structured development method. Modelling is about representing a perceived reality in a structured and simplified manner. Imposing a structured method on a modelling process must be addressed carefully, so that the required creativity and personal views of the users are not overlooked. On the other hand, introducing discipline into a modelling process, taking advantage of past experiences, is useful and supports both model quality and validity. It is here argued that a structured process can provide this discipline, without necessarily restraining the essential human ingredients.

Section 6.5 provides an exhaustive review of some of the most relevant past developments of SD project models. The aim of this review is to identify and extract those elements of the development processes followed which have proven useful in practice, so that they are incorporated in the SYDPIM method. The development processes adopted in these past developments were based on personal approaches and were not made explicit by their authors. This review tries to make these processes explicit so that they can be analysed.

Section 6.6 provides a detailed description of the SYDPIM Model Development Method, proposed in this research. First, a rationale for a life-cycle approach to a structured process is provided in sub-section 6.6.1. This type of approach provides various benefits and has proven useful in many other fields, like software development. While structured in nature, it can be implemented in a flexible and iterative manner. It can respond to short time-scales and unstable requirements through prototyping and incremental development.

In sub-section 6.6.2, an overview of the SYDPIM development life-cycle is described, and in sub-sections 6.6.3 and 6.6.4, the two phases of model design and model implementation and validation are described in detail. In the sub-section 6.6.3, the three stages of the design phase are described separately. This includes the specification and exemplification of the formal design elements proposed in the SYDPIM method. In sub-section 6.6.4, some initial considerations and assumptions about the simulation language used to implement the model are discussed. The basic principles of the SYDPIM model implementation and validation phase are then presented, and an overview of the life-cycle of this phase is provided. In the SYDPIM method, model validation is integrated into the implementation process. An integrated validation framework is proposed, based on a review of existing work in this critical area. The actual implementation of the model, based on the formal design elements, depends on many specific aspects of the modelling situation. The SYDPIM implementation and validation stages are therefore illustrated through a simple generic example.

Finally, section 6.7 provides an overview discussion of the SYDPIM Model Development Method, focusing on the critical aspects for a successful implementation.

6.3 Overview discussion

There were two critical issues that motivated the development of the SYDPIM Model Development Method: the need of a project model for the case-study, and the practical feasibility of SYDPIM.

There were two requirements for the project model to be used in the case study: it needed to be made available quickly and it had to gain the acceptance from senior managers. Time was a critical factor because System Dynamics was being introduced in a large-scale time-pressured project, which was taking place within an organisation with limited knowledge about the methodology. In order to gain the required management commitment to support the case-study, a "visible" model need to be available. The project could not wait a long time for SYDPIM to be tested and play a role.

The SYDPIM project management framework requires that a SD project model is available. The practical feasibility of SYDPIM therefore depends on an organisation's ability to develop this model.

Regarding the case-study, there were three main possible solutions: (1) developing a new model, (2) using an existing model available in the literature, or (3) a compromise between these two, where parts of existing models would be re-used as appropriate. The first possibility had the practical disadvantage of requiring a considerable amount of effort and some involvement of senior managers. The second possibility had the critical conceptual disadvantage that the model would not incorporate the personal views of the decision-makers of this specific project – it would have to be assumed that the structure of a past model could be transferred for this specific project, only with some minor changes. This would counter the basic principle of System Dynamics, where the process of model development should involve decision-makers, so that the model reflects their personal views about the project reality. To the author's opinion, this was an essential requirement, which can be found in the wider domain of decision modelling, as reflected in the concept of *requisite model* introduced by Philips (1982): the model development process must work as a framework for the iterative development of a coherent representation of reality, and hence it is necessary to involve all those who are in some way responsible for decision-making. In order to balance the effort required, the time available, and the need for a requisite model, the author decided for the third route. A previous review of past models (Rodrigues 1994a, Rodrigues and Bowers 1996a) was the basis for the development of a new model. Where available, some existing sub-structures were be re-used.

The next decision was to decide about *how* to develop the model. Is there a structured process available? In chapter 2, the implementation process of the System Dynamics methodology was described as comprising a set of sequential and iterative steps. One of these steps is the development of a quantitative simulation model. The reality of SD modelling is that there is no commonly agreed formal structure for this process. As also stressed in chapter 2, different researchers and practitioners advocate different views. For example, there has been much discussion about the relationship between qualitative influence diagrams (IDs) and the "level/rate" diagrams, where the latter is used in most software packages as the basis for quantitative simulation models. In this research, the author has assumed the perspective of using qualitative IDs as preceding the development of level/rate diagrams, which are thereby considered as part of the simulation model itself. This implies that the model development process should start with a high level causal analysis using IDs. However, more is needed to establish a structured process for the model development process. Unfortunately, going back to the SD literature reviewed in chapter 2, the past applications to project management do not go beyond high-level descriptions of the generic SD process. Personal approaches to the development process are reported, where qualitative influence diagramming is generally used priori to quantitative simulation modelling (e.g. Roberts 1978, Richardson and Pugh 1981, Wolstenholme 1990, Morecroft and Sterman 1994, Eden 1994, Coyle 1996). A structured process for the model development process was not available.

Regarding the practical feasibility of SYDPIM, the availability of this process is an important requirement so that an organisation can implement SYDPIM. The reality is that the vast majority of the potential practical users of SYDPIM do not have a SD project model available, and in many cases have little or no SD expertise. Therefore, while SYDPIM focus in the *use* of a SD project model, it is also crucial that it delivers a solution for this problem. Three possible approaches can be considered: deliver a complete "generic project model", deliver a base-model which can be tailored for specific projects, or deliver a generic process to develop a new model. To the author's opinion, a totally generic model can hardly be a valid model as it will overlook specific issues of a project. The second solution is feasible but it requires the availability (or development) of a robust base-model capable of representing any project through some level of tailoring. This would have to be a proven model successfully applied to various projects. As it will be seen in the next section, there are only two models available in the literature which could be considered for this purpose (Abdel-Hamid and Madnick 1991; Ford 1995). However, these two models have only been applied to one project each and therefore they cannot be considered as generic and practically well-proven – for example, the comprehensive model presented in Ford (1995) has later undergone structural improvements (Ford and Sterman 1998). Restricting the feasibility of SYDPIM to the validity of an existing model was not considered by the author as the appropriate approach.

The solution chosen was to incorporate in SYDPIM a structured method for the model development process. In this way, any organisation willing to implement SYDPIM has a means to develop a new model (or readjust an existing one) for their specific projects. The author therefore developed a generic structured method for the development process of a SD project model. This method, called "SYDPIM Model Development Method", is an integral part of the SYDPIM methodology. The importance of this method within SYDPIM is emphasised by the fact that the validity of the SD model used affects the whole usefulness of the approach. Much of the decision-making within the project will be based on the results produced by the SD project model. For this reason, the SYDPIM method gives particular emphasis to model validation.

Another important issue of the method is the type of projects to which it can be applied to. Most SD project models developed so far are applied to development type of projects. Development projects generally comprise a phase of system design followed by implementation. The concept of "project" within this research was defined earlier as "...a complex and unique undertaking aimed at the design, realization and delivery of a tangible product." The method here proposed is therefore specialised to this type of projects. However, as it will be seen, the essential principles underlying the method are generic and can be adjusted to other type of projects.

The SYDPIM Model Development Method was developed prior and during the case-study. Following the outline of an initial high-level framework, this was further refined and improved throughout the case-study.

Finally, it is important to clarify that it is not the purpose of the proposed method to provide a generic SD project model, nor a set of generic SD sub-structures from which a model can be assembled. The focus of the method is exclusively on the *process* of developing a SD project model. Furthermore, it was not the purpose of the case-study, nor of this research, to develop a comprehensive project model. The implementation of the method here proposed does not imply the development of a complex model. The method can be used to develop from simple single-phase models to complex multi-phase models. The level of complexity chosen should be according to the managerial needs of the project manager. For the purpose of the case-study, a simple single-phase model was first developed. This provided two main benefits: a short development time, and a simpler testing of the conceptual integrated framework. This allowed the testing and refinement process to be progressive, with multi-phase issues being considered later in the case-study.

6.4 A conceptual rationale

Is there a structured method for developing SD models in general? If so, what is the relationship with the method here proposed? Considering that modelling is relative process of representing a perceived reality, creativity plays an essential role. To what extent can therefore a formal structured process be imposed? Can the resultant rigidity be a barrier to creativity?

Since System Dynamics was first introduced, considerable attention has been given to the definition a well defined and structured process of implementing the methodology: if a modeller wants to develop a SD model, here are the steps to be followed (e.g. Richardson and Pugh 1981, Wolstenholme 1990, Coyle 1996). However, there has been difficulties in achieving a commonly agreed process among both researchers and practitioners. In part, this stems from the fact that model conceptualisation in simulation modelling is inherently a creative process. The purpose of modelling is to represent a perceived reality of a system in a simplified manner. Therefore, a model is relative to the modeller's perceptions about reality, to the modeller's needs that the model will address, and even to the modeller's "modelling style". Furthermore, in large-scale and complex social systems, it is generally accepted that there is no full understanding about the system, prior to the model being developed. During the modelling exercise the modeller will learn continuously about the various structural and behavioural aspects of the system. This learning process requires creativity to devise various modelling alternatives. Learning is often considered as the main benefit of the modelling exercise. Because creativity is so important in this learning process, imposing a formal structured set of rules is difficult. And it must be approached with care, not to become an obstacle to the learning process.

However, despite the important role of creativity, any experienced modeller has found him/herself repeating steps when developing a model. It is generally acknowledged that when a process is repeated several times, performance can be improved through the explicit definition of the process and through the development of a set of rules to guide that process. A good example is the software development field: while in the early days successful products would depend mostly on the unstructured *art* of the programmers, repeated past experiences have motivated the definition of well structured frameworks for the software development process , called "process models" (e.g. classical life-cycle model; Boehm 1981). The ideal of a process model is to ensure that a software product can be developed within controlled objectives of time, cost, and quality, regardless of the specific human resources employed. While not always these objectives are under control, these process models have proven beneficial to the field. In this same line of evolution, it can be argued that a similar "step forward" should be attempted in the field of simulation modelling. As this need is recognised, there have been attempts to develop a unified theory of modelling and simulation (Zeigler 1976, Cellier 1982, Neelamkavil 1987). However, while the software development field may provide a good precedent, care must be taken in transferring the structured principles to the field of simulation modelling: creativity must be given sufficient scope.

As already mentioned, a model is relative to various aspects of the modeller: personal views, needs, modelling style, and others. Therefore, different SD models can be developed for the same system. However, there is one particular aspect of any model that must be ensured: its "quality" (i.e. how accurately does it represent the real system and how effectively does it address the modeller's needs). The main argument in favour of a structured development process is that it is generally recognised that the "quality" of product depends in great part on the "quality" of the development process adopted. If a well defined structured process is followed, uncertainty is reduced and a successful outcome is more likely.

For the purpose of implementing SYDPIM, a "quality" SD project model is required. It is argued in this research that for the purpose of developing such model, some structuring of the model development process is needed. The model development method proposed in this research provides a structured process aimed at ensuring that a SD project model can be developed with this required "quality", while conforming with some basic SYDPIM requirements.

Finally, it is important to note that the model development method here proposed must not be considered as the only way to develop the "correct" model. The concept of "correct" model is not even applicable to SD modelling. There is not a unique model for a unique project. Findings about generic structures (Lane and Smart 1996, Pugh-Roberts Associates 1993) suggest that while some core model structures can be applied to more than one project, some tailoring is always needed. On the other hand, different modellers would probably develop different models to serve the same purpose in a same project. The model development process here proposed is formal and intended to be generic. However it should be implemented in a flexible manner to address the specific issues of the project.

6.5 Review of past developments

6.5.1 Overview

This section provides a brief overview of the more important approaches to model development undertaken to present in the field of project management, as identified by the author in the current literature. This overview is not intended to focus on the detailed contents of the various models developed, but rather on the underlying *processes* followed to model conceptualisation, development, and validation. These are discussed in chronological order as follows:

- Roberts (1964) – the dynamics of R&D;
- Cooper (1980) – modelling of a large shipbuilding program;
- Richardson and Pugh (1981) – the dynamics of R&D;
- Abdel-Hamid and Madnick (1991) – the dynamics of software development;
- Lin (1993), Lin and Levary (1989) – the dynamics of software development;
- Williams et al (1995), Ackerman et al (1997) – the dynamics of a large design and construction project;
- Ford (1995), Ford and Sterman (1998) – the dynamics of product development projects;
- Pugh-Roberts Associates (1993) – the PMMS tool for modelling development programs (in use up to present).

The relevant issues analysed in each development are:

- *model development* – overall process, sources of information, and steps followed;
- *model structure* – breakdown of the project into sub-tasks (i.e. project decomposition), entities and their flow life-cycles;
- *model quantification* – sources of information, estimating methods adopted;
- *model validation* – overall approach and procedures adopted;
- *model use* – the way in which the model was used to fit its purpose within the specific project.

It should be noted that this analysis is strictly restricted to the information available in the literature.

6.5.2 Roberts (1964, 1978) – a simple model of R&D projects

The work developed by Roberts was first presented in doctoral dissertation thesis completed in 1962 and published later in 1964. A simplified version of this work is also available in Roberts (1978), which is here used as the source of information for the review.

Model development

This study focused on developing a model to simulate the undertaking of a R&D project within its surrounding environment. This environment included the market perception for the need of the product, the customer, the firm and their commitment to fund the project. The overall strategy adopted to conceptualise this model was based on a "top-down" approach, where a simple high-level model is first developed to represent the very basic feedback structure of the project system. "Level/rate" diagrams were used at this stage, supported by a high level contextual influence diagram identifying the model sectors and their interrelationships. As new relevant factors were identified and discussed, these were added to the model "level/rate structure", which grew more complex.

The starting point to conceptualise this structure focused on identifying the basic *control mechanisms* of project management and the associated *policies*. This project control is primarily based on schedule adjustment and resource allocation: as differences between the scheduled and the forecasted completion date are detected, possibly indicating delays, more resources are brought into the project in an attempt to catch up with progress. This simple mental model can be represented in a level/rate diagram as shown in figure 6.1 below:

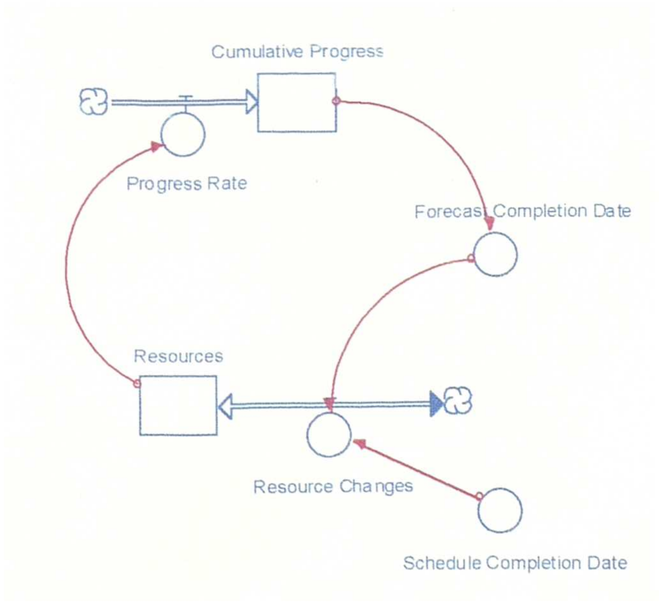


Figure 6.1 – The basic feedback process of project control presented in Roberts (1964)

The next step was to recognise that the schedule is not the only driver of project performance, but it must be compromised with the other two objectives of cost and quality. Therefore, similar control loops could be entailed with this one, representing budget and quality control, through the employment or deployment of resources. These three objectives could then be related through control *policies* which trade them one another. After having represented in the model the fundamental control processes of project management, the next step was to consider and represent the policies used to *perceive* information about the project, which may differs from the "real" reality. In particular, it is considered that errors are initially unperceived and are progressively recognised as the project approaches completion. This assumption is supported by the fact that in reality progress monitoring sources are not perfect. The difference between the real and the perceived project status is considered explicitly in the model. The final step was to consider that the performance of the actors in the project, both technical staff and managers, is strongly influenced by motivational issues like penalty-reward structures, knowledge of schedules, costs, perceived past performance, among others.

Model structure

The model structure did not consider any breakdown of the project work into phases or any breakdown of project resources. A single flow of tasks to be accomplished from "to do" to "completed" modelled the product development process. Undiscovered rework is explicitly modelled as work tasks that will need to be re-done. A single flow of resources into and to outside the project modelled the resource management process of the project.

Model quantification

After a stable level/rate structure for the model was achieved, representing the identified relevant factors of the dynamics of R&D, the next step was the quantification of the relationships in that structure. This included the quantification of subjective relationships, like the effects of perceived schedule delays on staff productivity, the staff reluctance in reporting tasks completed and finalised work to high level managers, among others. Since this model was intended to study the general dynamics of R&D projects, the quantification of these relationships was based on empirical evidence and reasoning.

Model validation

The model did not have to conform with accurate replication of observed behaviour of a specific real project. The basic model behaviour had to conform with the empirical knowledge of how R&D projects generally behave.

Model use

The next step was to carry out experimentation where different policies of assessing progress and handling scheduling delays were tested.

Summary

Overall, the model development process consisted of the two main phases of conceptualisation and quantification. The phase of model conceptualisation, aimed at identifying and representing the system feedback structure, was developed in a "top-down" fashion and based on a *policy* perspective: it starts from identifying first the core feedback processes of managerial control, using a level/rate notation. The definition of policies requires the identification of the sources of information used, and further the expected effects on the system. From this basic control feedback structure, other relevant factors and feedback processes can be identified. This basically includes issues about managerial perceptions and the secondary effects of control policies. Relationships between the variables in the model are then quantified. In general, these can be of two main types: objectively quantifiable, or subjective. Subjective relationships usually involve intangible or unmeasured information and relate to human issues (e.g. managerial perceptions or staff motivational aspects). In this study, the quantification of subjective relationships is primarily based on the modeller's empirical knowledge about R&D projects, gained from real life past experiences. Model validation is assessed by two main model features: its ability to replicate the *typical general* behaviour of an R&D project, and to provide plausible explanations of further behaviours reproduced in "what-if" experimental scenarios.

6.5.3 Cooper (1980) – a model of a real large-scale shipbuilding program

This was the first major reported application of a System Dynamics model to a real project. This work resulted in the development of a large and highly complex SD project model (with thousands of equations, as reported by Cooper). The model was primarily aimed at supporting a major delay and disruption claim. The work was undertaken by Pugh-Roberts Associates, a management consulting firm specialised in the use of System Dynamics.

Model development

The model development process was undertaken over the period of two years by a relatively small team, which involved some managers and a lawyer. The overall process followed a sequence of eight main phases (as reported by Cooper 1980):

(1) high level conceptual design,
(2) detailed design,
(3) specification of data requirements,
(4) data collection,
(5) preliminary quantification of a prototype model for a single project phase,
(6) informal review and rework of the singe-phase model prototype,
(7) statistical testing of the model to reproduce past behaviour,
(8) development of comprehensive model in three main iterations.

The work reported in this study does not provide any formal definition of the work undertaken within each of these phases, and in particular about the design phases. Nor formal elements of model design are reported. Since the model was multi-phase the conceptual design phase probably specified the project network and the basic feedback processes to be captured in the model. The detailed design probably consisted in the "level/rate" diagramming of the model.

The various phases of the model were modelled by specialised SD sub-structures (some of which sharing generic core structures), which were then assembled together to form the final model. The strategy followed for the model development process was therefore a "building bock" approach.

A key determinant during this process, as reported by Cooper, was the close involvement and commitment of managers from the programme. According to this author, the model development process followed was crucial for the success of this application. He further argues that the core of the model structure remained stable when applied to other future projects, suggesting some level of model generalisation.

Model structure

In terms of structure, the model represents and simulates the life-cycle of several projects being implemented in parallel. Each project consisted of several phases: engineering (system design, detailed design), production planning, material procurement, and a sequence of construction stages. The model simulates work being accomplished and flowing throughout this sequence of interrelated phases. The model structure further captures and simulates the several managerial decisions at different levels within the organisation. This includes: progress monitoring, reporting, and forecasting, work scheduling, acquisition and allocation of manpower, and high-level program and division management.

These project phases were modelled by specialised SD sub-models, which all share a common core structure, later reported by Cooper (1993, 1997) as the "rework-cycle". The full model also considers the many interactions between each of the phases of a single project and amongst several projects. Cooper claims that, as in reality, these form an essential part of the model feedback structure, and include: work availability interdependencies, error propagation and its impacts on quality, sharing of resources and its impacts on schedules.

The activities of management control incorporate the processes by which managers perceive performance, interpret deviations, and produce reactive control decisions.

Model quantification

The model was quantified in an iterative manner. A preliminary quantification was followed by several refinements, where the model was informally reviewed on the basis of expert opinion and data collected. The model was further statistically tested for reproduction of the desired behaviour, with specially attention to those areas where it was found to be weak or sensitive. This iterative process finished when the final model was able to replicate with accuracy the full programme history, as described by the vast amount of data collected and other information available.

Model validation

Validating the model went beyond behaviour reproduction, and consisted in providing evidence that the model was reproducing the right behaviour for the right reasons. This was achieved in two ways: defining "valid ranges" of values for the model parameters (these could not be violated when the model was calibrated to reproduce past behaviour), and further defining conditions of consistency among parameters values across the full model. Again, these conditions could not be violated in calibrating the model; secondly, the model was also subjected to "extreme-condition" tests (referred to in this study as "shock" tests): under radically changed scenarios, the model would still had to reproduce plausible behaviours. Finally, alternative model formulations were tested to check possible final structural improvements. The model was then used to suit its main purpose.

Model use

The model was used as a forecasting tool though retrospectively. The model was required to replicate accurately the outcome of a past programme, which was formed by several interrelated projects being implemented concurrently. The model was required to quantify the Client-responsible delays and disruptions costs, and to explain how these related to Client actions. This was achieved by simulating the programme with and without the Client-actions and by comparing the results.

Summary

Overall, the model development process implemented in this study introduced some important and original ideas. The fact that the model had to conform with a real project, in particular having to produce accurate "what-if" estimates, required more rigour in validation than in previous SD applications. The process was clearly implemented in a life-cycle fashion, comprising the following five main phases (not make explicit by the author):

- *model structural design* – although this is not described in detail, the study suggests that the model structure was build under a *process* perspective, where the first aspects of the system to be conceptualised were the system entities and their life-cycle. These entities consisted of work flowing throughout the several phases of product development, as well as the material and human resources flows;
- *specification of data requirements* – based on the conceptualised model structure, this phase consisted in identifying numerical data required to quantify the relationships in the model, and to represent the project behaviour that the model had to reproduce;
- *data collection and gathering of expert information* – some of the data required is measurable and can be collected directly from the programme information system. In a real scenario, this can be time and effort consuming task and is described by the author as a "massive effort". Where the data required is intangible, unfeasible to collect, or is unavailable, information based on expert opinion can be gathered from managers and staff . This is used as the basis to estimate the required numerical data;
- model quantification – using the data collected, the relationships in the model are initially quantified so that the model "works".
- model testing and validation – as described above, this phase is aimed at ensuring that the model is able to reproduce plausible behaviours within different scenarios. In the first place, it had to reproduce the observed past behaviour for the correct reasons. In this study, the author used statistical tests to validate the model's ability of behaviour reproduction (so called "goodness-of-fit" tests; see Sterman 1984), and has further imposed conditions of consistency among various parameters values. The model was also subjected to extreme conditions tests.

An interesting novelty was the use of prototyping, where an initial single-phase model was first developed prior to the development of the main model. This technique is typically used in software development, when the requirements of the system to be developed are not well defined. Iteration was also used, indicating that the above phases were not implemented in a purely sequential manner but rather the process has cycled back a few times. Finally, the author also refers to the use of extensive reviews of the model formulation in weak or sensitive areas based on managers expert opinion.

Overall, the development process can be described as a core logical sequence of five main phases, finishing with model validation and with the use of prototyping in the initial phases, and as requiring extensive data collection and close management involvement.

6.5.4 Richardson and Pugh (1981) – a simple model of R&D projects

This is another simple model of an R&D project, similar to the previous work developed by Roberts (1964, 1978). However, this author has used this model as an example to propose a structured process of model development in System Dynamics.

Model development

The model did not have to conform with a specific real project. The first step was to develop an informal but clear definition of the problem and of the model purpose. The problem was defined as persistent schedule and cost overruns in past R&D projects. These overruns lead to unexpected needs of hiring and training staff throughout the project life-cycle. The study focused on analysing the dynamics of an individual R&D project, to explore the role of the dynamic feedback processes. In this way, while simple the model was aimed at providing a *generic* representation of R&D projects.

Once the problem was identified and the model's purpose was well defined, the next step was the dynamic representation of the problem in the form of behaviour patterns over time. Two different scenarios were described in this way: a problem free project running smoothly towards its targets, and a problematic scenario where overruns occur and the project deviates from its targets. This representation led to the identification and definition of the main variables in the model that would represent the project status over time: estimated completion date, staff level, fraction of work accomplished, and work rate. These dynamic scenarios are referred to by the author as the *reference modes* of behaviour – the model must be able to reproduce both scenarios. The comparison between these two scenarios is the basis to develop an initial understanding of the main relationships among the project variables. A possible difficulty in this initial phase is the lack of numerical data to quantify the behaviour patterns – the required data might not be measurable (so called "soft" data), or it might not have been collected in the past. The author argues that in these cases data will have to be inferred based on intuition and expert opinion from those actors who are close to the problem. Therefore, as also argued by Cooper (1980), close management involvement is essential.

The next step was to define the model boundaries. This includes the elements of the real project to be included in the model, and the level of detail. Exclusion and aggregation of certain elements should be explicit and must always be justified on the basis of the model's purpose or on a balance between the benefits and the effort required. The author suggests three main areas of analysis to guide this process: (1) the physical processes within the system, (2) the information process through which these processes are perceived by managers and staff, and (3) how these perceptions motivate and are used to generate reactive actions, which in turn affect the physical processes. This is clearly a *process* oriented approach: model conceptualisation starts at the physical level of the system structure and then moves towards the formulation of the managerial control policies.

The next step was to conceptualise the system feedback structure in qualitative terms using "word and arrow" influence diagrams (IDs). The author proposes a "top-down" approach, starting with the development of a simple structure for a certain functional area of the project. From here, the model grows as other areas are identified along with their processes and overall feedback structure. Advocating a *process* perspective, the author argues in favour of identifying first the physical processes, then the managerial and staff perceptions about the system state, and the desired state of the system by these actors, then defining the gaps between desired and perceived state, and finally this information generates pressures (in particular management decisions) to change the system state towards the desired targets. The "top down" process evolved throughout various iterations where new areas of the system were modelled progressively as follows:

- physical process of workforce, and staff hiring control;

- physical process of work accomplishment and impact of workforce on this process;
- monitoring of work progress and its impact on staff hiring;
- schedule adjustment process, and schedule control policy;
- impact of progress monitoring on schedule control;
- rework and its impact on the physical process of work accomplishment;
- impact of rework on perceived progress and perceived productivity.

An important aspect of this qualitative phase is the statement of the *dynamic hypothesis*. This consists of a proposed explanation of the reference modes of behaviour, according to which these are as generated by the feedback loops identified in the IDs. This hypothesis is important and should be questioned throughout the whole model development process.

Having achieved a stable feedback structure for the model, the next step of the process was quantification. An important issue is that the author considered the IDs at the level of final quantification. This implies that each variable in the qualitative ID is directly translated into a mathematical equation. Since model quantification in System Dynamics is based on a "level/rate" definition of the system feedback structure, this also implies that all variables in the ID can be classified as a level, as a flow-rate, or as an auxiliary. This process consisted in identifying first the levels by testing the meaning of the variables in the ID as "accumulations", checking if they have a static meaning, and checking if their change is associated with a delay. Next followed the rates, identified as the variables which affect directly the levels. Finally the auxiliaries were identified. Again, the author proposed an approach to quantification based on decomposing the system into main functional areas and quantifying these separately.

Model structure

The final model is a simple single-phase model and highly aggregated. Aggregation is primarily imposed by the level of detail in which the physical processes are represented. For example, the workforce process does not consider explicitly staff training. Equally, the physical process of work accomplishment considers the full project as a single phase with no intermediate sub-phases and schedules.

Model quantification

Quantification of individual relationships in a System Dynamics model is critical to model validation. The first basic requirement proposed by the author is to ensure the *dimensional consistency* of the variables – while this is often referred to as a validation test, it must be always checked in the formulation of any equation. The author provides a rationale for the *structure* of the equations in the project model. Some of these equations will be easy to define (generally the ones which involve "hard data" and produce "hard data", like "work rate" = "staff in the project" x "individual productivity"). Other equations may involve or produce "soft" data, in particular the ones which involve the definition of managerial policies. For these, the author suggests the use of expert judgement, empirical evidence found in the literature, and ultimately "guestimates". A "guestimate" consists of developing a reasoning to produce an "intelligent" estimate as opposed to just a random guess. The author does not propose any particular classification for the type of relationships in the model. However, such a classification framework is proposed for the parameters and methods of parameter estimation – this is an important element of the quantification process. The author classifies the parameters as follows:

- *measures* – usually refer to "physical" characteristics of the system, like the size of the code in lines of instructions;
- *conversion factors* – usually refer to process characteristics of the system which imply transformation, like productivity which is used to transform man-power employed to tasks accomplished per day;
- *normal reference parameters* – usually refer to the base values of the system's rates that occur in normal circumstances, and are affected by the use of multipliers;
- *growth or ageing factors* – usually characterise exponential growths or decays;
- *adjustment times* – usually refer to delays in human perceptions of changes in the system, like "time to adjust perceived productivity".

This author argues that this classification is useful because each parameter type may call for different estimating techniques, as follows:

- directly from "firsthand knowledge" (i.e. from a *priori* observation of the characteristics of the real system, data collected, literature, interviewing people);

- directly from knowledge or data on individual relationships in the model (i.e. the parameter must conform with, and can be deduced from, a certain instance of a relationship in the model);
- directly from overall observed system behaviour;
- deduced from bounds and mental experiments;
- deduced from knowledge of system processes below the level of aggregation assumed in the model structure;
- deduced from patterns of observed behaviour in the real system.

As general guidelines, the author recommends that when parameters need to be deduced, this should be preferably done using knowledge below the level of aggregation. The more the assumptions, the more the likelihood for errors: using equations makes the estimate dependent on the model itself, thus limiting the level of confidence in the estimate. On the other hand, estimating parameters fully on the basis of the behaviour produced by the model may lead to "fixing" the estimate to fit observed behaviour. Finally, the author stresses the risks of using statistical techniques which typically aggregate cause-effect relationships, above the level of aggregation in the model. These are informal but important guidelines to the model development process, in which parameter calibration plays an important role. Further formality can be developed based upon the classification of parameters and types of relationships.

Model validation

In this work the author proposes an iterative approach to model development. Each iteration should finish with a testing phase, where the modeller may identify the need for reformulation of the model. This testing phase is based on experiments of model *sensitivity* to changes in parameters and structure. The author stresses the importance of differentiating the three main types of sensitivity: numerical, behavioural, and policy. This is important in understanding both the likelihood of model insensitivity to some parameters, and the significance of model sensitivity in relating behaviour to structure. This understanding is important to identify the need for reformulation.

The author also proposes the set of validation tests proposed by Forrester and Senge (1980), which are divided in four categories: structural suitability for purpose, behavioural suitability for purpose, structural consistency with reality, and behavioural consistency with reality. These tests support model reformulation throughout the model development process.

Summary

Overall, this study focused on the development of a high-level single-phase generic model of an R&D project. Before conceptualising the model's feedback structure, the author suggests the dynamic definition of the relevant scenarios of project behaviour. This includes both a problem-free scenario and a problematic scenario. The next step is to identify the model boundaries and to develop the system feedback structure using IDs. The author proposes a "top-down" approach, dividing the model in functional areas. A process perspective is also proposed to identify first the relevant physical processes, and upon these the information-feedback processes.

The structure of this model incorporates the physical processes of workforce, work accomplishment, and rework, the managerial perceptions of productivity and work accomplished, and the control policies of schedule adjustment and staff hiring. There is no work decomposition and the project is considered as a single phase to be undertaken within a single schedule. An important guideline for the conceptualisation of the model feedback structure is the definition of a dynamic hypothesis, which explains the dynamic problem as being generated by the feedback processes. The project ID is developed at the same level of detail as the level/rate diagram, thus at the quantitative level. The next step is therefore to identify the variables in the ID as levels, rates, and auxiliaries. The quantification of these variables is a critical issue and the author proposes a classification of the parameters in the model, as well as a classification of possible parameter estimating techniques. The author stresses the risks of statistical and of behaviour based estimating techniques, and advocates the use of process information below the level of model aggregation. The overall development process is proposed as being iterative and wherein sensitivity testing are essential to improve and reformulate the model. Model validation is based on the principle of confidence. It must pass a set of structural and behaviour tests regarding both purpose suitability and of consistency with reality.

6.5.5 Abdel-Hamid and Madnick (1991) – a project model of software development

In this work, the authors developed a SD project model for a software project. This is one of the most comprehensive descriptions of a project model available in the literature. Preliminary work was developed earlier (Abdel-Hamid and Madnick 1983, Abdel-Hamid and Morecroft 1983, Abdel-Hamid 1984). The primary aims of this study were: (i) to help understanding better the *generic* process through which software projects are managed, and (ii) to make predictions about the overall technical and managerial process by which software systems are developed.

A key feature of this SD model is the integration of all the separate functions of software projects. This study consisted in three main steps: developing the integrative model, use a case-study real project to test the model, and use the model to analyse the performance of alternative managerial policies in the software development process.

Model development

The model development process followed in this study consisted in using first the simple R&D model of Roberts (1964, 1974) to identify and describe the basic project management process under a feedback perspective. The second step was to define the model boundaries in terms of:

(1) *software development life-cycle phases covered* – this ranges from the beginning of the design phase to the end of the testing phase. The phases of requirements definition and system installation and maintenance were excluded;

(2) *software production activities considered* – development (designing, coding and integrating), quality assurance (QA activities, like formal inspections, unit testing), rework, and testing (including the rework of errors);

(3) *managerial functions considered* – schedule and workforce adjustments, human resource management, and other control based on cost monitoring;

(4) *organisational boundaries* – these include only the development team, comprising technical staff and managers. The interactions with the Client were excluded.

(5) *project dimension* – a medium size project is considered within the range of 16 to 64 thousands lines of code.

The model boundaries reflect some important assumptions, in particular that the system requirements are kept stable throughout the project life-cycle. Another major assumption was to aggregate the whole project into a single major phase, with a single schedule and budget. Within this phase the three activities in (2) are performed continuously. Therefore, like in the previous models by Roberts (1964, 1974) and Richardson and Pugh (1981), there is no breakdown of the project into sub-phases.

The third step was a "top-down" approach to define the model high-level structure as incorporating four major interrelated sub-systems: human resource management, planning, software production, and controlling. This decomposition is consistent with the traditional dynamic view of project management, as discussed in Rodrigues and Bowers (1996a). The first two sub-systems refer to the basic project planning function; software production corresponds to the engineering process of product development, and controlling is the basic project monitoring function. The definition of this high-level structure (referred to in this research as model *architecture*), specifies the model sub-systems, their interrelationships, and the associated information exchange (e.g. the planning sub-systems indicates the current schedule to software production).

The fourth step in model development was to carry out an initial set of interviews with software project managers. These were "focused interviews", aimed at collecting information about how projects are really implemented in each of the four areas identified above. On the basis of this information, an initial simple model was developed as a "prototype" (referred to by the authors as "skeleton model"). This prototype model was then used as a "road map" in the next step, which consisted in an extensive inter-disciplinary literature review of the various areas of: project management, management control, psychology, and others. The information gathered in this literature review was used to develop a more detailed model.

The final step consisted in an iterative process wherein a second set of interviews was carried out. These interviews provided the answers to the remaining critical modelling questions. They were used to expose the detailed model to criticism and review in an iterative manner, until a satisfactory model was achieved. An important issue is that the model was presented to the interviewees only in the form of simple influence diagrams, identifying the main feedback loops. This was perceived to be a language more accessible to managers than the quantified "level/rate" diagrams.

Model structure

The final model had a simple but rich structure, containing a few hundred equations. The overall process structure consisted of three main entity flows: human resources, work tasks, and errors. The basic "level/rate" structure of work tasks and errors is shown in figure 6.2 below: the software tasks, measured in lines of code, flow throughout the three main activities (rates) of development, quality assurance, and testing. The associated error co-flow indicates that errors can be detected and reworked during QA, or can escape being later detected and fixed in testing.

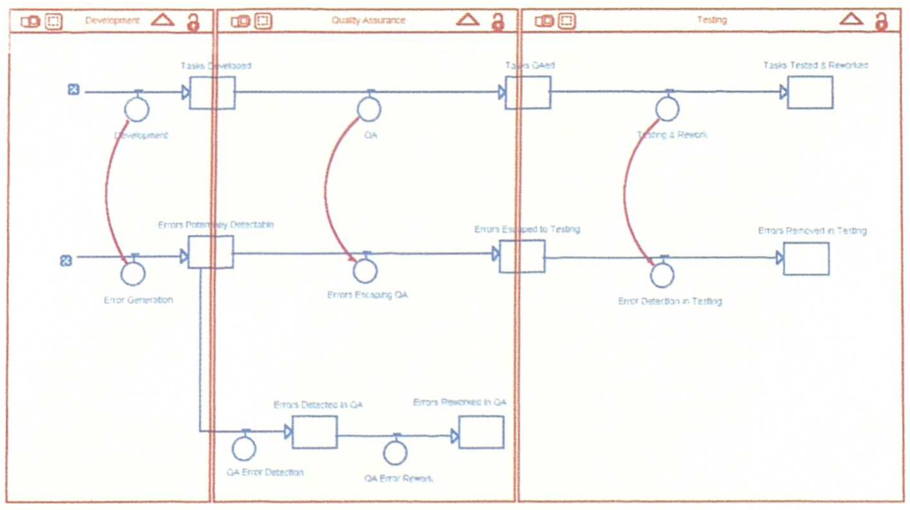


Figure 6.2 – Basic co-flow process structure of work and errors in the model proposed by Abdel-Hamid and Madnick(1991)

Within each of the four sub-systems several factors affecting software development and management were considered in detail. Many of these relate to "soft" human issues. A particularly interesting feature is the error "re-generation" process considered in the model. A summary of the main factors considered in the four model sub-systems are presented in the table 6.1 below:

Sub-system	**Main factors addressed**
Human resource management	• Workforce experience level • Training overheads • Staff hiring and assimilation delays • Staff transfer from or to other projects • Staff turnover
Software production	• Real and perceived cost to rework • Desired rework delay • Shift in man-power allocation between development and testing • Impact of work progress on: productivity, error generation, error detection, and error rework • Impact of schedule pressure on: man-power availability, error generation, QA "cuts" • Impact of communication overheads on man-power availability • Use of over-time and slack-time • Staff exhaustion • Impact of staff experience level and of schedule pressure and on error generation • Impact of undetected errors on error generation (i.e. error re-generation) • Impact of error density on error detection
Planning	• Schedule adjustment • Impact of workforce stability on workforce adjustment • Impact of schedule stability on workforce adjustment • Actual schedule and maximum tolerable completion date
Controlling	• Development productivity: planned, real, perceived, projected • Testing productivity: planned, real, perceived • Cost to rework: real, perceived • Budget adjustment • Impact of "size growth" (underestimation) on budget adjustment

Table 6.1 – Major factors addressed in Abdel-Hamid and Madnick (1991)

Model quantification

The structure of each of the model's sub-systems and the inclusion and quantification of the factors in the table 6.1 above, was supported by a rationale based on two major sources of information: the interviews, and the literature review. The quantification was undertaken separately for each sub-system.

Model validation

While the quantification process does not ensure full model validity, it does provides the ground for empirical structural and parameter validation (i.e. an empirical approach to the validity of concepts and of data). In order to validate the model, behaviour validation was also required (Forrester and Senge 1980, Barlas 1994). The authors have calibrated the model to reproduce the past behaviour of medium-size software project at NASA. Based on this tentative application, the model was further readjusted and improved.

Model use

The model was finally used to diagnose the past of the case-study project. This uncovered important conclusions about managerial performance, which otherwise could have remained "hidden" in the project history. Experimentation with the model focused on the analysis of staffing policies under conditions of schedule pressure (e.g. Brook's law), on the economics of Quality Assurance (i.e. impacts of different QA levels on the project outcome), on the impacts of the initial estimates (different estimates create different projects), and on the occurrence of the 90% syndrome.

Summary

In summary, this study focused on the development of a project model integrating software management with software production, for medium-size software projects. The model focused on the internal development team, and comprised that part of the life-cycle from system design to system testing. The managerial control policies considered refer to schedule adjustment and to staff and man-power acquisition. The model development process was based on a "top down" approach of decomposing the project system into four main functional sub-systems, and further identifying the exchange of information among these. The detailed conceptualisation of the model structure and its quantification was undertaken separately for each sub-system. The overall process was also based on extensive information and data collection from the literature and from interviews with several software managers from different corporations.

A prototyping approach was also used, where an initial "skeleton" model was first developed. This model was the basis for the collection of detailed information, and was further refined through several iterations into a detailed model. The information and critique from software managers played an important role in this process. While the model was directly developed using "rate/level" diagrams, general influence diagrams were preferred to communicate its contents to managers.

In terms of structure, the model decomposes the project into four main areas of activities: in the management process, planning, human resource management, and control are considered; in the engineering process there is a single software production area; software production is modelled as a single task wherein four main activities take place continuously: development, QA, rework, and testing. Associated with these activities, the software production is also decomposed into two interrelated flows of work and errors. The whole project is decomposed into three main "physical" entities: work, errors, and staff. The model does not consider any schedule and budget decomposition of the project into individual sub-tasks, and hence it can be considered as being highly aggregated.

Model quantification was primarily based on the data and information collected from the interviews and from the literature. The calibration of the model parameters and relationships for specific scenarios was based on quantitative data about those scenarios, and on the anticipated expected output from the model. As proposed by Richardson and Pugh (1981), definition of bounds, mental experiments, and knowledge of the processes below the level of aggregation were used to estimate the calibration parameters. However, no formal or structured calibration procedures are proposed.

While this System Dynamics study is considered by many as a landmark in the field of SD applied to software project Management (Yourdon 1993; it has been the subject of further studies like in Madachy 1996), the model has some important weaknesses. For example, it fails to consider a pre-planned staff profile within the planning function (as in reality), and it does not consider the natural changes of work intensity that result from the internal structuring of the project work into sequential, parallel, or overlapping sub-tasks. These natural changes of work intensity are an essential factor for a varying "problem-free" shape of the staff profile (typically a "Raleigh curve", Boehm 1981, DeMarco 1982). Instead, the final staff profile is fully determined by problems and deviations from the plans, and in particular by the fact that the estimation of the current "cost at completion" in the control sub-system does not take into account any contingency for those errors, which are always likely to be detected in the future (hence the "cost at completion" is underestimated). Other limitations could be discussed beyond the scope of this chapter. The ones just presented above are also found in the other models discussed in this chapter (based on information available in the literature).

The major contributions of this work are the proposal of a generic System Dynamics structure for software projects, the identification of important feedback effects, like "error re-generation", and the extensive data collection and rationales presented to support empirical validation of model quantification.

6.5.6 Lin (1993), Lin and Levary (1989) – software project models: SLICS and SEPS models

This work was undertaken at NASA's Jet Propulsion Laboratory. The aim was to develop a *generic* model to simulate the development life-cycle of a software project. Preliminary work is reported in Tausworth et al (1983) and McKenzie et al (1984). Later work presented in Lin and Levary (1989) reports the development and use of a generic SD project model for the software development process called "Software Life-Cycle Simulator" (SLICS). The particular feature of this development is that the use of the SD model is embedded within an expert system. This incorporates a knowledge database, which is updated from the experiments carried out in the model. The authors propose the use of this tool to support the management of on-going software projects.

Model development

The model development process was based on a "top-down" approach of decomposing the software project into three main sub-systems: product development process, staffing, and project control based on cost analysis. These sub-systems are interrelated and exchange information.

The model was developed and represented using "level/rate" type of diagrams.

Model structure

There are two important features of this model: (i) the product development process is decomposed into a sequence of phases, according to the classic life-cycle of software development, and (ii) the model considers explicitly the occurrence of requirements changes in each of these life-cycle phases. Therefore, the model considers a breakdown of the project into a sequence of sub-tasks, each with its own planned schedule and budget. This is more detailed than the "single-phase" approach adopted in other models, like in Abdel-Hamid and Madnick (1991). As a consequence, the managerial decisions must consider work accomplishment towards intermediate schedules and budgets. However, this level of breakdown does not consider parallel tasks but only a linear sequence of life-cycle phases.

Model quantification

The identification and quantification of the several cause-effect relationships in the model was based on two main sources of information: current literature, and empirical research. The second consisted of questionnaires and interviews with managers of software project.

The authors propose an intensive calibration process during the life-cycle requirements phase. This phases is sub-divided into several time segments. At the end of each segment, data and information is collected to "characterise" the project. The parameters in the model are calibrated according to this data. This is further tested and re-adjusted so that the model reproduces the expected project behaviour, at this stage in the life-cycle (i.e. according to the plans). At the end of the requirements phase, this base calibration is completed and the quantification in the model is assumed to reflect the real characteristics of the specific project being analysed.

Model validation

Model validation was primarily based on testing the model's ability to reproduce the behaviour of past projects. The model was calibrated according to the procedure above, using historical data from the requirements phase of a past project. Statistical tests of "goodness-of-fit" were used to assess the model's accuracy in replicating the actual project behaviour throughout the full life-cycle.

Model use

The model is aimed at supporting the management of on-going projects. An expert system is used to enhance this process.

The use of an expert system has three main purposes: (i) to ensure validity of the inputs to the SD model, (ii) to help interpreting the output produced by the model, and (iii) to create a knowledge database capable of suggesting recommendations and of anticipating results, preventing the need to perform simulations. While this is an interesting ambitious application, it raises some critical issues like (a) replacing the personal judgement of the user, and (b) the validity of the knowledge database in the face of changes in the model.

Further developments – the SEPS model

Later, further work was reported by the same author at NASA JPL presenting the development of a *generic* project model for the software development process, called Software Engineering Process Simulation (SEPS, Lin 1993).

Again, the development of this SD project model is based on a high level architecture, where a software project is decomposed into the two main life-cycle processes of engineering and management. The engineering process comprises the various activities of product development (e.g. designing, coding, testing), while the management process comprises the basic control functions of planning and monitoring.

The structure of the SEPS project model comprises five major sub-components: software production, workforce, scheduling, budgeting and cost monitoring, and management decision infra-structure. The production sub-model simulates the engineering process of software development, which is decomposed into a sequence of life-cycle phases, like in the SLICS model. The workforce component simulates the human resources process where the staff flows through the project. The scheduling and the budgeting models simulate the management functions of time and effort monitoring and estimating. The management infra-structure uses the information provided by these four sub-models to generate reactive decisions of project control according to the policies incorporated in the model. An important feature of the model structure is the explicit breakdown of the project into a sequence of phases, thus considering intermediate schedules and sub-budgets.

The validation process of the SEPS model, as reported by the authors, included the "traditional" confidence tests proposed in Forrester and Senge (1980), complemented by some additional techniques. Three are reported by the authors: the first consisted in replacing the policy decision roles incorporated in the management component of the model by real managers – i.e. the simulation is interrupted in each control cycle, so that managers could take their decisions instead of the model. This was intended to test whether the original decision-rules in the model were producing decisions consistent with the real managers "mental rules". A second technique focused on testing the model for sensitivity: key parameters were changed continuously within bounds of extreme values and the behaviour changes produced by the model were checked for consistency by managers' expert judgement. Finally, the authors have used Turing type of statistical tests (Turing 1950), where managers are given various scenarios of project behaviour. Some are real scenarios while others are simulated by the SD model. Managers are then asked to "guess" the origin of the scenario: real or simulated. This process is intended to demonstrate that the behaviour patterns produced by the model are consistent with reality and hence managers cannot distinguish between the two. Although these are interesting techniques that help to build confidence in the model validity, they cannot alone guarantee model validation.

Summary

In summary, although the SLICS and the SEPS models are presented as different models it appears the latter is an evolution of the first and hence both have many common features. The model development process was based on a "top down" approach of decomposing the software project into main functional sub-systems. The project is decomposed into a sequence of sub-tasks according to the classic life-cycle of software development, and hence the models consider explicitly intermediate schedules and budgets. The main sources of information used to quantify the subjective relationships were the data published in the literature, questionnaires and interviews with software managers. Model validation is primarily based on confidence tests using managers' expert judgement. These tests assess both the plausibility of the behaviour produced by the model, and the ability of the model to reproduce observed behaviour in past projects. Both models are presented as "generic, and hence their application to particular projects is based on parameter calibration for the specific project. The models are also proposed as control tools to be used to support the management of on-going projects on a regular basis.

6.5.7 Williams et al (1995), Ackerman et al (1997) – a model of large-scale design and construction project

This work was the first major application of System Dynamics modelling to a real project in Europe. This was a litigation case of delay and disruption, where the model was used to support dispute resolution after the project has been completed. In some ways this work was similar to the application reported by Cooper (1980), but the methodology employed in developing and using the model was different and innovative.

Model development

The authors have adopted a cyclical iterative process mixing the use of "soft" and "hard" methods of Operational Research. There were mainly three methods: cognitive mapping using COPE, influence diagramming also using COPE (i.e.

qualitative System Dynamics), and SD simulation modelling using Stella (i.e. quantitative System Dynamics). These methods were also supported by some spreadsheet modelling. The authors report considerable benefits of this inter-disciplinary methodological approach. These benefits were gained through various synergistic effects. Although this work does not propose a formal structured approach to the model development process, it makes discusses some critical issues and important difficulties of SD modelling, in particular model validity (Berlinsky 1976) and the numerical escalation of errors (Alonso 1968)

The overall development process was iterative but also followed a natural sequence of phases. The first step was to use the "cognitive mapping" technique (Eden 1988) to model each of the individual views of senior managers about the problems within the project. A cognitive map provides a very high level view of a system's elements and their interrelationships. It consists of concepts about the system linked through "lines of argument" (Ackerman et al 1997). This can lead to a large map, which can be divided into several sub-maps, and can also be aggregated into a more concise global map. The next step, still at the cognitive mapping stage, was to merge the individual maps into a single shared map, called the "group map". This would contain a shared view of the problems agreed by all the managers. The "group map" promotes the debate of conflicting views while the individual maps are being merged. The third step was to "convert" the group map into a System Dynamics influence diagram. This conversion process consisted in extracting the main feedback loops in the cognitive map, and replace their internal concepts and "lines of argument" through variables and cause-effect relationships. This leads to an increased level of formality, from the cognitive map to the influence diagram, in representing the feedback structure. The final step was the quantification of this influence diagram into a simulation model, represented in Stella as a "level/rate" diagram. Within this sequence, the process cycled-back several times to previous stages. The modelling techniques of Cognitive Mapping, Influence Diagramming, and SD simulation modelling, were brought together to develop three different types of models of the same system. The authors stress that these methods and models inform and strengthen each other.

The structure of this mixed development process highlights an important issue in quantitative modelling of complex social systems. Although the aim was a formal quantitative simulation model, the first step in modelling a complex (and poorly understood) system required a flexible framework in order to structure, refine, and represent managers' informal mental models. In this work, the authors have found that even the qualitative influence diagramming framework of SD was too rigid to start with. The first step in structuring and bringing discipline to managers mental models was implemented using Cognitive Maps. This highlights that the more formal (and potentially more accurate) is the modelling method, the more rigid is the underlying framework of representing informal mental models. This rigidity makes it difficult to develop the model from a blank sheet as well as to validate its structure. On the other hand, the less formal and flexible is the method, the easier it is to represent these mental models. However, this is at the expense of accuracy in describing the system and in predicting its outcome in "what-if scenarios". The mixed approach suggests that the modelling process should start with the softer methods towards the hard quantitative ones. Each method develops a model capable of producing as an output a more formal and structured description of the system and of the problem; this output is then used as an input to the subsequent modelling method. The process evolves until the desired formal quantitative model is achieved. The process iterates until all models are perceived to be valid. Finally, the authors argue that the models not only serve the purpose of informing each other, but they also have the role of providing different views and perspectives through which the problem can be diagnosed and a solution can be found.

Model structure

The cognitive group map produced in this study is reported to contain 760 concepts and 900 links. The System Dynamics influence diagram developed from here included 87 variables and 146 cause-effect relationships. This considerable reduction in the number of descriptive elements reflects perhaps the elimination of redundant descriptions in the more informal cognitive map, the selection of the more relevant factors and feedback effects, and also the aggregation of these factors and effects. The quantitative simulation model is reported to contain about 323 equations, including levels, rates, and auxiliaries. The increase in the number of variables reflects perhaps the need to decompose aggregated variables, as imposed by the stronger formality of a "level/rate" diagram.

The structure of the final model decomposes the project into two aggregated major phases: design and manufacturing, with various feedback effects being considered between the two. Although each of the components holds considerable complexity, this is still a high level of aggregation in decomposing the project (this reflected the model purpose: it was not intended to be a detailed "generic" model for that type of projects, but to analyse the complex feedback effects within and between these two major phases in the specific project). The flows considered within each of the two sub-models consisted basically in work-flow entities, and resources flowing throughout a life-cycle of several states. An important feature is that loops were considered in the work-flows (i.e. work tasks might flow back to previous states). This reflects a concern to consider iterations within the engineering process, as it happens in reality.

Model quantification

The quantification process was based no two main sources of information: interviews and workshops with senior managers, and extensive metrics collection from the project (reported as an intensive effort-consuming task). The authors report "typical" problems of quantification in System Dynamics (Ackerman et al 1997). These have to do with the data requirements of a SD model, which can be a major problem to validate and calibrate any SD model. This includes:

- data collected in most management systems is "event based" hence inappropriate for the continuous perspective of SD;
- in large project management information systems, data and information provided by different sources is often inconsistent one another;
- in some cases the required data is simply not available;
- the format of the data available does not match the required format for the SD model; the conversion process is not obvious and sometimes is expensive in terms of effort.

Model validation

Validation at the qualitative level was primarily based in expert judgement from senior managers, through a series of workshops where the cognitive map was developed. Validation of the quantification process has followed some basic confidence tests, like the dimensional consistency of equations. The main requirement to validate the model was its ability to reproduce with acceptable accuracy the problematic project past behaviour, and that the parameter values for this calibration were supported by acceptable evidence – i.e. the model had to produce the right behaviour for the right reasons. The model also had to reproduce in an acceptable manner a set of hypothetical scenarios likely to be required in a court case. In judging the model's ability to reproduce real scenarios, the authors argue in favour of a general match of curve slopes, maximum, minimum, and total cumulative values.

Model use

The use of the model followed a "standard" process adopted in this type of application (i.e. dispute resolution), as reported by Cooper (1980). The model is first calibrated to reproduce the "problem free" scenario, where the disruptive Client actions are removed. In this scenario the model shows that some over-runs would eventually occur, less than what happened and for which the Client is not responsible. The model is then calibrated to include the Client actions as they occurred in reality (e.g. delays in approving documents, requirements changes). In this second scenario the model must reproduce the observed over-runs. The difference between the results of the two simulations quantifies the portion of the actual over-run for the which the Client is to be held responsible.

A critical issue regarding model use reported by the authors is that while a SD model aims at providing a simplified representation of reality, the model itself may still be too complex at the eyes of the end-users. In order to overcome this difficulty, the authors argue in favour of using the qualitative Influence Diagrams to identify the key feedback loops and to illustrate their influence over the project behaviour. The simulation model is then used to quantify the actual impacts. The authors argue that a simulation model represented in a "level/rate" type of diagram does not provide a clear view of the individual feedback loops and of the overall feedback structure (unless the model is extremely simple). The qualitative COPE map and influence diagram played an important role in relating the model behaviour to the basic feedback structure, a major current difficulty in System Dynamics (Richardson 1996).

Summary

In summary, this study was aimed at developing a model for a specific project, capable of recreating its past behaviour with accuracy. The model development process was based on a mix of three main modelling methods, all aimed at identifying the system feedback structure and relating this to the observed behaviour. The methods assume different levels of formality, from the "soft" cognitive maps, to a qualitative influence diagram and finally a "hard" quantitative SD model. This stepwise approach has proven effective in converting and refining managers' mental models into a formal quantitative model. The process followed a sequence of increasing formality and was also implemented in an iterative manner.

The sources of information used were mainly information form interviews with senior managers, and extensive data collection from the project. The model structure assumed a breakdown of the project into two major sub-tasks of design and construction. The entity-flows in each of these tasks consisted of work flowing through several stages of product development, and several types of resources being employed. The occurrence of errors was modelled through work-entities flowing back to previous stages.

The quantification of the relationships was subjected to some basic confidence tests, and was based on data estimated from managers' mental models. Like in most applications, no specific techniques are provided to quantify subjective relationships involving "soft" variables. Model validation was primarily based upon the model replicating the project past behaviour, and calibrating parameter values to which there was strong evidence of occurrence. The model's ability to reproduce plausible outcomes under different scenarios was subjected to managers judgement. The final SD model, although consisting of a simplified representation of reality, was still a complex model with nearly 350 equations.

Given the difficulty in providing a clear representation of the system feedback structure, qualitative influence diagrams were used to help relating the model's behaviour against this feedback structure. Like in the previous studies discussed, the authors have followed a personal approach to model development based on the generic steps of SD modelling, thus reflecting the absence of a common structured framework.

6.5.8 Ford (1995), Ford and Sterman (1998) – a generic model of product development projects

This work proposes a generic model to represent development projects that deliver a durable and tangible product. This generic model focuses on the analysis of two key factors that affect the performance of this type of projects: (i) the structure of the development process, and (ii) project co-ordination among the development tasks (caused by concurrent development). The structure of the development process is characterised by a network of tasks which are inter-related in various ways. The proposed model is therefore a multi-phase model.

The generic model is proposed as the basis to develop a proper SD project model for any specific product development project. A case-study project is provided as an example, where the model was calibrated to reproduce the project past, and to carry out retrospective "what-if" analysis to assess the impacts of different co-ordination policies. Adjusting this generic model for a specific project requires parameter re-calibration as well as structural tailoring. However, while this example is useful, no structured process is proposed to carry out this modelling task.

This work was first proposed by Ford (!995) in a PhD dissertation thesis and a later discussion of the same model was presented by Ford and Sterman (1998). The basis of the following summary is the first version presented in Ford (1995).

Model development

The model was developed by linking various SD sub-structures as "building-blocks", although no structured approach is proposed to conduct this process. Each of the "building-blocks" was developed separately and these were then integrated into a single model, which was tested for behaviour reproduction.

In this work, the model is more "presented" to the reader as a final model than it is developed in a step-by-step fashion. This presentation follows a top-down approach. First, the overall model structure is presented as a set of main subs-systems, wherein various project activities take place (including both product development and management type of work). In the "process structure" sub-system, the structure of the product development project is presented as a network of inter-related tasks. Within each sub-systems, various SD sub-structures are required. These are presented separately, supported by rationales and literature. Finally, the behaviour of the integrated model is analysed and sensitivity testing is carried out.

Model structure

The author's main priority was to capture the product development process in more detail than in previous models. There is also an attempt to merge several important project dynamics already analysed separately in these previous models, in order to capture their combined effect – the author claims that this is a novelty, not attempted before.

The model therefore re-uses SD structures published in the literature for some areas, and proposes new structures for other areas. The overall areas covered by the model are described by the following sub-systems:

- *process structure* – models the product development process as a network of tasks linked through dynamic inter-dependencies. Within each task, development and rework activities are carried out, including co-ordination due to rework discovery in other inter-related tasks;
- *scope* – models the scope growth due to generation of errors and consequent accomplishment of rework. Rework can emerge due to intra- or inter-phase error discovery;
- *targets* – models the process of adjusting the project targets of schedule, cost and quality and assesses project performance;
- *resources* – models the process of adjusting the project resources. It captures the various dynamics of human resource management, previously explored in other models.

The sub-systems where new structures are proposed, are the focus of the research – according to the author, previous SD models do not address them properly.

These are the process structure and the scope sub-systems. In some cases, the new structures proposed re-use some elements or concepts present in previous models. The areas of targets and resources are modelled using sub-structures from existing models, with some modifications as required.

In the process structure and scope sub-systems, the product development process is modelled as a network of inter-dependent phases (i.e. an inter-phase model). While the author does not impose an upper limit to the maximum number of phases, the case-study model considers four phases and five phases are suggested as a proper representation of any type of product development project. Within each phase, the SD generic structure proposed considers five activities: base-work, quality assurance, rework and co-ordination. The latter activity adds to the other three already proposed in the model of Abdel-Hamid and Madnick (1991), whereas Cooper's model (1980) only considers the two activities of base-work and rework. The inter-dependencies between the phases are dynamic, in the sense that the restrictions they impose will change as progress evolves in the two inter-related phases. The inter-dependencies considered refer to various aspects of the product development process and include: the impact of work progress in the predecessor on work that can be started in the successor, the impact of quality in the predecessor on error generation in the successor, rework discovery impacts, and co-ordination of rework. Internal work dependencies within each phase are also considered, modelled as intra-task dynamic dependencies. Rework is generated within a phase due to errors discovered by its internal QA activity or by another downstream phase. Whenever an error discovered refers to an upstream phase, or whenever a downstream phases discovers an error released by the task, co-ordination work between the tasks is required in order to accomplish the rework.

The targets and resources sub-systems incorporate the management processes through which product development is controlled. While the SD structures are taken from existing models, there is one important feature considered: the assessment of performance and the readjustment of the targets is implemented at both tasks and project levels. The control of the whole project is implemented through a structure which merges the results at the task level. The performance at the task level influence the target readjustments at the project level, which in turn affect managerial control at the task level. So there is a continuous feedback between the management control processes at the task level and at the project level. Another important assumption is that one single resource type is considered per each task in the product development network.

At the task, level the SD structures are replicated through the use of arrays, and hence they are exactly the same for all tasks.

The generic SD structure proposed to model the process structure and scope sub-systems is claimed by the author as the major novelty of this work. It is suggested that any development process in various industries can be modelled using an appropriate network of tasks, each modelled by this generic structure. Developing a new model therefore consists in linking several of these structures and calibrating them (in particular their dependencies), in order to represent the specific product development process. The other sub-systems of the model (targets and resources) are then linked into this network, to model the managerial control of the product development process.

Model quantification

The approach followed to quantify the model is very similar to the one reported in Abdel-Hamid and Madnick (1991). The key sources of information were: (i) data available in the literature, (ii) data collected and/or available in the organisation (e.g. from metrics programs), and (iii) informal interviews with managers and staff. Some of the referenced literature includes previously developed SD project models, like the one in Abdel-Hamid and Madnick (1991), as well as publications on software engineering and product development metrics. Data from literature is used mainly to calibrate relationships in the model which tend to be independent from the specific project. Data collected and interviews generally address project or organisational specific type of relationships.

An important issue in this work is that, first a generic five-phase model is proposed for product development process, and then this model is re-calibrated and structurally readjusted to represent a specific case-study project. Re-calibration refers mainly to parameter changes that reflect the case-study project and organisational issues. The structural readjustment included not only the establishment of the appropriate network of tasks, but also changes to the generic SD structure previously proposed to represent any product development task. This suggests that structural tailoring needs to be part of a proper model development process, even when generic SD structures are in use. A structured approach to accomplish this modelling work is not proposed

Model validation

Part of model validation is carried out when the SD "building blocks" are presented separately. This type of validation focuses essentially on structural issues and is based on logical rationales to justify the relationships proposed, in some cases supported by the literature.

The next step in validation was to carry out structure-oriented behaviour tests on the generic model. This consisted in: (i) checking whether the model was able to reproduce typical modes of behaviour of development projects (including single-phase multiple-phase scenarios), and (ii) identifying sensitive parameters and further assessing the model's sensitivity to changes in these same parameters.

These two steps were aimed at validating the generic model initially proposed. The final step was to validate the new version of the model readjusted for the case-study project. This required the validation of both structural readjustments and parameter re-calibration. Structural changes were mainly required to represent the specific characteristics of the product development process of that project. Evidence of these characteristics (e.g. project documentation) and logical rationales were presented. Calibration was mainly validated through data collected in the project, and from structured interviews with managers and staff (no formal technique was used to merge the various points of view of each interviewee). Finally, the behaviour produced by the model was checked against the historical behaviour observed in the case-study project (which was a multi-phase scenario).

Model use

The case-study was a past project and so any "what-if" analysis could only be carried out retrospectively. The purpose of this study was not to actively influence the management of an on-going project. The main aims of the case-study project were: (i) to validate the generic model proposed, showing its ability to reproduce a real project, and (ii) demonstrate that the structure of the product development project and of co-ordination have a strong impact on the performance of a development project. The latter was achieved by carrying-out retrospective "what-if" analysis in the case-study project, and thereby showing that different co-ordination policies produce significantly different results. For this purpose, this control policy was first described and identified in the model structure (I.e. how the policy was represented in the model). The parameters that reflect different implementations of this policy were identified and different scenarios were proposed and tested in the model. The validity of the generic model was based on the demonstration that the model's feedback structure was required to explain the produce behaviours.

An important aspect of this work was that the model behaviour was explained using qualitative IDs, which represented the model structure. Similarly to most of the previous authors, these IDs were considered as a more appropriate means to explain the model behaviour, than the "level/rate" diagrams. Furthermore, the author placed particular emphasis on using the concept of "feedback-loop dominance". This consists in identifying the specific loops which are dominating the project behaviour in certain time periods, according to the observed shapes (e.g. exponential growth causes by a specific reinforcing loop). However, the approach used to carry out this "feedback loop analysis" was informal, and was not supported by any type of quantification (e.g. possibly, indices that would quantify the "strength" of the loops over-time).

Summary

The project model proposed in this work is intended to be a *generic* multi-phase model of product development type of projects. In order to represent a specific project, the model needs to undergo structural adjustments and parameter re-calibration. The major novelties of this generic model are the explicit structuring of the product development process and the incorporation of co-ordination work. Therefore, the model is appropriate to assess the impacts of these factors on project performance. In this model, the author also attempted to merge various other dynamics of development projects considered separately in other past SD project models. The data and other information required to support model validation was gathered from the literature, from other SD past projects, from data collected and informal interviews in a case-study project.

Tailoring this generic model for a specific project, requires the explicit representation of the product development process as a network of inter-dependent tasks. Each of this tasks will then be modelled by a generic structure proposed in this work (the "development task" structure). These structures are then linked according to the specified network, through the establishment of dependencies. Various types of inter-phase dependencies are considered in the generic model, all of which are of dynamic nature. While this type of dynamic dependencies is present elsewhere (Homer et al 1993), the author provides valuable descriptions of these dynamics, which are not present in most past project models (though not reported in the literature, it is probably also considered in Cooper 1980 and in Williams et al 1995). The explicit consideration of co-ordination leads to the inclusion of a co-ordination activity within the "development task" generic structure, together with base-work, rework and QA activities.

While a case-study project is provided as an example of how to tailor the generic model to a specific project, no structured approach is proposed for this modelling process. This includes the validation of the model, which in this work is based on the model's ability of reproducing and explaining behaviour and on sensitivity testing. Similarly, the calibration of the model is based on data collection and informal interviews, but again no structured approach is proposed to carry out this important work.

The proposed use of the model in practice is based on the basic SD process of "what-if" policy analysis. The model is mainly used to explain the outcome of a project and to demonstrate that both the structure of the development process and co-ordination have a strong impact on the project outcome. No structured process is proposed to describe how a tailored model should be used in project management to support an on-going project.

This work provides valuable elements to the fields of SD modelling applied to project management. It suggests that multi-phase modelling of projects can be implemented in a flexible and practical manner, and it provides strong evidence of the benefits and importance of modelling explicitly co-ordination and the structure of the product development process. However, this work also has some limitations.

In the first place, it must be noted that it focuses essentially on proposing a generic structure that accounts for the structure of the product development process and for inter-phase co-ordination, and to demonstrate the relevance of these factors. Furthermore, the model is also presented as generic and hence with potential to represent any project. However, if a project manager wants to use this model in practice, two important requirements need to be addressed: how to tailor and validate the model for the specific project, and how to use it on a daily basis to support an on-going project. This is not explained in this work.

A potential weakness of this model is also the representation any development task using a common SD structure, regardless of the type of work carried out within the task. This implies, for example, that a software design task would be modelled by the same structure as a software testing task, while it is well know that these are rather different types of work. It can be argued that there can be important work-specific issues that should be represented in different ways in a SD structure. In order to prove the "hypothesis" of a valid single generic structure, a significant number of projects must be modelled. Cooper (1980, 1993) supports a similar principle, claiming the successful use of the PMMS (Pugh-Roberts Associates 1993) on many real projects. However, these applications have been implemented in a consulting environment and no research work has been published to support this claim. On the other hand, the complex model developed by Williams et al (1995) considers two major tasks of a product development process (design and build), which are modelled by rather different SD structures, perhaps suggesting that tasks specialised in the type of work are required to model a development project properly.

Other two areas of potential weakness of this model are: (i) the limitation on the number of tasks that can be linked into a network to represent the product development process; four tasks are considered in the case-study project and the generic model is designed for a maximum of five tasks. No discussion is provided regarding both practical and conceptual limitations of increasing this number; and (ii) the model's ability to consider that some project tasks (or phases) finish early in the life-cycle of the project. According to the generic structure proposed for the development tasks, a continuous need for rework of released errors to downstream tasks will be required. This prevents a task to be 100% completed practically until the end of the project. This gives the tasks a nature of continuous activities, as in the work of Cooper (1980) and in the PMMS (Pugh-Roberts Associates 1993). Most likely, continuous activities and precise phase-ends are both required to model a project according to reality – for example, this need has been recognised by the conceptual process models in the software industry (Rook). Therefore, a more detailed SD structure is probably required than the one presented in this work.

6.5.9 Pugh-Roberts Associates – the PMMS for modelling development programs.

The work reported earlier in Cooper (1980), developed at Pugh-Roberts Associates, was further refined by this consulting firm. This led to a generic framework for the development of SD project models, called the "Programme Management Modelling System" (PMMS) (Pugh-Roberts Associates 1993), which has been applied to many real projects over the last decades. The materials published about the PMMS are very restricted in terms of detail, for obvious commercial reasons. Although this framework does not provide an answer to many critical issues of SD project modelling, it can be considered as a major milestone towards the establishment of a structured process. The PMMS is the first attempt at developing a generic framework for the model development process.

Model development, structure and quantification

Developing a SD project model using the PMMS is not the same as developing a new model based on the generic SD approach. The PMMS framework provides new features to the model development process. It also restricts the process by imposing particular features over the model.

Conventionally, the SD process should start with a verbal description of the problem and with the definition of the model purpose. This description is then be converted to behaviour patterns over-time. Influence diagrams are then developed, where the problem is explained in terms of feedback processes. In the PMMS, this qualitative phase of influence diagramming is non-existent. Instead, a high-level feedback structure common to all projects is assumed. This structure was developed from numerous past developments, and is used to communicate with the Client. The main feedback processes considered are as follows:

- *work accomplishment* – this is the basic process of monitoring progress against the plan and of adjusting the workforce accordingly;
- *customer actions* – customer introducing changes increases the work remaining, reduces productivity and quality, and motivates staff to do "work out of sequence";
- *effects of staff hiring* – new staff brought into the project has lower experience level, thereby reducing productivity and quality. There are also training overheads and communications overheads due to team growth;
- *detrimental side-effects* – as work progress is slow, schedule pressure increases productivity but reduces quality and motivates work being done out of its natural sequence. Doing work out of sequence tends to reduce quality, and thereby work accomplished incorporates more undiscovered errors, which in turn contribute to the generation of even more errors. As rework is discovered schedule pressure increases even more;
- *impacts of upstream work on downstream work* – delays and poor quality of work from previous phases has serious impacts on the quality and work availability of the future phases.

This feedback structure identifies the main areas considered in the PMMS: project control through human resource allocation, effects of staff hiring and of team growth, effects of Client changes, effects of schedule pressure and of "work out of sequence", effects among the project phases. The key performance factors considered are productivity and work quality. The assumption is that these factors and effects are present in any project, and hence it is possible to consider a generic feedback structure at the core of any project model.

The PMMS is based on the principle that the structure of a project model can be built from elementary "building blocks". These blocks represent the main activities within the project, and can be interrelated according to various types of dependencies. These activities are performed continuously throughout the entire project life-cycle. Within each "building block", a particular type of work is performed. The building blocks considered in PMMS are: (1) design activity, (2) construction activity, (3) procurement activity, (4) testing activity, (5) staffing, and (6) program management. The basic principles underlying this "building block" framework are as follows:

- a project consists of a life-cycle of four main phases: a product is first designed (design); based on the design specification the materials required to build the product are acquired (procurement); the product is build (construction); and the product functionality is then tested (testing);
- human resources are the main drivers of work accomplishment, and should be controlled according to work progress. Different categories of human resources can work in a project;
- the whole project is controlled by program management activities, which monitor progress in the life-cycle, adjust the schedules, and allocate human resources to the project.

Although not specified by PMMS, this set of building blocks can be clearly divided into two main categories: design, construction, procurement and testing are engineering type of activities as they relate directly to product development. The blocks of program management and staffing represent management type of activities, and relate to the process of controlling the project. A PMMS project model consists of an assembly of these elementary building blocks. The generic feedback processes considered in the PMMS approach as described above are captured within and across these building blocks. At the core of the engineering type of building blocks is the rework cycle structure, as described in chapter 5 (Cooper 1993).

The way in which several building blocks are assembled to form a project model is not specified in detail by the PMMS. The authors argue that the first determinant is the level of detail in decomposing the project into activities, and this depends on the model purpose. The only rule specified in the PMMS brochure (Pugh-Roberts Associates 1993) is that a model should contain at least one of the building blocks specified above, except for "testing".

The model development process is proposed as a sequence of phases:

- *design* – in this stage the purpose of the model is clarified. The model architecture is defined as describing the activity-breakdown of the whole project into the appropriate level of detail. This is based on interviews a with managers and staff;
- *programming* – the building blocks specified in design are assembled to form the whole model. Preliminary calibration is carried out with the Client support, so that the model reproduces the basic project behaviour;
- *refinement* – detailed calibration is carried out so that the model is able to reproduce with accuracy the project reference mode of behaviour. This includes hard and soft behaviour patterns, based on managers' descriptions. Model calibration is based on parameter adjustment. A structured calibration process is mentioned but is not specified.
- *tests* – once the model is capable of recreating the reference mode of behaviour, it can be used for its purpose supporting "what-if" analysis of various project scenarios.

The authors claim the use of an "expert system" to guide the whole model development process, based on automated software routines. This system can help the inexperienced modeller to develop a PMMS project model ready to be used. The semi-automated nature of such system implies that it must incorporate a set of formal rules capable of "deciding" about:

- how the whole project is decomposed into a set of interrelated activities (i.e. activity-breakdown), and according to the "building bocks" available;
- the interrelationships between these activities and hence how the corresponding "building blocks" are to be assembled;
- the procedures for model calibration based on parameter adjustment, so that the model replicates accurately the reference mode of behaviour;
- a criteria to validate the model structure, and to validate the calibration process.

Whilst this constitutes the core of an "ideal" approach to SD project modelling, these formal rules have never been published. No further work claiming or confirming the actual existence of this system and tools has been published so far. Therefore, this system cannot be subjected to theoretical analysis and hence it is of little value for research purposes. It can only serve as an ideal, which probably applies to any modelling technique.

Model validation

Very little is described in the PMMS brochure (Pugh-Roberts Associates 1993) about model validation and how this process is conducted. No structured framework is presented. In the author's personal view, model validation in the PMMS approach must comply with the following requirements:

- each of the "building blocks" SD generic structures proposed must represent well the corresponding project activity in the real world. It must be capable of reproducing with accuracy the behaviour exhibited by that activity;
- the activity-breakdown of the project must represent well the real world structure of the project;
- the assembly of the various building blocks must capture well how the correspondent project activities are interrelated in the real world;
- the PMMS final model must be capable of replicating the various reference modes of project behaviour, in particular the project history (if available).

The particular ways in which these assumptions are tested and verified is a critical issue for model validation in the PMMS. These are not specified in the detail by the authors.

Model use

The range of uses proposed by the authors covers three main types of applications: diagnosis of past projects, analysis of on-going projects, and analysis of prospective projects. Depending on the type of application, slightly different approaches are implemented to use the model. They are all based on calibrating the model for a reference mode of behaviour, called the "base case", and then introducing changes in the model to reflect real world situations. "What-if" analysis can then be performed. The following applications are described in the PMMS:

- *past projects: contract claim* – the model is first calibrated to reproduce the past "problematic" behaviour as the base case. The Client disruptive actions are "removed" in the model calibration, and the two scenarios are compared. The difference quantifies the impacts of Client actions to be claimed;

- *on-going projects: change management, program management, division management* – the model is calibrated to reproduce the likely project outcome as the base case. Risks are then "introduced" in the model calibration, and the impacts are assessed. Potential management actions are also introduced in the model calibration and their performance in mitigating risks is assessed;
- *future projects: prospective bids* – the model is calibrated to reproduce a "similar" project. Problems that eventually occurred in that project are "removed" from the calibration so that a "problem free" scenario is achieved. The particular features of the prospective project are "introduced" in the calibration and the base case is achieved. "What-if" analysis is carried out to assess the impacts of possible risks.

Summary

In summary, the program management modelling system (PMMS) constitutes the first major attempt to develop a generic framework for the development process of SD project models. Implementing the PMMS approach differs in some important ways from developing a new SD project model by following the general steps of the SD process. The PMMS approach is based on two main premises: (1) all projects have in common a core feedback structure, and (2) projects can be decomposed into elementary sub-activities of work, which can be represented by generic SD sub-structures. Model conceptualisation in PMMS therefore focuses on decomposing the project into a set of interrelated activities at an appropriate level of detail, and linking these activities. Having specified the model structure in this way, the PMMS modelling process shifts to a second phase of parameter calibration and model validation. Here the model parameters are adjusted to reflect the specific project by reproducing its reference mode of behaviour. The authors claim a range of uses for PMMS project models which includes past projects, on-going, and future projects.

The major weakness of the PMMS approach is the lack of any published material describing in detail the required procedures to: (1) decompose the project into a set of interrelated activities, (2) specify the type of interrelationships between these activities; (3) assemble the SD "building blocks", (4) undertake parameter calibration so that the model reproduces a given reference mode of behaviour (according to a criteria of "goodness-of-fit"), and (5) to validate the model structure and calibration. Without this information, the PMMS framework provides some good ideas and a reference for an ideal tool but its research value is very limited.

6.5.10 Summary conclusions

The past SD project models reviewed in this section introduce important concepts for the establishment of a structured model development process. There are various elements which are common to most of these past developments, perhaps reflecting that some modelling issues need to be addressed in a specific way. For example, most developments exhibited a life-cycle process of phases, where a final validation phase focused on behaviour reproduction. The main characteristics of these past developments are summarised and compared in tables 6.2 through 6.6. This includes the overall development process followed, the model structure, model quantification, model validation and model use.

It is important to note that none of these developments attempted at establishing a generic and structured process for the development a SD project model. Most of the characteristics and underlying concepts presented in tables 6.2 through 6.6 were not made explicit by their authors. They have been deduced or uncovered for the purpose of this research. For example, the concept of "model architecture" referred to in the review of Cooper (1980) and of PMMS is not made explicit by this author. Some of the models presented are aimed at being generic models for certain industries (e.g. Abdel-Hamid 1990 for the software industry, Ford 1995 for product development projects). The PMMS is presented as a generic platform aimed at supporting the rapid development of a SD project model for large development programmes, within a consulting environments. The work of Richardson and Pugh (1981) focuses on bringing structure and discipline to the generic SD modelling process, where a project model is used as an example, providing valuable ideas. The work of Williams et al (1995) focuses on the importance of mixing soft and hard techniques in a modelling process, and provides a good example of how this can be done in practice. However, while these studies propose important ideas none of the developments proposes a generic and structured process for the development of a SD project model. Personal approaches were followed, more or less based on the generic SD modelling process.

The aim of this chapter is to propose a generic structured process for the development of a SD project model. For this purpose, the following sub-sections identify those elements present in these past developments which will be included in this process.

Model development

Looking at table 6.2, it is clear that in most cases a life-cycle of phases was followed. These phases were implemented sequentially but also in an iterative manner. In some cases, this life-cycle starts with a qualitative phase, aimed at identifying the feedback structure underlying an observed or known project behaviour. A high-level model design phase is present in the main practical studies, preceding a more detailed definition of the model structure. Although validation is carried throughout the whole development process (in particular, structural validation), in most cases the life-cycle finished with a validation phase, focused on behaviour reproduction and purpose suitability.

The use of influence diagrams (IDs) to conceptualise the project feedback structure, before the development of the quantitative simulation model, is not present in all studies. In some of the practical applications (Cooper 1980, Abdel-Hamid and Madnick 1991, Williams et al 1995, and Pugh-Roberts Associates 1993), influence diagrams played the role of providing a concise description of the key feedback loops incorporated in the project model. These diagrams were used to communicate with the Client, being perceived as a more effective means to describe the model's contents than the "level/rate" diagrams at the quantification level. However, this qualitative phase is different in some studies. For example, the qualitative phases in Richardson and Pugh (1981) and Williams et al (1995) are distinctive: the first includes the explicit dynamic definition of the project reference modes of behaviour, while the second introduces Cognitive Mapping as providing the required flexibility to initiate the development of the model's feedback structure from managers mental models.

Regarding the conceptualisation of the model's feedback structure, in most cases (and in particular in the practical applications), the authors have adopted a "process perspective", where the "physical processes" are the first elements of the system to be identified as flows of entities. In most cases, the development of the quantitative simulation model also followed a "top down" approach, starting with a high level definition of the model's main sub-systems, their contents in terms of areas of project activity (e.g. human resource management), and their inter-relationships.

In the practical applications to real projects, there were extensive efforts of data collection and of information gathering from managers and staff. This highlights two important issues: a SD project model tends to require numerical data which is not readily available from a typical project management information systems (in particular "soft" data); secondly, managers' informal judgement is essential to decide what to include in the model and the appropriate level of detail. The gathering of information from managers and review of the model took the form of meetings, interviews, questionnaires, and in some case workshops to present and discuss the modelling process. Another important aspect of the practical applications was the use of a prototype model in the beginning of the process. In general, prototyping is useful when the requirements of the product to be developed are not clear in the early stages. This typically happens with a SD model.

From the analysis of these past models, it is proposed in this research that the following features incorporated in a generic model development process:

(1) *life-cycle process* – a sequence of main phases comprising:
   - qualitative feedback analysis using IDs and graphs over time,
   - high level model design representing the model's main components,
   - detailed model design specifying the contents of the components,
   - data collection and information gathering,
   - quantification and test of prototype model,
   - quantification of full model,
   - model validation;

(2) *Iterative process* – although the process follows a sequence of main phases, it can cycle-back at any stage;

(3) *top-down approach* – "top-down" approach of project decomposition into sub-systems and sectors;

(4) *use of IDs* – to identify the feedback loops to be captured in the simulation model, and to communicate with the Client;
(5) *managers and staff involvement* – when the model is to be used in a real project, this is essential to identify the factors and feedback loops to be captured in the model, as well as to review the results produced by the model.

Model structure

Regarding model structure, it is important to analyse the level of detail adopted under three different perspectives:

- work breakdown into sub-tasks, and consequent breakdown into sub-budgets and intermediate schedules,
- the "physical processes" considered, through the specification of "entity-flows" and associated activities;
- range of managerial decisions considered,
- factors and feedback loops considered.

The earlier models by Roberts (1964, 1974, 1978) and Richardson and Pugh (1981) are highly aggregated and do not consider any breakdown. The project is modelled as a single task to be accomplished by a single team, within a single schedule and budget. The physical processes considered are the flows of work and of human resources, with the correspondent activities of work accomplishment and staff hiring / attrition. The range of managerial decisions is limited to the trade-off between extending the schedule and hiring more staff. Only few general effects are considered, like the impact of schedule pressure on productivity.

The model of Abdel-Hamid and Madnick (1991) also adopts a "single-task" perspective of the project, but decomposes the work flow into a dual-flow of tasks and associated errors. The activities considered are more detailed than just work accomplishment, including: development, QA, rework and testing. In comparison with the previous model, a major novel contribution of this work is the range of effects considered, which includes many "soft" and intangible factors which participate in many feedback loops (e.g. the impacts on productivity, error generation, and error rework, and the effects of error "re-generation", among others). The range of managerial decisions includes more complex project control policies, which are not present in previous models (e.g. adjusting the QA level).

The model by Williams et al (1995) decomposes the project into two major tasks of designing and construction, each with an individual schedule and budget. Within these two tasks, detailed activities of work accomplishment are considered some of which relate to rework. Unlike in previous models (except for Cooper 1980), the work flow cycles back to capture the rework phenomenon (i.e. flawed work tasks return to previous states in their life-cycle process-flow). Changes in product requirements are considered as an essential feature for the model purpose and this affects the life-cycle of the tasks process-flow. Various flows of human resources are considered to represent different staff categories working in the project.

The model for software projects presented in Lin and Levary (1989), though not published in detail, considers the breakdown of the project into a sequence of tasks according to the phases of the classic software development life-cycle model. Each of this phases has an individual schedule and budget. This model goes beyond the one by Abdel-Hamid and Madnick (1991), considering the occurrence of requirements changes throughout the life-cycle of the project.

The model by Ford (1995) provides a major step forward. The model is published in full and is described in great detail. The model considers a breakdown of the product development process into a network of inter-related tasks. These inter-relationships are modelled as dynamic dependencies which capture various inter-task effects. There are also internal dependencies within a task. At the core of each task a new generic SD structure is proposed to model any type of development process (regardless of industry and type of work). This structure considers a new activity no include din previous co-ordination due to rework related with other tasks (e.g. released errors found by downstream tasks, or errors detected from upstream tasks). This structure considers that tasks with errors cycle-back in order to be reworked. It also considered a co-flow of errors associated with the tasks. The other activities of work accomplishment considered are: base-work, QA and rework. The control decisions and policies captured in the model include schedule and resources adjustments and also a new policy for rework co-ordination. Control decisions are implemented at both task level and project level. The structures of work accomplishment within the tasks and for the control of the tasks are replicated and hence are exactly the same for all tasks. The control process at the project level inter-acts continuously with the control process at the task level.

Finally, the model by Cooper (1980) further refined into the PMMS (Pugh-Roberts Associates 1993) considers the breakdown of a project into a "network" of interrelated tasks as in Ford (1995). The implementation of these tasks can overlap and are interrelated according to technological dependencies (not specified by the author, but probably in a similar fashion as in the model by Ford 1995). At the core of each tasks, the "rework cycle" structure models the process of work accomplishment. This considers that work tasks cycle-back due to the generation of rework. These models also consider various types of staff categories. The "building block" approach assumes a breakdown of the management process into individual management activities, increasing the range of managerial control decisions considered. The feedback effects considered are intended to be generic to any project, and focus on: Client actions, work out of sequence, schedule pressure, and work quality. In terms of structure there is a strong similarity between the PMMS and the model from Ford (1995). The only key difference is the generic structure proposed to represent the process of work accomplishment within the tasks: the structure by Ford appears to be more sophisticated. Since neither the model by Cooper (1980) nor the PMMS have been published, a detailed comparison cannot be established.

Over-time, the structure of the project models developed has evolved towards an increasing degree of complexity. This is reflected in four main features of the models: (1) the breakdown of the project into sub-tasks, (2) the breakdown of the "physical process" of work accomplishment into several "entity-flows" and associated activities, (3) the range and detail of managerial decisions considered, and (4) the range and detail of factors and feedback effects captured in the model structure.

The breakdown of the project into sub-tasks has evolved from "single-task" models (Roberts 1964, Richardson and Pugh 1981, Abdel-Hamid and Madnick 1991) to the dis-aggregation into sequential major phases according to the product development life-cycle (Lin and Levary 1989, Williams et al 1995), and finally into a high-level network of interrelated tasks (Cooper 1980, Pugh-Roberts Associates 1993, Ford 1995). The level of project decomposition reflects not only a natural evolution towards more detail, but also the particular purpose of the model.

The breakdown into entity-flows evolved from a simple flows of work with two sates, "to be done" and "done" (Roberts 1964, Richardson and Pugh 1981), to more detailed work flows with several intermediate states (Williams et al 1995, Ford 1995, Pugh-Roberts Associates 1993), and in some cases co-flows of work and errors (Abdel-Hamid and Madnick 1991, Ford 1995). The range of activities captured also increased from the two basic activities of "base-work" and "rework" (Cooper 1980, Richardson and Pugh 1981) to include QA (Abdel-Hamid and Madnick 1991), co-ordination (Ford 1995) and others (Williams et al). The entity-flows of human resources employed in the project also evolved from simple single-state flows (Roberts 1964, Richardson and Pugh 1981), to more detailed flows with several states which consider training requirements and staff experience level. The single entity-flow of resources also evolved towards the decomposition into sub-flows, to consider various types of human resources employed in the project.

The initial models considered the management process as the simple trade-off between the two decisions of delaying the project schedule and hiring more staff into the project (Roberts 1964, Richardson and Pugh 1981). This has evolved to consider further management actions like changing the QA level, adjusting the project scope, Client actions, and multi-task and multi-resource control (i.e. individual control of various tasks and resources). With the decomposition of the project into sub-tasks, and the with decomposition of the work and resources entity-flows into sub-flows with more detailed life-cycles, the basic policies and control decisions of schedule adjustment and human resource management could also be dis-aggregated into more detail (e.g. changing the staff training period, changing the QA level, or adjusting intermediate schedules and sub-budgets).

Finally, the factors, the causal relationships and the feedback loops captured in the models has also evolved towards increased detail. Unfortunately, a detailed description of all the models reviewed is not available in the literature (except for Roberts 1964, Richardson and Pugh 1981, Abdel-Hamid and Madnick 1991, and Ford 1995). Furthermore, similar factors and effects can be captured in a model in a more or less explicit manner, and at different levels of detail (e.g. the cause-effect relationship between "work progress" and "productivity" may encapsulate more or less effects, like "learning", "work difficulty" and others). In practice, this often makes it easy for authors to claim that certain effects were captured in the model, as aggregated into certain relationships. In many cases, these factors and effects are also strongly inter-related and hence a clear distinction is difficult to establish. A comprehensive list of the factors, critical cause-effect relationships and feedback loops considered in each of these past models could prove useful in identifying a core "generic" feedback structure (as attempted by the PMMS framework). However, while the inclusion of factors and effects must depend on their "real" existence in the system, this must also be determined by their relevance regarding the model purpose.

Overall, in the studies discussed above there was an attempt in the initials models to include generic factors and effects at a high level of aggregation (the models were intended to be generic and illustrative). The following models naturally included these same factors and effects, having considered them in more detail, and also considered new ones. For example, the model by Lin and Levary (1989) considers requirements changes and claims more detail than the one by Abdel-Hamid and Madnick (1991) in representing the software process. In the practical developments by Cooper (1980) and Williams et al 1995) the factor "relevance" was more determinant, but these models considered even more detailed feedback structures. For example, the PMMS approach considers generic feedback structures within each of the building blocks proposed, and feedback effects amongst these blocks which can be adjusted for the specific project.

In summary, from the analysis of these past models, it is proposed that the structure of a SD project model should be conceptualised under four different perspectives of project breakdown. These perspectives are interrelated and can be used to guide the process of model design:

- *task breakdown* – the project should be decomposed into several interrelated sub-tasks, reflecting the structure of the specific project being modelled. While the level of decomposition is necessarily more aggregated than in an operational PERT/CPM network, the various dependencies between the project tasks should also be considered leading to a network;

- *entity-flow breakdown* – the real "physical processes" of the project should be decomposed into flows of entities (called "entity-flows"), which have a well defined life-cycle. The two critical "physical processes" to be considered in a project are (i) work accomplishment and (ii) resource allocation to the project;
- *breakdown of management decisions* – the management process of project control should be decomposed according to the breakdown of tasks and of entity-flows, considered above. The range of managerial decisions that can be considered in the model is determined by these two levels of project breakdown;
- *feedback loop breakdown* – the feedback structure captured by the project model will include various factors, cause-effect relationships and feedback loops. Some of these elements will be local to the individual sub-tasks and other will cross various tasks. Feedback loops can be captured in the model at different levels of aggregation. Some can take place within the tasks while others can cross several tasks. They should be considered according to the three levels of structural breakdown above.

Model quantification

In general, after the model structure has been defined the next step in the model development process is the critical issue of quantification. Model quantification can be analysed under different perspectives:

- the two different problems:
  - (i) the structuring of mathematical equations to quantify causal relationships,
  - (ii) the assignment of parameter values;
- the sources to obtain the data and information required for the quantification;
- the types of relationships and parameters in the model and the implications for quantification;
- definition of ordinal scales for intangible variables or parameters;
- the two possible scenarios:
  - (i) direct deduction from numerical data available and logical reasoning,
  - (ii) need for empirical reasoning using numerical data and subjective information;

- structure of the reasoning processes in scenario (ii) above. Possible classification of scenarios and development of specialised quantification techniques.

As identified above, quantifying a SD model requires two different types of quantification: defining the structure for the mathematical equations that quantify relationships, and assigning values to parameters. These situations can be more or less difficult to handle, depending on: (1) the types of variables involved, and (2) on the data and information available. In the models discussed above, the main sources of data and information used were:

- extensive data collection carried out in the real project;
- related literature;
- management and staff expert judgement (interviews, questionnaires, workshops, presentations, etc.)

The use of these sources reflects current practise in System Dynamics modelling (Richardson and Pugh 1981, Coyle 1996).

Assigning parameter values raises the problem of whether the parameter represents an unmeasured or intangible factor of the real world. In the latter case, data will never be available from the real world and hence some form of scale and estimating are required – an ordinal scale is required to define the possible values for the parameter. Defining the mathematical structure of an equation also raises the problem of whether that equation involves unmeasured or intangible variables, and whether the relationship between the variables in the real world is not well known. In these cases some form of reasoning is required to infer or to deduce the structure of the equation.

Depending on the type of relationships and parameters, two different scenarios can be considered: (i) the structure of the equation or the parameter value can be logically deduced from knowledge about the real system; and (ii) no logical deduction is possible and some sort of empirical reasoning is required to estimate or infer the parameter value or equation structure – the terms "hard" and "soft" are often used to refer to these two situations.

Scenario (ii) typically occurs when an unmeasured or intangible factor is present, or when the relationship in the real world is not known. In all the SD projects discussed in this chapter, all authors recognise this problem. The most common solution adopted was to develop an intelligent "guess", based on an empirical rationale. The three sources of information referred above were used to support this process. The studies reported in Roberts (1964), Richardson and Pugh (1981), Abdel-Hamid and Madnick (1991) and Ford (1995) provide detailed descriptions of rationales developed to quantify parameters and relationships. However, no attempt is made to classify the possible situations and to further propose structured methods to develop the "intelligent guess". The PMMS approach adds to the three sources of information referred above, the vast number of real projects to which PMMS models were calibrated to reproduce their behaviours.

The main conclusion from these studies is that the quantification process still very much an "art", where empirical reasoning and past experiences are essential elements. There has been some attempts to structure the model quantification process (e.g. Coyle 1996), but no standard structured method is yet available. The general recommendation, as a good modelling practice, is to document the quantification process stating explicitly the rationales developed and the sources of information used for each variable and parameter in the model.

Model validation

Model validation in System Dynamics was previously discussed in chapter 2. Model validation rests on two main requirements: consistency with the real system, and suitability to fit a given purpose. Both model consistency and suitability cannot be demonstrated scientifically, and hence the validation process rests on the use confidence tests (Forrester and Senge 1980).

In general, it is recognised by SD researchers and practitioners that validation should be a continuous exercise throughout the whole model development process (Lane). However, there has also been recent arguments in favour of a final phase of "formal validation", after the model has been conceptualised and quantified. In this final phase, confidence tests are carried out more intensively (Barlas 1996).

In the studies discussed above, the validation process has been undertaken in this way (in a more or less structured manner). The lack of detailed documentation on how the model validation process was conducted in most applications is a major problem in System Dynamics (Peterson and Eberlein 1994). In the past models reviewed, validation has been primarily based on two major requirements:

- *behaviour reproduction* – the model's ability to reproduce a reference mode of behaviour. This can be the historical behaviour of a real project, or expected general patterns;
- *validity of calibration* – providing acceptable evidence that the model is calibrated with parameter values or equations, as these occur in the real system.

Structural validation is even more important than behaviour reproduction and parameter calibration. In these past models, this has been mainly based on logical rationales developed during model conceptualisation. Whenever a model did not satisfy behaviour reproduction or calibration validity, then its structure was revised and the underlying rationales were reformulated.

The procedures followed to ensure model validation were based on personal approaches, by using the well known framework of confidence tests from Forrester and Senge (1980). It is further suggested in this research that the validation of a SD project model must address four critical areas:

(1) the model's ability to reproduce the project history, if available;
(2) the model's ability to reproduce generic behaviour patterns within "familiar" scenarios;
(3) validity of parameter calibration, by corresponding to the real world;
(4) structural validity, by corresponding to the real world.

These areas were addressed in the studies discussed above in the following manner:

- *reproduction of project history* – implemented formally through statistical tests of "goodness of fit" (Sterman 1984), or more informally by checking the match of: (i) final results of cumulative patterns (e.g. total effort), (ii) general shape of the behaviour patterns;

- *reproduction of generic behaviour* – informally by checking how well the model matches "well known" project behaviour patterns, within typical scenarios (e.g. reproduction of the "Raleigh curve" for a software project). Turing-type of tests were used to ensure similarity between the behaviour produced by the model and the behaviour of real projects;
- *validity of calibration* – providing a logical or empirical rationale as well as evidence to support parameter values and the structure of the equations. Informally, by checking the model's ability to provide plausible explanations for the produced behaviour. Formally, by imposing ranges of validity for parameter values and "axioms" of consistency among the values of several parameter;
- *structural validity* – development of rationales to justify the logical structure and level of detail assumed. These rationales were supported by managers' judgement. Implementation of "extreme- condition" tests to check the consistency of the model's response to extreme scenarios. Testing and comparing performance of structural alternatives. Testing decision-making sub-structures, by replacing the model's decision roles by real managers.

Based on this review, it is argued in this research that a structured approach to model development must include a basic structured framework to address these critical areas of validation.

Model use

Using the model consists basically in calibrating the model to reproduce a "base-case" upon which alternatives are tested through "what-if" analysis. In cases of post mortem diagnosis, like in dispute resolution, the "base-case" is defined as the disrupted scenario which occurred in reality. In cases of fictitious or prospective projects, the base-case is generally the "problem free" scenario, where the impacts of risks and disturbances are tested.

The "what-if" analysis can address four categories of performance factors:

- external risks, out of managerial control (e.g. staff shortage);
- internal risks, out of managerial control (e.g. product complexity);
- changes in the product development process within managerial control (e.g. changing the level of concurrence);

- changes in the management control process (e.g. policies).

The model development process must ensure that these basic "functional requirements" of a project model are satisfied by the model structure. This is, a proper SD project model must be able to support these four types of analysis.

Selected elements for a structure method

In this section, eight of the most relevant past developments of SD project models were reviewed an analysed. From this analysis, the following principles have been identified to incorporate the structured generic model development method proposed in this chapter:

(1) the model development process should follow a life-cycle of phases, allowing for iterations, it and should be based on a top-down approach to model conceptualisation. While this process is aimed at developing a quantitative simulation model, it should include the use of qualitative IDs to support the modelling process and as the basis to communicate the model with the Client;

(2) the top-down approach to model conceptualisation should consider a breakdown of the project system under four main perspectives: work tasks, flows of entities, management decisions, and feedback loop decomposition;

(3) the quantification of the model should be supported by data collection, data and information available in related literature and on management expert judgement. The quantification of subjective parameters and relationships should be supported by empirical rationales and by past experience where available. These quantifications should be documented in the model regarding the source of information and rationale assumed;

(4) while the use of the model depends primarily on the purpose of its application, the development process musty ensure that some basic types of use are guaranteed. These include assessing external and internal risks, changes in the product development process and changed in the project management control process.

Based on these principles, the SYDPIM Model Development Method is proposed in the next section.

6.6 The SYDPIM Model Development Method

6.6.1 Rationale: a life-cycle approach

The life-cycle of a product development process usually comprises four main phases:

(1) *specification of the product requirements* – rigorous statement of what the product is supposed to be and to do;
(2) *design* – abstract representation of the product in order to meet the requirements;
(3) *development* – "physical" construction of the product according to the design;
(4) *testing* –verification that the product meets the requirements.

Because products are often complex a "top down" approach is normally used to decompose the product into sub- components. These are first developed separately and are later integrated into a single system. This decomposition usually takes place in the design phase of the development life-cycle. In this phase, a high level design is developed to specify the product sub-components and their interrelationships. The individual components are then subjected separately to a more detailed design. In this way, the design phase can be divided into two sub-phases: the high level design of the whole product, and the detailed design of the individual components. Once the individual components are developed according to the design, they are tested separately against some expected individual functionality. Once these tests are passed, these components enter an integration phase where they are assembled into a single product system which is then further tested as whole to meet the requirements. This is the general logic of the systems development life-cycle, which is applied in various fields.

While this logic advocates the steady implementation of a sequence of phases, in the real world these phases sometimes need to overlap (e.g. concurrent development). Sometimes, the needs to iterate several times. However, iteration does not necessarily imply that the work of a certain phase have to be repeated in full. Instead, it generally implies that some of the activities that take place within that phase continue to be accomplished beyond the completion of the phase. Generally, a phase is considered as complete when all of its deliverables are perceived as

100% complete and are passed onto the next phase in the life-cycle. These deliverables may then be changed and reworked, even though the phase has already been complete.

It is proposed in this research that a similar life-cycle approach can be applied to the development process of a SD project model. This is an adequate approach for two main reasons: a project model can be complex (regardless of size), and hence it is an appropriate strategy to break it down into major sub-systems (as adopted in most of the SD past developments reviewed). Secondly, the structured development process proposed in the SYSPIM development method adopts a "building-block" approach. This requires the project to be decomposed into a network of tasks, which will be modelled by specialised sub-models and are integrated to form the whole model. As just discussed, the life-cycle approach supports this process of decomposition followed by integration.

The life-cycle proposed for the SYDPIM Model Development Method is based on the following principles:

- *sequence of phases* – the process consists of a sequence of phases with precise phase start and end points;
- *deliverables* – a the end of each phase is a milestone where some sort of deliverable is passed onto the following phase;
- *continuous activities* –there is a set of modelling activities which take place continuously throughout the whole process;
- *activities relate to phases* –the modelling work in each phase is dominated by a certain modelling activity. This activity will absorb the majority of the effort spent in the phase. The other activities that take place in the phase support this main activity;

The last principle is particularly important. For example, it allows for validation to be carried out throughout the whole development process as well as for a validation phase to be placed at the end of the process. Similarly, once the model has been designed and quantification starts, the designs can be still reviewed in later phases by the continuous design activity.

The application of this life-cycle to the SD modelling process must be able to cope with important modelling issues. One aspect of particular importance is the fact that the model design is likely to be unstable throughout the development process, due to the poor understanding about the project system in the beginning of the modelling process. One of the aims of a model is precisely to improve this understanding, and this is particularly true for a SD model. As it will be seen, by "model design" it is here meant some form of conceptualised structure for the model, prior to quantification in a "level/rate" diagram. A possible solution to counter this difficulty is the use of a "prototyping" approach, as applied in some of the past developments just reviewed (e.g. Cooper 1980, Abdel-Hamid and Madnick 1991). Developing a prototype model delivers an initial model at low "cost". This model helps the modeller and the user to clarify what the final model is really supposed incorporate. Another possible solution to cope with an unstable design is to adopt an incremental approach. This consists in developing the model in increments of increased scope – an evolutionary process. Although this approach is not reported explicitly in any of the past SD models reviewed, this may prove an useful approach. The life-cycle process here proposed does not make explicit the use of prototyping and of incremental development, but it accommodates both approaches. It can be implemented within an initial prototyping phase, or within each of the increments.

A structured life-cycle process could also have an undesirable impact on learning. Learning is based on the continuous interaction and overlap between model conceptualisation, quantification, and experimentation. The life-cycle approach advocates a stable model structure. This can lead to a poor interaction among these three activities. While some SD researchers and practitioners would argue fiercely under this line of thought, others consider that the major gains are to be achieved after the model structure has been conceptualised, via structured experimentation like optimisation (Coyle 1996). It is not the purpose of this research to provide an in-depth discussion of this issue. Nevertheless, some of the major past applications here discussed indicate that the development of robust and stable SD project models was essential because the usefulness of the model was gained after the it has been developed (e.g. PMMS applications, Cooper 1980, Williams et al 1995, Abdel-Hamid and Madnick 1991, Ford 1995). These gains include both, the results produced by the model and the learning achieved through repeated experimentation where the core structure of the model was not changed. Therefore, it can be argued that in many applications in the real world the usefulness of a SD project model can be found on its "post-development" application. Furthermore, the potential for learning throughout the model development process is not overlooked by the proposed life-cycle approach: iteration, continuous activities and incremental development, all provide the basis for an effective process of "modelling for learning".

6.6.2 Overview of the SYDPIM development life-cycle

As described in chapter 2, the generic SD process starts with an initial qualitative phase of influence diagramming. This phase is extremely important to clarify mental models and the understanding about the system. This initial qualitative phase is also essential to ensure that creativity is brought into the modelling process. The SYDPIM Model Development Method proposes that the modelling process starts with an initial phase of causal analysis, prior to the specification of the model's formal requirements. This phase focuses on identifying the problems and issues to be addressed in the model. It also provides an initial description of the feedback structure of the project system, which will be captured in the quantitative model.

The phases and stages of the development the life-cycle of the SYDPIM method are proposed as follows:

(A) Model design

- (0) *causal analysis* – identification and definition of the core feedback structure to be captured in the model;
- (1) *requirements definition* – range of analysis that the model must cover and results it must produce;
- (2) *formal design* – model architecture, identifying the model sub-components and their interrelationships. Identification of generic specialised SD structures (sub-models) required to model these components. Specification of the interrelationships between the components and of other characteristics.

(B) Model implementation and validation

- (3) *component development and testing* – level-rate diagramming and quantification of generic specialised sub-models. Tailoring of these specialised sub-models for the specific project components. Validation tests of these individual tailored sub-models;

(4) *model integration and testing* – integration of sub-models according to the model architecture and the model design. Basic calibration and testing of full model;

(5) *final model validation* – validation tests to ensure that the full model is consistent with the requirements, designs, and managers' mental models.

Overall, six major stages are proposed for the development life-cycle, grouped into the two major phases of (A) design and (B) implementation and validation. This division between design and implementation is a very important feature of the SYDPIM model development process.

The design phase focuses on the *specification* of all the required characteristics of the SD model, in respect to its purpose. These include, for example, the explicit definition of feedback loops to be captured in the model and the range of managerial decisions that the model will consider. The design phase is aimed at ensuring that the "right model" will be developed.

The implementation and validation phase focus on the "coding" of the simulation model into a structure and equations. This model will have to satisfy all the characteristics specified in the design, in a way consistent with the real world. For example, the model may have to consider scheduling decisions because this is a requirement specified in the design. However, this alone is not sufficient: the mathematical formulation of these decisions will have to be represent the actual way in which managers generate these decisions in the real project. Implementation and validation are considered together in this phase for an important reason: while there is a final validation phase, validation takes place throughout implementation.

The logical sequence proposed in this life-cycle does not imply that the phases do not overlap. It is assumed that each phase will be considered as complete once its internal deliverables are complete for the first time and passed onto the next phase. These deliverables will then be used, reworked, and updated throughout the whole life-cycle. The various modelling activities will occur with more intensity within a particular phase, but will extend until the end of the development process – this is according to the general life-cycle approach described above.

Although the SYDPIM Model Development Method is intended to be generic, it is specialised in development type of projects. It is assumed that the project model to be developed is aimed at capturing both the engineering process of product development, and the management process of project control. The engineering process includes development work carried out within the prime-contractor organisation as well as sub-contracted work, if necessary. Likewise, the management process may also capture control decisions being undertaken in different organisations involved in the project, including sub-contracting management decisions. The principles behind the method are generic and can be transferred to other types of projects.

The phases and stages of the development life-cycle of the SYDPIM method are now described separately.

6.6.3 Phase A – Model design

The model design phase comprises the three main stages of causal analysis, requirements definition and formal design, as shown in figure 6.3 below.

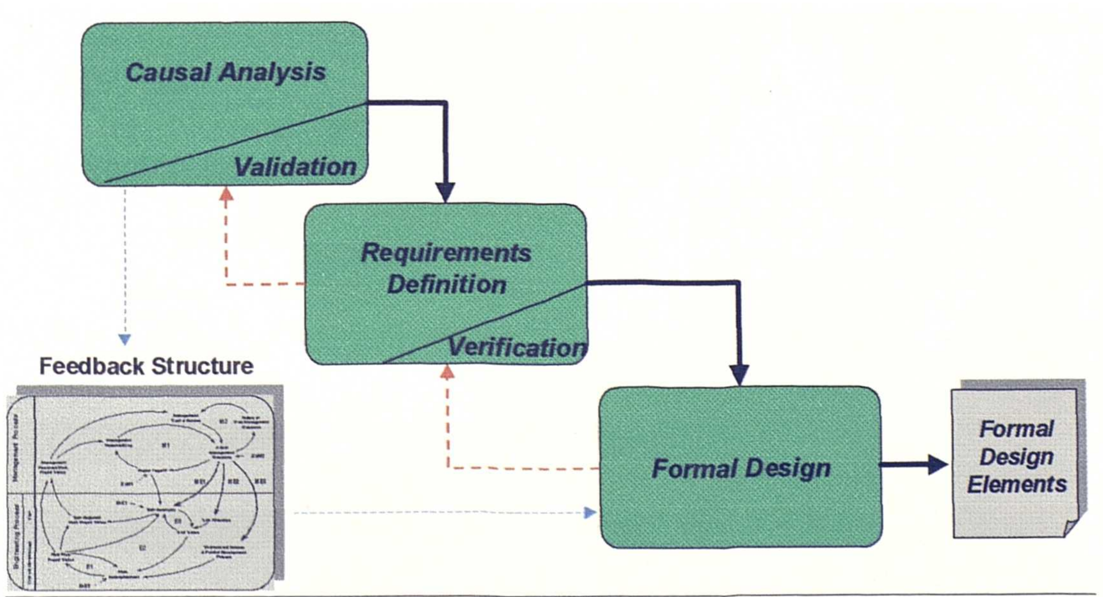


Figure 6.3 – Overview of the SYDPIM model design phase

Each design stage produces its won deliverables to the following stages and phases in the process. The two main deliverables of this phase are the description of the project feedback structure in the form of IDs and a set of formal design elements. The feedback structure is produced in the initial stage of causal analysis and the formal design elements are produced in the formal design stage. Both deliverables are very important as they will used to support the following phase of model implementation and validation. The feedback structure is also an essential input to the formal design stage within the design phase. The requirements definition stage also produces a deliverable in the forma of a brief document specifying the requirements that the model must meet (not show explicitly in figure 6.3).

The causal analysis stage ends with a validation process to ensure that the appropriate feedback structure was developed. The requirements definition also ends with a verification process to ensure that the users agree on the analyses that the model must be able to perform. The formal design stage is comprised of sub-steps which will be described in more detail. As shown by the dotted arrows, the process iterative and hence it can cycle-back to previous stages. This does not mean that the whole past stage will be implemented again. A stage is revisited to implement changes which may result from developments in the later stages. When this happens, the need for changes is verified within the stage, and the required changes are implemented and verified before the process proceeds.

These three stages of model design are now described separately in more detail.

Stage 0 – Causal analysis: conceptualisation of the project feedback structure

<u>Overview</u>

This phase is dedicated to the causal analysis of the specific project to be modelled (or type of projects). The main output from this phase is a set of influence diagrams specifying the feedback structure of the project. The conceptualisation of the this feedback structure requires the modeller to take important modelling decisions. The following elements need to be specified:

- the boundaries to be imposed in the model;
- the elements and factors to be explicitly considered and captured in the model;

- the cause-effect relationships to be explicitly considered and captured in the model;
- the level of detail to be considered in breaking down these elements, factors, and relationships.

Validation is important at this early stage in the model development process. Therefore, the feedback structure to be conceptualised is subjected to qualitative validation based on a set of project behaviour modes.

There is no formal method available to conceptualise the feedback structure of a dynamic system. This issue has been the topic of interesting debate. In the course of a very interesting debate in the email-group of the System Dynamics society, one author has argued that model conceptualisation is essentially a creative process and hence the way in which a credible feedback model takes shape "...must not be unduly restricted." Still (he argued), "...we all share a discipline for discovering the essence of dynamic complexity in practical situations". He then discussed the advantages a policy-oriented perspective to the conceptualisation process where policies are identified first, followed by the information used by these policies, and finally the effects of these policies on the system are modelled (see Morecroft 1984). Another interesting contribution from another author argued that the starting point for model conceptualisation can be any one of the five elements that make up organisational systems: processes, policies, boundaries, information, and delays. This author then explained how starting from one of these elements would then lead the conceptualisation process to spread onto the others (see Wolstenholme 1994).

The SYDPIM principles

A premise of the SYDPIM method is therefore that the conceptualisation phase can be structured, and hence some discipline can be brought into the process. Developing the feedback structure of a project for the first time should certainly be a creative process. It should not to be overly restricted by pre-conceived principles and knowledge about the system. However, important lessons learned from previous modelling efforts should not also be overlooked. A hybrid approach is proposed in SYDPIM, where a structured framework is used to guide the conceptualisation process. This framework is based on the following principles:

(1) *hierarchy of IDs* – the project feedback structure is represented through a set of Ids, which capture the key feedback loops and exogenous factors. These IDs capture the various areas of the project system at different levels of detail. A top-level ID covers the whole project, while progressively more detailed sub-IDs focus on the individual areas of the project. This hierarchy of IDs can have as many levels of decomposition as required, depending on the perceived need to address and describe the relevant issues in the project. The bottom-level of detail does not need to match the equation level in the simulation model; for complex models that should not be the case to prevent excessive complexity. Coyle (1996) proposes a hierarchy of IDs (referred to as "cone") with four levels of decomposition, the last level matching the equation level. However, in this work the SD models presented are generally quantified at a very high level of aggregation;

(2) *use of cognitive mapping prior to the development of IDs* – the cognitive mapping technique (Eden 1994) can be a very useful to support the development influence diagrams. Cognitive maps are typically developed at higher levels of aggregation and are less formal than influence diagrams. They provide the modeller with more flexibility in the initial steps of the conceptualisation process. The practical experience reported in Williams et al (1995) and Ackerman et all (1997) suggests that cognitive maps facilitate the development of IDs. The conceptualisation process is initiated with the development of cognitive maps, and then moves to influence diagramming, iterating several times as required;

(3) *generic core feedback structure* -- although projects are unique, they all share common features. The analysis of the past developments discussed suggests that most projects share a common core feedback structure. This implies that generic factors, causal relationships, and feedback loops are present in any project system. In these past developments, the author has identified some of these loops. This common feedback structure can be represented at an abstract level, and can be further used as a formal classification framework, helping to identify the feedback structure of the specific project;

(4) *management-engineering dual view of a project system* – the project system should be considered as a dual process of management and engineering interacting continuously over time. The engineering process captures the product development process, while the management process captures the

managerial actions of project control. This view helps to clarify the types of feedback loops that can be found in a project system. SYDPIM proposes that they can be logically grouped into three main types:

- feedback loops that take place within the management process;
- feedback loops that take place within the engineering process;
- feedback loops that take place between management and engineering.

The SYDPIM classification framework

Based on these principles, the SYDPIM method proposes a generic classification framework, which is used to conceptualise the project feedback structure. This framework provides a generic feedback structure for a project, as shown in figure 6.4. The project system is conceptualised according to the dual view of management and engineering. Each variable, causal relationship, and feedback loop in this generic framework represents an *class* or *type* of variables, exogenous factors, relationships and loops. In the actual feedback structure of a specific project, one or more elements will fall in each of these classes. For example, the class of variables "project targets" in the generic framework can correspond to the overall project planned schedule as well as to the planned schedule of an individual project task.

Three main types of loops are proposed in this generic framework: M (management), E (engineering), or ME (management-engineering); two types of exogenous factors are considered identified as ExM (exogenous-management) and ExE (exogenous-engineering).

The classes of feedback loops are discussed first. Two classes of loops are considered in type M:

- M1 – this class represents the managerial response to a gap between the project status and the project targets, which adjusts the target towards the likely outcome. For example, when the project status indicates that work progress is behind the target schedule management responds to this gap by extending the planned schedule. Alternatively, management may increase the planned budget or decrease the work scope.

- M2 – this class represents the important feedback phenomenon according to which future management decisions depend upon decisions already taken in the past. In the real world, management control policies are dynamic. In project control, managers respond not only to the perceived gap and current project status, but they also respond on the basis of the "history" of past decisions. As an example, if in the early stages of the project the work progress is behind schedule, managers may choose to delay intermediate schedules a few times. However, if after a while the delays persist, managers will probably believe that delaying intermediate schedules is no longer the right solution, and another type of control action is likely to be pursued. This feedback effect also captures the subjective human issue of managerial continuous learning, from decisions taken in the real world.

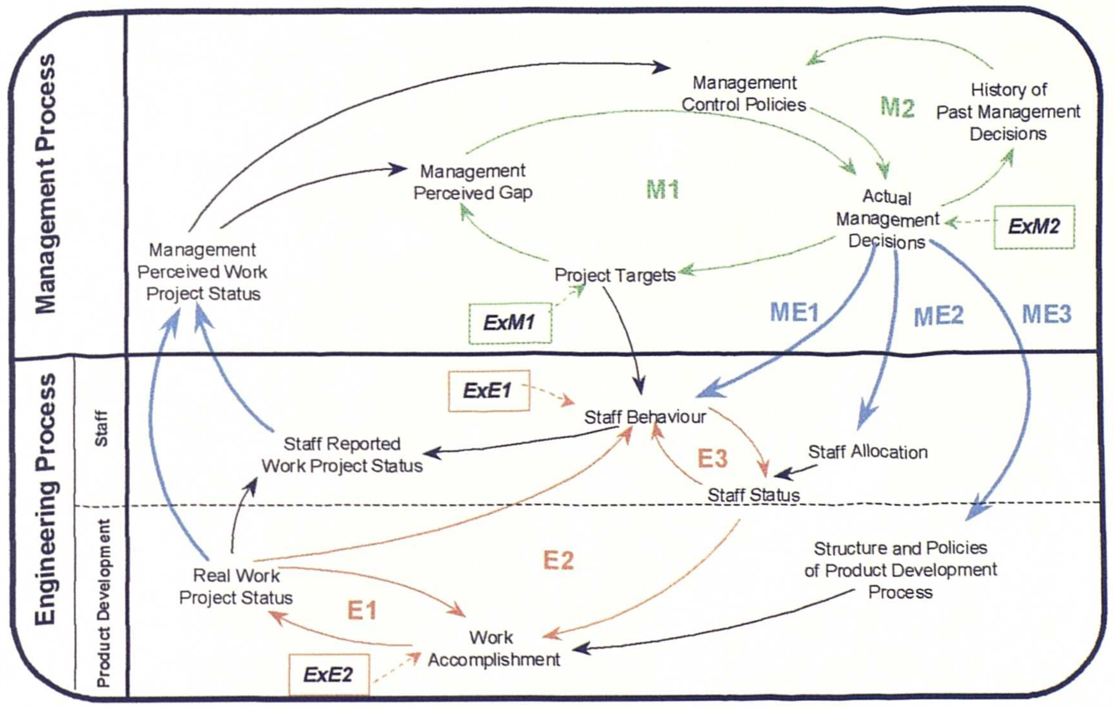


Figure 6.4 – A generic classification framework for conceptualising the feedback structure of a project management system

It should be noted that this framework does not suggest that only two feedback loops take place within the management process of a project. Instead, this classification framework imposes that *any* feedback loop taking place within the management process will fall in one of these two categories.

Three classes of feedback loops are identified in the engineering process (type E):

- E1 – this class represents the "natural" feedback effects of work progress on work accomplishment. A typical example are the effects of "technical feasibility": the more work is completed, the more the amount of work that can be started (e.g. the more the design is completed the more code that can be developed). Another example, are the error "re-generation" effects (Abdel-Hamid and Madnick 1991): (1) the more errors are generated, the higher the number of undiscovered errors and hence the stronger the potential for generation of further errors; (2) on the other hand, as the work progresses, undetected errors become "passive" thereby reducing the potential for an increased error generation. In these examples, the evolution of the work in the project affects work accomplishment, which in turn affects work progress;
- E2 – this class also represents the "natural" effects of work progress on work accomplishment, but this time through intermediate effects on staff behaviour -- technical staff is here considered as part of the engineering process. The staff naturally adjusts their behaviour in reaction to work progress. This leads to changes in the staff status, which in turn affect work accomplishment. This is a very important class of feedback loops, since staff behaviour is a crucial factor for project performance. Typical examples include staff reactions to schedule delays, like working non-reported over-time or reducing QA efforts without reporting to managers. Another very important effect is the "natural" learning process: as work progresses the staff become more familiarised and will work faster, making less mistakes, and reworking errors more efficiently;
- E3 – this class represents the important impacts of changes in staff behaviour. The concept "staff status" incorporates any characteristic of the workforce in the project, like experience level, staff in training, staff exhaustion, and others. A good example is presented in Abdel-Hamid and Madnick (1991): under schedule pressure, staff reacts by working extra hours; over-time staff will then become exhausted, making more mistakes and eventually reducing the number of working hours. Another example is when new staff is brought into the project. The experienced staff will need to dedicate some amount of their effort training this new staff. As training is accomplished, the training requirements tend to reduce. This is also an important class of feedback loops, because many of the project dynamics are of human nature and hence relate to "intra-staff" feedback effects.

The third type of feedback loops identified represents on of the basic principles underlying the System Dynamics approach: social systems are managed systems, controlled through a continuous information feedback process that takes place between management and the "physical! processes. The "physical" process of a project system is here considered as the engineering process of product development. As work progresses along the life-cycle, managers monitor progress, compare with the current targets, and take control actions aimed at ensuring that work progress meets these targets. However, it is also known that management actions often produce undesired effects which may worsen the original problem. Therefore, positive feedback loops may emerge, resulting not in control but rather in vicious circles. Hence this type of loops include both managerial attempts to achieve control, as well as their secondary undesired effects. These types of loops are identified in figure 6.4 as ME (management-engineering). Any of these loops will cross both processes of management and engineering. The specific individual classes within this type of loops can be identified by tracing all possible closed loops that cross the two processes. In figure 6.4 a total of 24 loops can be identified. It would be very time-consuming to discuss each of these classes of loops individually. Nevertheless, each represents a class of potential feedback loops in a project system (it is therefore not surprising that the feedback structure of a real project may contain hundreds of loops). Furthermore, while some of these individual classes are straightforward to understand, others do not represent an obvious feedback process. Therefore, for the sake of clarity this type of loops has been split into three more general classes, termed as ME1, ME2 and ME3. They represent the three generic ways through which ME loops can take place as follows:

- ME1 – this class represents the feedback loops that result from managers taking control actions aimed at affecting the staff behaviour. A typical example are the "penalty-reward" incentives, which are known to affect staff productivity. Other examples are the explicit use of over-time, or reducing training efforts. The direct effects on staff behaviour propagate in many ways throughout the feedback structure within the engineering process, eventually affecting work progress and thereby feeding back into the management process;
- ME2 – this class represents the feedback loops that result from managers making changes to the staff allocation to the project or to the staff scheduling among the project tasks. A typical example is the recruiting of more staff into the project when the work is perceived behind schedule. Another example is

scheduling the man-power available to those tasks perceived to be more delayed;

- ME3 – this class represents the feedback loops that result from managers taking actions to change or re-structure the product development process. This may take the form of radical changes in the life-cycle structure, like removing or adding new phases, increasing concurrency among tasks, or more simple changes like reducing the amount of effort allocated to QA activities or extending the planned rework period (i.e. errors waiting to be reworked for a longer period of time).

Overall, this framework suggests eight different classes of feedback loops. Any specific feedback loop of a project system will fall in one of these categories.

This framework can also be used in the same to classify the factors and causal relationships in the model. Each concept in figure 6.2 represents an aggregate class of factors in the feedback structure of a project system. For example, the concept "Management Perceived Gap" can be a schedule delay, a budget over-run or low system quality. Likewise, "work accomplishment" can represent the development of code, rework of errors, integration of system components, review of documents, or any other activity of product development. Similarly, each arrow in the diagram represents an aggregate class of cause-effect relationships in the project feedback structure.

This framework also proposes an explicit classification of exogenous factors into four main classes: ExM1, ExM2, ExE1 and ExE2. Exogenous factors represent external or internal forces that can influence the project behaviour, but which do not participate in the feedback loops – they are not affected by any variable in the model. They can exacerbate, attenuate, or invert the "direction of change" of a feedback loop, but they are not influenced by the evolution of these loops. It should be noted that while by definition all exogenous factors are external to the project feedback processes, some can be internal to the project system, in the sense that they are forces that emerge from within the project boundaries. The other exogenous factors are external to the project system in the sense that they are forces that come from outside the project. Exogenous forces can interfere with the project at both management process level and engineering process level. Two classes are considered at the management process level:

- ExM1 – this class represents those forces that will affect directly the project targets. As they impose changes in the targets, this will motivate management to react. Thus their impacts propagate throughout the whole project. Typical examples are work scope increases, or schedule compressions imposed by the Client;
- ExM2 – this class represents those forces that restrict management to undertake desired control actions. They forces restrict management scope for control and encourage alternative decisions, often inappropriate for the particular problem. For example, shortages in the labour market restrict the recruitment of skilled staff, possibly encouraging current staff to work over time.

The exogenous forces at the management level change the conditions of management performance. The exogenous factors at the engineering process level affect the performance of technical product development. Two classes are considered in the SYDPIM classification framework:

- ExE1 – this class represents the exogenous forces that will affect staff behaviour. For example, productivity being lower than expected due to poor skills, unexpected work complexity, or because of staff unexpectedly leaving the project. These are human issues of project performance at the engineering process level;
- ExE2 – this class represents the exogenous forces that will affect directly the physical process of work accomplishment. They are disturbances that may restrict work progress, but which do not result from changes in staff behaviour. Typical examples include delays or changes regarding dependencies from the outside world, like late delivery of required materials, late design approvals form the Client, requirements changes, among others.

It is proposed in this framework that any relevant exogenous factor of a project system can be classified within one of the four classes above.

It should be noted that the loops and factors proposed in this framework can be more or less *generic*. Project systems are unique, but they also have many common underlying characteristics. While some feedback processes are likely to be observed in most projects, others can be unique to the specific project.

In the SYDPIM Model Development Method, the classes of loops in this framework are used to guide the identification and conceptualisation of the feedback loops relevant for the specific project being modelled. In the same way, the classes of exogenous factors are used to guide the identification of the project risks.

During the conceptualisation phase, it is very important that the inclusion of exogenous factors and feedback loops in the project feedback structure is subjected to a careful analysis to balance *evidence* against *relevance* – i.e. "is the loop really there?", versus "does it matter?" (Rosenhead 1989). In particular, it is important to distinguish and avoid representing indirect cause-effect relationships through direct relationships – this is a temptation in time-pressured modelling environments, and tends to lead to models which are very hard to diagnose. Another critical issue in the conceptualisation of the project feedback structure is the level of detail. The process may easily move from a high-level view to a more detailed perspective, with the initial variables and relationships being dis-aggregated. As a good modelling practice, the author suggests that several influence diagrams should be used (Coyle 1996). These IDs will differ in the level of detail and scope. Each ID should be simple enough so that its feedback effects can be clearly individualised and understood – "spaghetti" type of IDs may look impressive at the eyes of the non-expert, but they will be of little practical help to the modeller.

This SYDIPM classification framework is grounded on a feedback-loop approach to model conceptualisation (i.e. the modelling process is driven by the explicit attempt at identifying loops; see Wolstenholme 1982). However, it also incorporates both a policy and a process perspectives. The process perspective is explicit in the decomposition of the project system into a management process and an engineering process, and the latter further decomposed into the product development and human resources "physical" sub-processes. The policy perspective is explicit within the generic feedback structure of the management process, through the feedback interaction between management and engineering (classes of loops M and ME).

Qualitative validation

The feedback structure identified in this initial modelling stage will affect and condition the characteristics of the SD project model to be developed. In the following stages, the formal model design will be driven by these structure. The model design will ensure that the factors and feedback loops identified in this stage will be captured by the model structure. Likewise, feedback loops that were not considered relevant at this stage will not be included in the model. It is therefore important to ensure that the feedback structure conceptualised at this stage will lead to a model that represents well the project and the management needs. In other words, it is desirable to *validate* this qualitative structure before proceeding to model design.

The validation to be carried out at this stage is necessarily of qualitative nature. It is proposed in SYDPIM that this is based on imposing two main requirements to the feedback structure:

- *scope* – all factors perceived as relevant must be captured by the feedback structure. This can be more or less explicitly and more or less detailed. For example, various factors might be aggregated together in a concept in the IDs;
- *explanatory power* – the feedback structure must be able to explain the occurrence of key modes of behaviour. These can include observed past behaviour or well known generic behaviours.

The validation of the scope must be based on close management involvement. This process can be implemented more or less formally. For example, brain storming sessions can be used to check whether all relevant factors are being captured in the model. In order to bring more discipline into this process, the SYDPIM method suggests the use of the Cognitive Mapping technique (Eden 1994). As previously described in the work of Williams et al (1995), qualitative Cognitive Maps are developed from management input in a flexible manner, and provide a proper specification of the relevant scope. These maps are used as the basis to develop the IDs.

Testing the explanatory power of the feedback structure requires the specification of modes of behaviour for the project. Depending on the situation, there are two types of behaviour modes that can be considered: observed past behaviour, and well known generic modes of behaviour (e.g. the 90% syndrome). The first should be considered only when there is some past available from the specific project. In SYDPIM, the generic modes of behaviour to be considered must include three scenarios:

- *project implementation as planned* – in this scenario, the project plan is implemented successfully within the targets. This scenario portrays a *steady* behaviour, where the targets of cost and schedule are constant over-time;
- *project implementation with over-runs* – in this scenario deviations occur over-time and management control actions take place. This will refer to poor performance below the targets, like schedule delays and cost over-runs. This scenario portrays an *unsteady* behaviour, where the targets of cost and schedule vary over-time;
- *project implementation with under-runs* – in this scenario deviations also occur over-time and management control actions take place. This will refer to good performance above the targets, like cost under-runs. This scenario also portrays an *unsteady* behaviour, where the targets of cost and schedule vary over-time. In the real world this is not a typical scenario;

While some of these scenarios might not be typical and not observed in the actual project being modelled, it is important the conceptualised feedback structure explains these three kinds of behaviour. This is aimed at ensuring the model to be developed will be support validly scenario-analyses in these three domains. In SYDPIM, this is considered as a key model requirement because in the SYDPIM Project Management Method the SD project model will be used no only to reproduce observed past behaviour but it will also have to reproduce both "implementation as planned" and "implementation above the targets". In a sense, this requirements extends the model's validity beyond a particular observed outcome to the full range of possible project outcomes.

In System Dynamics, modes of behaviour are specified as graphs with patterns over-time (e.g. see Richardson and Pugh 1981). Since there is not quantification of the feedback structure at this stage, it is proposed that qualitative graphs are used, with ordinal scales if necessary (e.g. from MIN to MAX). For example, plotting a persistent schedule slippage at the end of the project caused by the typical "90%

syndrome", could consist of plotting the patterns "Estimated completion date" and "Rework remaining". In case, quantitative plots are also available (e.g. from past applications or from the project past), these can also be used. At least one specific project behaviour mode is required for each of the three scenarios described above.

Once the required behaviour modes have been specified, the next step is to try explaining them based on feedback loop analysis. This analysis is primarily based on the principle of feedback loop dominance. According to this, during any period of time only a few feedback loops are dominating the project outcome and hence determining the shape of the behaviour patterns (Richardson 1995, Ford 1998). Therefore, by looking at particular behaviour changes and behaviour segments, the modeller must be able to identify the feedback loops responsible for the respective shapes. In a quantified model, the identification of this loop dominance can be supported by quantitative analyses (Richardson 1995), although there is no formal technique still available. At this initial stage, where the model structure is not quantified yet, the process will have to be informal and strongly supported by managers' mental models.

At the end of this validation exercise, all the behaviour modes identified must have been explained in a plausible manner by the feedback structure.

Deliverable

The output form this phase should be a document specifying the feedback structure developed for the project. This consists of one ore various influence diagrams inter-related in a hierarchy. All the feedback loops and exogenous factors identified should be explicitly classified according to the SYDPIM framework proposed in figure 6.4.

At this stage in the SYDPIM Model Development Method, the conceptualisation of the project feedback structure is not driven by the range of analysis that the final simulation model will be required to perform. Instead, the primary concern is to identify all the factors and feedback loops perceived as relevant to the problems of project performance that concern managers. The next phase in the model development life-cycle focuses on the specification of the model requirements.

Stage 1 – Requirements definition

The purpose of a project model in SYDPIM is to analyse the behaviour of an on-going project. Therefore, there are user *requirements* that the model must satisfy: for example, it must allow the manager to try various decisions regarding work scheduling staff allocation, among others.

The feedback structure specified in the previous stage already implies a range of analysis that the model must be able to perform. However, it is important for the modeller to know exactly what is the set of analyses required, and their level of detail. For example, what specific schedule milestones can be changed and what likely risks need to be tested. These model requirements are here referred to as *analysis requirements.*

In practice, the development of SD models if often started without an up-front precise definition of these analysis requirements. In SYDPIM it is argued that while a final specification might be difficult to achieve at an early stage, a preliminary definition of these model requirements is important, as it brings discipline and guidance to the whole modelling process. This discipline helps to prevent the need for significant changed in the quantitative model at the later stages, which in practice tend to hurt seriously the validity of the model.

The main questions that any project manager is likely to ask when faced with the possibility of using a SD project model, are as follows:

- can the model be used to test the performance of this specific re-planning decisions or policies? In what level of detail?
- which project phases can be analysed individually? What is the project breakdown considered in the model?
- does the model produce the appropriate output so that one can assess the relevant aspects of project performance, and can further understand the underlying causes?
- are the inputs required by the model possible to collect or to estimate in a practical way?

The main aim of this phase is to answer these questions. A specification of the model requirements is produced, ensuring that model being developed will satisfy the needs and practical restrictions of the user (i.e. the project manager).

A first aspect of the model's requirements is the range of analysis provided. This includes both *what* the user will be able to analyse, and the level of *detail*. This range of analysis should cover the testing of *management decisions* and the impacts of uncontrollable factors, or *risks*. It must also address the project areas perceived relevant, where problems may occur and management actions will be required (e.g. life-cycle phases). Another aspect of model requirements has to do with the output produced. Again, both range and detail of this output is important, as this is the basis for the manager to assess the project outcome and to understand the underlying causes. The specification of the model's output is another element of the requirements definition. Finally, in order to assess scenarios, the model requires input parameters for calibration. The model's input requirements are very important: on one hand, the manager must be able to relate practical meaning to the numerical data required, and on the other hand this data should be obtainable from the project information system.

In the SYDPIM method it is proposed that the model's requirements are grouped in three main categories:

(1) *range of analysis supported* – this specifies the "what-if" analyses supported by the model and the level of detail. It can be divided into three sub-categories:
    (i) management decisions;
    (ii) risk factors (internal and external);
    (iii) life-cycle phases covered.
(2) *output produced* – this specifies the output that the model produces to describe and assess the project outcome;
(3) *input required* – this specifies the input data required to perform the range of analyses specified in the first category above.

In the beginning of this stage, the specification of the model requirements should be guided by the influence diagrams developed in the previous stage. However, this time the priority concern is to ensure that the model will cover the practical needs of its user. As mentioned, the influence diagrams already imply some of these requirements, but these are now made more explicit.

The specification of the model requirements in this stage may impose some changes or adjustments to the influence diagrams. This is according to the iterative nature of the proposed life-cycle approach.

The output from this stage is a brief document describing the specific analysis requirements that the model will have to meet, grouped into the three categories identified above. Before proceeding to the next stage, this document should be subjected to verification through review and approval from the managers who will use the model.

Stage 2 – Formal model design

With the causal feedback structure and functional requirements specified, the modeller has addressed two critical issues of model implementation. However, there are still other questions that need to be answered before implementation can be initiated. This includes: the project breakdown into sub-tasks, the characteristics of the SD sub-structures used to model these tasks, the inter-relationships between these tasks, the "physical" and information processes to be captured within each task, and how the causal feedback structure maps into this task structure of the quantitative model.

According to the SYDPIM development method, the formal design of a SD project model comprises the development of the following five design elements:

(1) *model architecture* – it is proposed that a project model should be represented by a set of sub-tasks interrelated through dependency links, forming a network. Both tasks and dependencies can be of various types;

(2) *task classification* – some tasks in the network will differ one another, while others will be similar in many ways. Tasks are therefore classified into *generic categories*. Each of these categories will be modelled by a specialised SD sub-model;

(3) *tasks dependencies* – the characteristics of the dependencies between the tasks are specified in detail, both in terms of product development work and managerial information;

(4) *control decisions* – management feedback takes place through the monitoring of the project status followed by the generation and implementation of corrective actions. Some of these decision-making processes take place within a task, while others may address several tasks. Control decisions are mapped against the project tasks.

(5) *task-based casual structure* – the feedback loops and exogenous factors identified and specified in the feedback structure produced in stage 0, are mapped to the project sub-tasks. Some feedback loops may take place within a task, while others may take place across several tasks;

These elements are specified as formal design elements in the form of a network, matrixes, simple lists of items, graphs, and other familiar types of representation. These five elements are developed within this stage sequentially in the order shown above. This leads to a step-by-step process shown in figure 6.5 below.

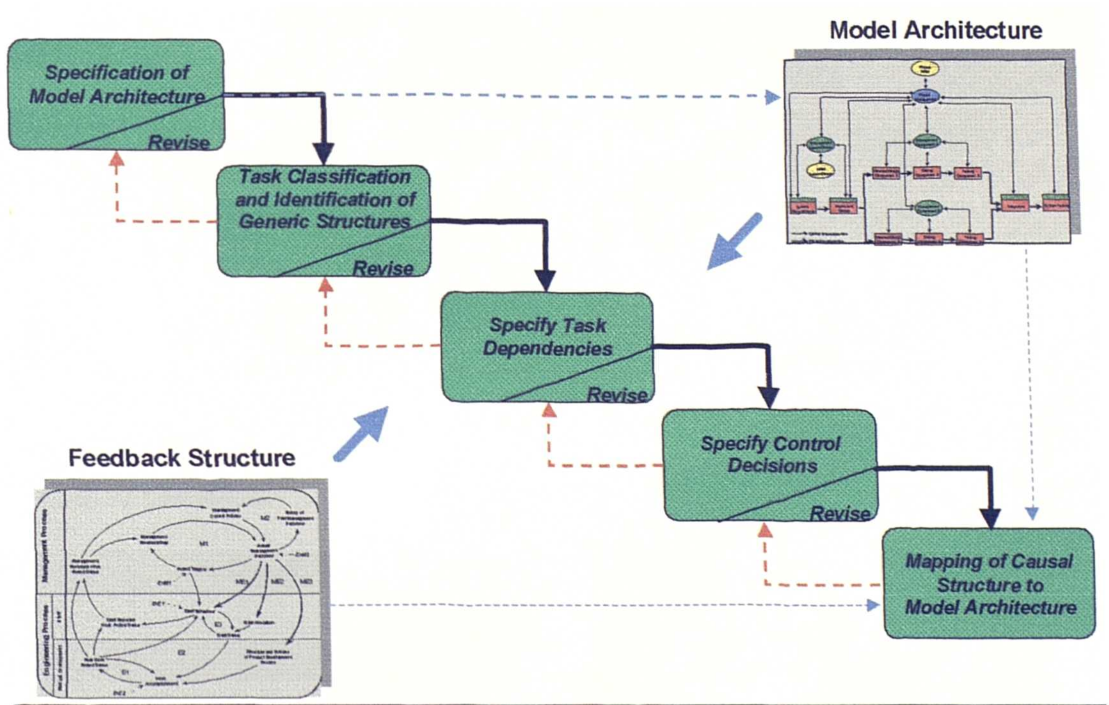

Model Architecture
Specification of Model Architecture
Revise
Task Classification and Identification of Generic Structures
Revise
Specify Task Dependencies
Revise
Specify Control Decisions
Revise
Mapping of Causal Structure to Model Architecture
Feedback Structure

Figure 6.5 – The step-by-step process of SYDPIM formal design (stage 1)

At the end of each step the design element produced is subjected to revision. As shown by the dotted lines, the process is iterative and hence stages can be revisited for changes in the design elements. When this happens, a revision of these takes place before the process proceeds.

As this figure also suggests, two design elements provide a crucial input to these steps: the project feedback structure, specified in causal analysis phase, and the model architecture which is produced in the first step of the process.

The five elements of formal design are now described separately in more detail.

Model architecture

The main aim of the model architecture is to provide a high level structural representation of the project. The project is considered as a system comprised of two major dynamics sub-processes: (1) an engineering process of product development and (2) a management process of project control. In order to model these processes, it is proposed that the project system can be represented as a network of interconnected tasks of management and engineering. Three main types of elementary tasks are proposed: engineering, management, and human resource management (HRM). Engineering tasks represent individual processes wherein work directly related with developing the product is carried out. Management tasks represent the activities of assessing the status of the engineering tasks and generating corrective decisions as required. HRM tasks represent the process of controlling the flow of human resources working in the various engineering tasks. It should be noted that this architectural framework is not intended to be rigid. Instead, it should be used in a flexible manner to accommodate the specific requirements of a project. For example, an engineering task can be coupled or merged with a management task forming, a single engineering-management structure.

The model architecture is specified as a network of interrelated tasks, which will be hereafter referred to as SD-TNet. Figure 6.6 provides an example of this design element for a SD model of a software project.

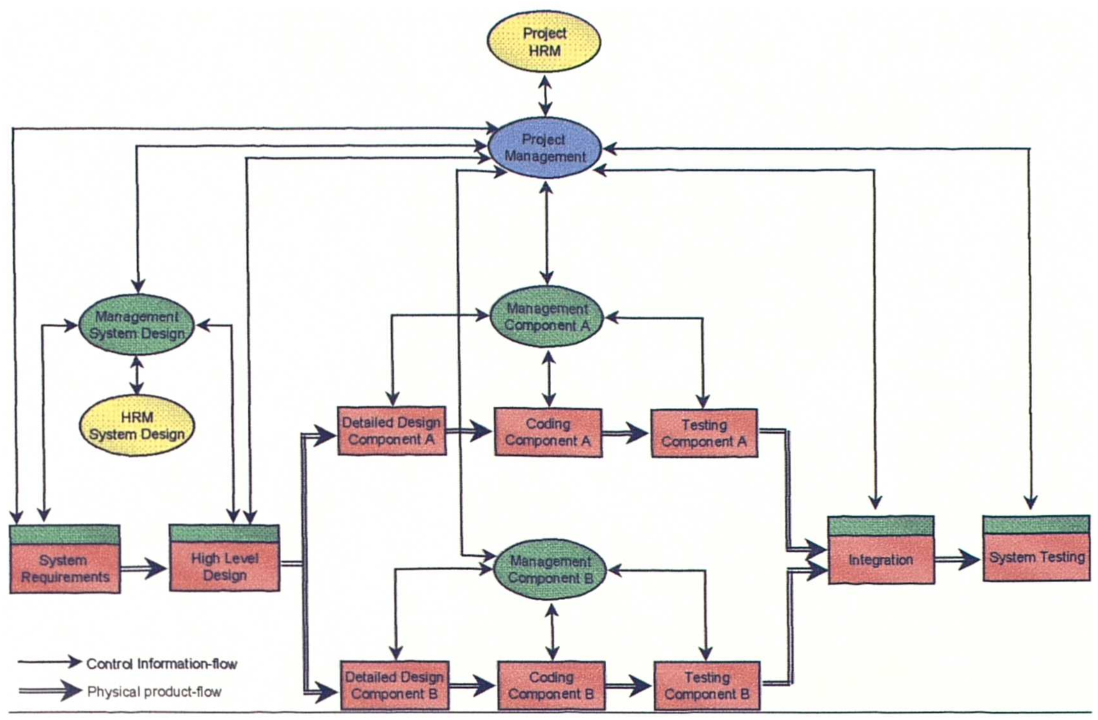


Figure 6.6 – Example of a design architecture specification for a SD project model

For illustrative purposes of the SYDPIM formal design stage this example is now described in some detail.

Overall, this model architecture specifies 10 engineering tasks (rectangles), four of which are coupled with a management task. Four management tasks are specified (ovals), one of which is the high level project management task (second from the top, in blue). Any project model must have this task, which co-ordinates the management of the full project. The two initial engineering tasks of system design have an internal management process which performs some level of internal control, like schedule adjustment within a contingency range. These two tasks are supervised by a higher level management task, which controls the flow of human resources working in them through an individual HRM task ("HRM system design"). The development of each of the two product sub-components is represented by an individual life-cycle of three engineering tasks, each controlled by a single management task. The high level project management task controls the allocation of the human resources to the whole project through a project-wide HRM task. A project model must have at least one project-wide HRM task. The arrows in the diagram identify dependencies between the project tasks, which can be of two types: control information-flow, and physical product-flow. The former identifies the exchange of control information, which can take place between two management tasks, between a management task and a HRM task, or between a management and an engineering task. The second specifies the flow of physical sub-products between engineering tasks (note that this does not imply that the two engineering tasks linked must be implemented sequentially; however, the implementation of the task that delivers will always start prior to the task that receives the sub-product). It is important to note that at this stage in the formal design process, there should be no attempt at specifying in detail the characteristics of these dependencies.

The model architecture will have a major impact on the following design stages. In order to identify the relevant tasks and to link them appropriately, it is important to understand two aspects of the SYDPIM method:

(i) what each of the three generic types tasks is aimed at representing;

(ii) what is the meaning of linking the tasks.

a System Dynamics project task can be seen as an individual system wherein a dynamic process takes place. Like any open system, it has inputs and produces outputs. All of the three generic tasks mentioned are considered in this way.

Let us first consider the case of an engineering task. As a dynamic process the task is fed by three main types of inputs: (1) control information, (2) man-power, and (3) input products (i.e. "raw material"). As the output it delivers a final product and produces information about its internal status. This view is represented in figure 6.7 below.

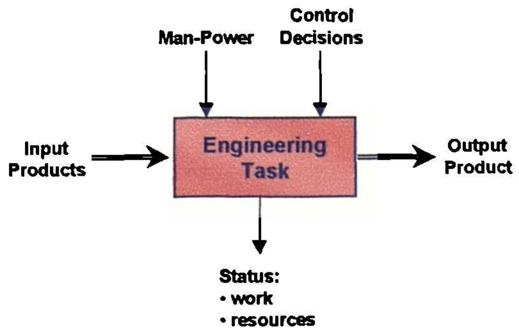


Figure 6.7 – An engineering task as a dynamic process

The work processes implemented within an engineering task consists of human resources carrying out working activities in order to produce the output product. The rate at which this is accomplished depends on the level of the daily man-power allocated to the task, which is specified by external management. As the work is accomplished, the task reports information about its internal status, which can relate to work (e.g. work complete, defects detected) and to resources (e.g. level of over-time being worked). The human resources will carry out the work according to control decisions received by external management. These typically include cost and schedule targets, as well as other decisions regarding the product development process (e.g. QA intensity).

A management task can also be seen as a dynamic process. It plays the role of a control mechanism that continuously monitors and controls another system. As input, it receives information about the status of that system, and as output it produces control decisions. For a project management task, the system being controlled can be either an engineering task or a HRM task. A management task can lay somewhere in the middle of the project management hierarchy. Therefore, it may also play the role of a controlled system, producing status information and receiving control decisions. For example, the task "Management System Design" in the model architecture of figure 6.6 works as both a control mechanism of the engineering tasks "Systems Requirements" and "High Level Design", and as a system controlled by the management task "Project Management". For example, the "Project Management" task receives aggregate information about the two engineering tasks from the "Management System Design" task, and decides about the man-power to be allocated to these set of two tasks. The "Management System Design" task then decides how this ,an-power is split between the two tasks. This view of a management task is generically represented in figure 6.8.

Depending on whether the management task is interacting with an engineering task or with a HRM task, the status information received and the control decisions produced are different. The control of a HRM task is based on specifying the desired target staff levels and on specifying hiring/firing policies (e.g. length of training period). It is here proposed that HRM tasks are controlled only by the specific management task with which it interacts. Therefore, a management task does not send status information about an HRM task to higher level management tasks, nor it receives control decisions regarding this task. For most cases of small and medium size project models, a single HRM task directly controlled by the higher level "Project Management" task is likely to be sufficient.

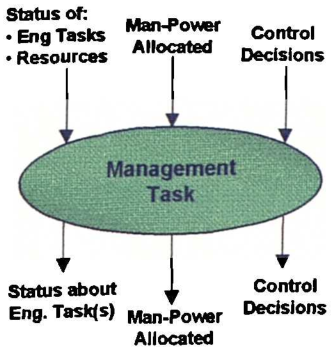


Figure 6.8 – A management task as a dynamic process

Also as a dynamic process, a HRM task receives control decisions form a management task as input, and delivers status information about the resources to that same management task as the output. The HRM task also exchanges resources with the outside world. This view is represented in figure 6.9 below.

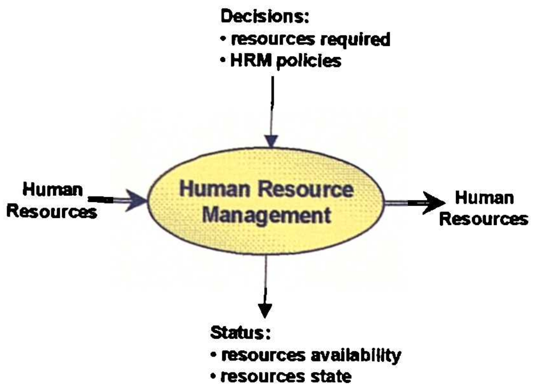


Figure 6.9 – A HRM task as a dynamic process

Whenever the distinction is not essential, human resource management tasks (HRM) will be considered as management tasks hereafter.

The architecture of a project model consists of network of engineering, management and HRM tasks linked as just described above. A management task can be linked with all these three types of tasks, and it always play the role of a controlling mechanism. However, when linked to another management task it may simultaneously play the role of a controlled system. An engineering task is always a controlled system and hence linked to a single management task. It may also be linked to other engineering tasks, with which it will exchange physical sub-products. A HRM tasks is *always* and *only* linked to a single management task. Figure 6.10 below provides a generic representation of how the three types of SYDPIM tasks are linked and how they exchange information and sub-products.

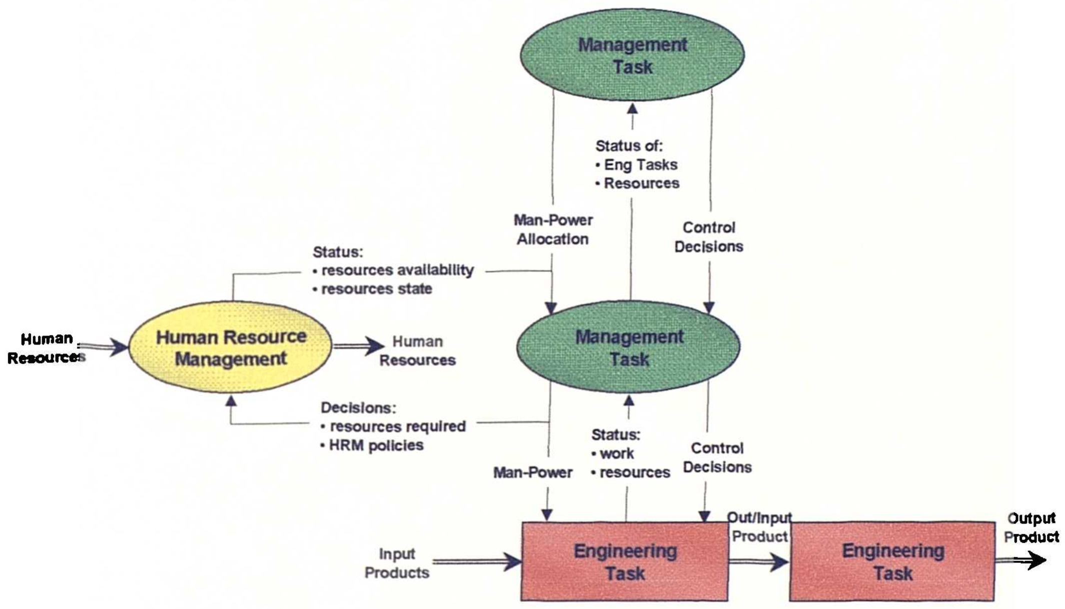


Figure 6.10 – Generic view of how project tasks are linked to form the model architecture

In summary, the model architecture is specified as a network of linked tasks. This network is developed based on to the following principles:

(1) project tasks can be of three types: engineering, management and HRM;

(2) engineering tasks can be coupled with a management task and thereby have an internal management process;

(3) engineering tasks represent the accomplishment of work directly related with product development. These tasks are linked through product-flow dependencies that represent the delivery of intermediate sub-products throughout development the life-cycle;

(4) management task represent decision-making processes where the project status is monitored and control decisions are generated. Management tasks exchange information with and control other management tasks, HRM and engineering tasks;

(5) HRM tasks represent the process of controlling the flow of human resources working in the project, according to specified target levels. This process includes the activities of hiring, training, transfer from other projects, attrition and even firing staff. A HRM task is controlled only by one management task;

(6) a project model must have at least one project-wide management task and one HRM task.

The SYDPIM model architecture represented in this way provides a high-level specification of the structure and scope of the SD project model to be developed.

Task classification

The tasks identified in the model architecture will be simulated in the SD model, through various processes of work accomplishment and information flows. In SYDPIM, each task will be simulated by a specialised SD sub-model. Some tasks can be very similar in nature and therefore can be modelled by common generic sub-structures. This way, in SYDPIM tasks are classified into generic categories so that it is possible to identify how many of these generic sub-models need to be developed. This classification also provides the specification of which specific tasks will be simulated by which of these generic sub-models.

The set of generic tasks categories that needs to be considered depends on the type of project being modelled and on the level of detail assumed in the model. For example, a software project is likely to require different types of tasks than a civil construction project. Furthermore, tasks that can be classified in the same category may always require some specific structural characteristics.

In the SYDPIM method here described there is no attempt to define a rigid set of categories, which would be universally applicable to all types of projects. As a guide-line, the following generic set is proposed:

(a) *Engineering tasks*

> **ETPD** (engineering task of product development/transformation) – this category considers tasks that receive some type of product specifications or design as an input, and either use this as the basis to create a new product as an output, or transform the input product into an output product;

ETPT (engineering task of product testing) – this category considers tasks that receive a certain sub-product as an input, and which functionality is to be tested. The task may also receives as input test specifications. The tests are carried out and the input sub-product is eventually "corrected" to pass the tests. The results and/or corrected sub-product are delivered as an output;

ETPI (engineering task of product integration) – this category considers a task that receives as an input two or more sub-products which is to be assembled within the task into a single integrated output product. The integrated product is typically tested to satisfy some basic functionality.

(b) Management tasks

MCST (management control of single task) – this category considers a management task that monitors the status and controls the scope, the schedule, the man-power allocation, and the resource allocation of one engineering task;

MCMT (management control of multiple tasks) – this considers a management task that monitors the status and controls the scope, schedule, man-power allocation, and resource allocation of two or more interrelated engineering tasks;

(c) Human resource management tasks

HRM (human resource management) – this category considers a management task that controls the transfer of staff resources into and to outside the project.

This generic set of tasks categories suggests that in terms of product development, three main types of engineering tasks can be typically found in projects: (i) the ones that create new products or transform existing sub-products, (ii) the ones that test sub-products against a certain functionality and eventually "correct" the sub-product, and finally (iii) the ones that integrate sub-products into a single product. In terms of management tasks, two types of tasks are typically found: (a) the ones that control a single engineering sub-system, and (b) the ones that control several engineering sub-systems.

It is stressed that this set of generic tasks is not intended to be rigid and universally applied to all projects. Other categories of tasks can be found in certain industries or projects. It is also possible that some of the categories above can be split into sub-categories, if more detail is required for the model's purpose.

Once a set of generic task categories is developed, the specific tasks of the project architecture can be classified in a matrix. For example, using the above set of task categories, the project tasks in the model architecture of figure 6.6 would be classified as follows in table 6.7 below.

	ETPD	ETPT	ETPI	MCST	MCMT	HRM
Engineering Tasks						
System Requirements	✓			✓		
High level design	✓			✓		
Detail Design Comp A	✓					
Detail Design Comp B	✓					
Coding Comp A	✓					
Coding Comp B	✓					
Testing Comp A		✓				
Testing Comp B		✓				
Integration			✓	✓		
System Testing		✓		✓		
Management Tasks						
Management Comp A					✓	
Management Comp B					✓	
Project Management					✓	
HRM Tasks						
System Design HRM						✓
Project HRM						✓

Table 6.7 – Example of a task classification matrix

In this example, the engineering tasks of "systems requirements" and "high level design" are classified as both ETPD (product development) and MCST (single task management control). This is because although these tasks are primarily engineering tasks, where sub-products are developed (the systems requirements specification, and the system design), they also have their own internal management process. This implies that some management control decisions are taken within the task (i.e. they have some degree of self-governance). The tasks of detail design and coding of the two components are also classified as ETPD, but have no internal management process (these tasks are governed by an independent management task). The tasks of component testing are classified as ETPT (product testing) (also governed by an independent management task). Integration is classified as ETPI (product integration) and as MCST (single task management control), again for having an internal management process, while system testing is classified as ETPT and MCST.

This classification matrix provides important information to the modeller: project tasks with the same classification will be modelled by a common SD sub-structure. For example, "systems requirements" and "system testing" will have their internal management process modelled by a same MCST sub-structure. Together with the component design and component testing tasks, these tasks will have their engineering processes modelled by a same ETPD sub-structure.

To each task category it will correspond a generic sub-model or sub-structure. In order to develop the final model, the modeller will use these sub-structures as "building blocks", which will be assembled according to the model architecture. While this approach is important to bring clarity and efficiency to the model development process, it must be implemented in a flexible manner, in order to cope with specific situations. For example, it may be required that the generic sub-structure used to model two different product development tasks (ETPD category), is subjected to some tailoring for each task in order to match their unique but relevant characteristics in the real world.

Tasks dependencies

In order to implement the model architecture into a SD model, appropriate sub-structures are required to model each of the tasks in the SD-Net. The next step is to decide how these structures will be linked according to the tasks' dependencies. In the definition of the model architecture, two types of dependencies were identified: control information-flow, and physical product-flow. The first type of dependencies can involve engineering and management tasks, while the second involves only engineering tasks.

1) Physical product-flow dependencies

Like in PERT/CPM networks, links between two engineering tasks model the effects that constrain or influence work accomplishment in the successor task, given the status of its predecessor.

The dependency links considered in PERT/CPM based networks typically specify a time-lag or time-lead between the start and finishing dates of two tasks. The simplest network models consider just finish-to-start *sequential* dependencies, where work on the successor task can only start after the predecessor task has been fully completed. In order to overcome the rigidity of this basic network approach, further developments have lead to the consideration of *overlapping* tasks (e.g. PDM, Nicholas 1990), to backwards links that allow for tasks to be revisited (e.g. GERT, Nicholas 1990), to stochastic links that allow the non-realisation of tasks (e.g. GAN, Elmaghraby et al 1995) among other approaches. Whatever the complexity of these developments, the core effect of dependency links in logical networks is essentially still the specification of a lag or lead time, which dictates when work in the successor task can start given the time elapsed in the predecessor task. Typically, the links relate the start or finishing dates of the predecessor task with the start or finishing dates of the successor (e.g. start-to-start, finish-to-start, finish-to-finish). These links are intended to represent the technological dependencies between tasks in the real world, where work accomplished in the predecessor task is a requirement to accomplish work in the successor task. This effect is hereafter referred to as "technical work requirements".

A System Dynamics project model that considers the breakdown of a project into sub-tasks must also, somehow, address the establishment of dependency links. The way in which the dependency links are represented in a System Dynamics model can differ considerably from the traditional network models. Potentially it can be more sophisticated in various degrees of complexity, like in the examples reported in Cooper 1980, Williams et all 1995, and Ford 1995. The types of links that should be considered in a SD project model will depend on the following factors: the specific project, the purpose of the model, and even the modeller's personal approach. For the purpose of the SYDPIM method, it is not necessary to consider all the various possibilities. However, there are some basic issues which the modeller should consider in any possible scenario.

The main reason why dependency links between SD-tasks and the traditional network dependencies are different has to do with the level of aggregation. In traditional network models, the tasks are specified at a detailed level of dis-aggregation. Simple precedence relationships, based on time-lags or time-leads, can then represent well the technological work dependencies of the real world. In contrast, SD project models consider tasks at a much higher level of aggregation. Typically, a SD-Task represents a major phase of the product development process, within which a large amount of work is accomplished. In the real world, these phases are interrelated in many ways and the work dependencies are therefore complex. The simple precedence relationships used at the operational level in traditional networks are therefore no longer adequate to capture the dynamics involved.

Dependencies between SD-Tasks must captures some important effects of the real world, as follows:

(1) *technical work requirements* – how much work needs to be available in the predecessor task so that a certain amount of work can be initiated in the successor task?
(2) *work quality* – what are the impacts of the quality of the work delivered by the predecessor task, on the quality of the work being developed in the successor task?
(3) *QA* – what are the impacts of discovering defects in the successor task, on the QA efficiency of the predecessor task, and vice-versa?
(4) *rework accomplishment* – what are the impacts that reworking defects in one task has on rework accomplishment in the other task (e.g. co-ordination requirements; Ford 1995)?

In order to decide about the particular effects to include in the dependency links, SYDPIM proposes that these four categories are considered as the basis. The specific effects to be included depend on their relevance for the project. The particular way in which these are modelled depends on the modelling approach being followed. For example, if the SD model considers precise phase-ends and the explicit delivery of defects to the successors phases the "work quality effects" will be modelled in a rather different way than in a SD model which considers on-going activities with defects never leaving a phase (e.g. the rework-cycle approach described in Cooper 1980, 1993). In the SYDPIM method, different modelling approaches are possible as far as consistency is ensured regarding the relationship between the structure of the SD model and the PERT/CPM network model.

The SD dependencies differ not only regarding the effects they incorporate, but also because they are dynamic, as opposed to the static nature of the PERT/CPM dependencies. The SD dependencies are dynamic in three different ways:

(1) the restrictions to work accomplishment in the successor task are continuously updated as the state of the predecessor task changes over-time. Reverse effects are equally updated in the predecessor task;

(2) technical work requirements can be defined in a dynamic non-linear fashion. For example, when 10% of progress has been achieved in the predecessor 5% of work can be initiated in the successor task, while 30% of progress in the predecessor releases potential progress in the successor task up to 40%;

(3) management decisions can change the characteristics of the dependencies. Given certain project conditions (e.g. progress largely behind schedule), management may decide to relax the constraints of technical work requirements, allowing more concurrent engineering (e.g. Williams et al 1995).

The effects of technical work requirements are typically defined through curves that specify how much work can be initiated in the successor task, given the current work progress in the successor task. These dynamic relationships can vary depending on the characteristics of the specific project. Figure 6.11 shows four examples of these curves. The "time-step "curve is practically equivalent to a time-lead static dependency in a PERT/CPM network model. The "highly constrained" curve represents a relationship where progress in the successor requires a large amount of progress to be accomplished in the predecessor, while the "loose relationship" represents exactly the opposite situation. At on extreme of the "time-step" curve are two purely sequential tasks, while at the other extreme are two purely parallel tasks. Apparently, this type of curves has been considered in some past models, and is presented explicitly in some detail in Ford (1995).

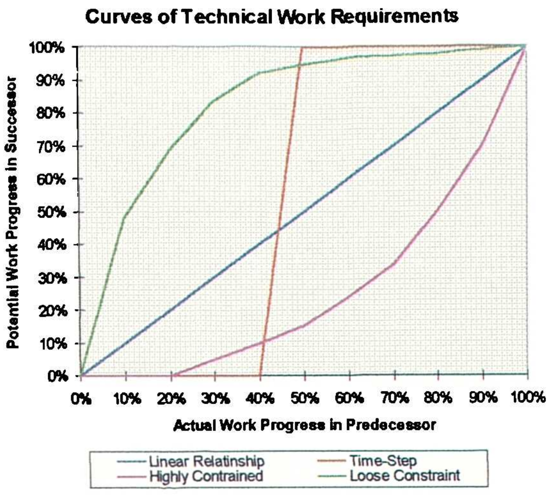


Figure 6.11 – Curves of Technical Work Requirements in a SD-Task dependency

The other three effects of SD dependencies presented above relate with work quality, and can be modelled in various, possibly using approaches similar to this one.

The SD dependencies to be captured in the model are specified in a design element of SYDPIM formal design. The model architecture already identifies the work-flow dependencies between the SD-Tasks. The aim of this step is to identify and specify the type of effects to be captured within each of these dependencies. This design element can be implemented as shown below in table 6.8.

Dependency	Predecessor	Successor	Technical Reqs	Work Quality	QA Effects	Rework Acc.
1	System Requirements	HL Design	✓	✓		
2	HL Design	Detail Design A	✓	✓	✓	✓
3	HL Design	Detail Design B	✓	✓	✓	✓
4	Detail design A	Coding A	✓	✓	✓	✓
5	Detail Design B	Coding B	✓	✓	✓	✓
6	Coding A	Testing A	✓			✓
7	Coding B	Testing B	✓			✓
8	Testing A	Integration	✓			✓
9	Testing B	Integration	✓			✓
10	Integration	System Testing	✓			✓

Table 6.8 – Example of a specification table for engineering task dependencies

For all dependencies, the various types of possible effects are considered. Then, this table is used to identify which potential effect will be incorporated in each dependency. This decision should be based primarily on the relevance of the effects. In principle, the effect of technical work requirements will always be considered between two linked engineering tasks. In this example, there will be no relevant impacts between discovering defects in systems requirements, and in high-level design – perhaps because in systems requirements the product itself is not being developed yet. Also, there are no relevant quality impacts between development and testing engineering tasks – i.e. the quality of undertaking testing activities does not depend, in a relevant manner, on the quality of the code. The decision to incorporating the possible effects in each dependency will depend in great part on the following factors: (i) characteristics of the product development process, (ii) the types of tasks involved in the dependency, and (iii) the relevance of the impact of the specific effect on the project behaviour.

2) Control information-flow dependencies

In the model architecture of figure 6.7, there are several dependency links between management tasks, and between management tasks and engineering tasks. These dependencies specify information regarding the project status and management decisions, which flows across the project. This flow of information is important because it describes how the overall project management process works in order to control the project towards the desired targets.

Management tasks will always receive status information, either directly from an engineering task or indirectly from another management task (laying below in the management hierarchy). Management tasks produce either updated status information or management decisions generated within the task.

The specific status information and management decisions that flow within each dependency link will depend, once again, on the characteristics of the project. It will also depend on the type of tasks involved in the link, as described in table 6.8. The way in which this is implemented depends on the modelling approach. In this step of formal design it is necessary to identify the relevant information contents to be considered in each dependency link. There are some generic contents that should always be addressed. For example, a management task linked to an engineering task will most likely send to the late decisions of schedule adjustment.

Based on the review of past models, and on the author's personal experience, a generic list is proposed in the SYDPIM method, as shown in table 6.9. This table can be used as a framework for specifying the information contents of the control information dependency links between the SD-Tasks. In this generic list, it is considered that an engineering task has no internal management sub-structure. Engineering tasks with such structure will also inherit the characteristics of management tasks in sending and receiving project status information. The information contents are divided into two main groups: (i) project status and (ii) management decisions. Each of these groups is equally decomposed into the various sub-groups of: work, time, effort, and resources. Each sub-group is then further decomposed into more detailed and specific information contents. This proposed classification considers the more common aspects of concern in project management: what to do? how long it takes? how much it costs? what does it require? While this is a generic and comprehensive list, it is not intended to be rigid. The modeller may well need to consider other control information contents.

According to this SYDPIM framework, engineering tasks report "raw" information about the project status to the management tasks. This represents information about the "physical" state of the task, which does not include any "processed" control information aimed at evaluating progress against the objectives and at generating control decisions. Control information is produced and sent by management tasks that receive this "raw" information directly from the engineering tasks. Management tasks can potentially send and receive any type of project status information, being it "raw" or aimed at control (e.g. SPI, *Earned Value*). When a management task sends status information to another management task, this usually refers to aggregate results from a set of tasks it controls. For example, in the model architecture of figure 6.6, the task "Management Component A" sends aggregate information to the management task "Project Management", regarding the aggregate status of the three engineering tasks "Detail Design Component A", "Coding Component A", and "Testing Component A" (e.g. the total effort spent in the three tasks). The "Project Management" task will handle this set of aggregate tasks as if it was an individual engineering task. Management decisions are only sent by management tasks (or management sub-structures within engineering tasks), and refer to action changes that will be ultimately implemented at the engineering level.

Information Contents	**Information flow**		
	Eng. → Man.	**Man. → Man.**	**Man. → Eng.**
Project Status	—	—	—
Work	—	—	—
Tasks accomplished	✓	✓	
Tasks remaining	✓	✓	
Defects/rework detected	✓	✓	
Defects/rework remaining	✓	✓	
Defects/rework removed	✓	✓	
Time		✓	
Elapsed		✓	
Estimated remaining		✓	
Estimated schedule at completion		✓	
Other estimates, performance indices		✓	
Effort	—	—	—
Total spent	✓	✓	
Spent decomposed by activity	✓	✓	
Over-time expenditure	✓	✓	
Estimated remaining		✓	
Estimated at completion		✓	
Other estimates, performance indices		✓	
Resources	—	—	—
Currently employed	✓	✓	
Cumulative employed	✓	✓	
Required to completion schedule		✓	
Extra required		✓	
Excess not needed		✓	
Other processed from the above		✓	
Management Decisions	—	—	—
Work	—	—	—
Scope increases		✓	✓
Scope changes		✓	✓
Scope reductions		✓	✓
Time	—	—	—
Schedule extension		✓	✓
Schedule compression		✓	✓
Effort	—	—	—
Allocation to engineering activities		✓	✓
Budget increases		✓	✓
Budget reductions		✓	✓
Over-time requests		✓	✓
Resources		—	—
Increase current availability		✓	✓
Reduce current availability		✓	✓
Adjustment of future planned profile		✓	✓

Table 6.9 – Framework for specifying information contents of management task dependencies

This list of potential information contents proposed in table 6.9 can be crossed against the set of information control dependencies in the model architecture. This generates a matrix where the information contents of the information control dependencies in the model architecture can be specified (similar as in table 6.8 for the engineering dependencies). Each cell will identify whether a certain information content will be included in the information flow of a certain control dependency.

Control decisions

A distinctive feature of System Dynamics models is to represent and simulate the human processes through which managers take decisions. In stage 1 of SYDPIM model design (requirements definition), the set of management decisions to be considered in the model was identified. The purpose of the design element to be developed in this step is to breakdown that set into more detail, thereby identifying what specific management decisions will be simulated in which task of the model architecture. This is an important design element because the SD sub-structures used to simulate the project tasks will depend on this requirement.

In the real world of project management, control takes place at different levels within the management hierarchy. Decisions at the lower levels are focused on specific areas of the project, and can be translated directly into operational actions (e.g. a team leader may decide to extend the scheduled completion date of a specific design task). On the other hand, higher level management decisions are more general and require a further breakdown prior to their implementation (e.g. the software manager may decide that the overall level of QA effort in the design phase of the project should be reduced from 20% to 15%). In SYDPIM it is proposed that the project model must incorporate the characteristics of this management hierarchy. In model design, this is first outlined in the model architecture, as shown in figure 6.6. Management decisions are simulated either in self-governed engineering tasks, or in most cases within management tasks. The lower level decisions take place within self-governed engineering tasks (e.g. Systems Requirements, High Level Design in figure 6.6). In the following level, decisions take place in management tasks that control directly engineering tasks. This hierarchy extends up to the higher level management task that controls the whole project. This hierarchy of management tasks in the model architecture should represent the management hierarchy as it happens in the real project.

In SYDPIM it is proposed that there are three *generic* types of control decisions that can be considered within a management task:

(1) *endogenous* – the task captures the whole process of: monitoring status, comparing this against targets, identifying a potential gap, and generating a reactive decision;

(2) *transmitted* – the task receives a control decision generated by an upper-level management task, refines this into more detail, and generates further control decisions down the management hierarchy;

(3) *exogenous* – the tasks receives a control decision generated from outside the model boundaries; eventually refines this into more detail, and generates further control decisions down the hierarchy.

Endogenous decisions are modelled through decision rules or management policies, which convert a perceived gap into a control decision. Typical examples of decision rules are "...add n% of the extra resources perceived needed", or "...extend the schedule by x% of the perceived slippage" – these roles can be more or less elaborated. Transmitted decisions generally need to be decomposed into more detailed sub-decisions. For example, the task "project management" in figure 6.6 may generate the decision of extending the schedule of the full development of component B by 20 days. Then, the management task "Management Comp B', receives this decision and needs to decide which of the three development sub-tasks will be delayed and by how many days, and then transmit these sub-decisions to the three engineering tasks. Exogenous decisions are used to represent decision processes that are difficult or even impossible to encapsulate in a decision role within the model. For example, management may decide to adjust the project profile and change the work-scope in a such complex way, which is difficult to deduce from the perceived gaps in the project objectives. In SYDPIM it is proposed that exogenous decisions can be considered in two different ways:

(1) *a priori,* in the beginning of the simulation – these are specified in the form of changes to be imposed in the model, in a certain period of time (e.g. adding a certain profile of resources to the project, from month *n* to month *n+k*). The user normally specifies these decisions after one base-run simulation (at least).

(2) the user specifies the decisions in a "gaming mode" –the user will interactively observe the project outcome in the SD model, and has the power to interrupt the simulation and "exogenously" impose certain decisions.

In order to specify which specific control decisions will be captured in each task of the model architecture, a list of all the potential control decisions is required. In the real world, control decisions can be of various types and of varying degrees of complexity. The set of decisions to be considered in the model will depend on the specific project. In stage 1 of model design (i.e. requirements specification), a first high-level list of the management decisions to be considered was produced. Depending on their complexity, some of these decisions may imply a breakdown into several sub-decisions to be considered in the SD model. In SYDPIM it is proposed that complex management decisions can be generated from simpler elementary decisions. For example, changes in the product development process can be represented by changes to the schedules of the development phases, resource allocation, and to effort prioritisation. SYDPIM therefore proposes a set of elementary generic decisions, as shown in table 6.10. This list is based on the principle that elementary decisions relate to one of the five main dimensions of a project system: work, time, effort, resources, and processes.

In principle, all elementary decisions listed in this table can be represented by endogenous decisions roles. However, some may require considerable complexity and the modeller may prefer not to consider them endogenously in the model. In particular, the decisions 5.1.3, 5.1.4, and 5.1.5 may require architectural changes, usually difficult to implement endogenously (although this is possible, for example, by considering alternative architectures that can be activated through parameter switches). This list is reasonably exhaustive and should be used as a basis. Again, it is not intended to be rigid. The modeller may identify other elementary decisions as relevant, or choose not to consider some proposed in this list.

Management Decisions	***Description***
1. Work	
1.1 Scope increases	New product functional requirements are introduced. The amount of work remaining to be accomplished is increased.
1.2 Scope changes	Some product functional requirements are changed. Some of the work currently in process or work already accomplished, may need to be re-done.
1.3 Scope reductions	The system functional requirements are reduced. Some of the work in process, work already accomplished, and work remaining to be accomplished may be discarded.
2. Time	
2.1 Schedule extension	The scheduled completion date or start date of a task is extended.
2.2 Schedule compression	The scheduled completion date or start date of a task is moved back.
3. Effort	
3.1 Prioritisation to engineering activities	The priorities in man-power allocation among the effort-consuming engineering activities within a task are changed. Examples of typical engineering activities are (1) development, (2) QA, and (3) rework accomplishment.
3.3 Effort allocation to engineering activities	The allocation of the available or planned man-power among the effort-consuming engineering activities within a task is changed (e.g. reducing the QA level).
3.5 Budget increases	The man-power allocated to a task is increased.
3.6 Budget reductions	The man-power allocated to a task is reduced.
3.7 Over-time requests	Staff working in a task is required to use over-time effort.
4. Resources	
4.1 Hiring/firing/transfer	Human resources are hired/fired from the project, or transferred from/to other projects.
4.2 Increase availability	Extra resources are made available to the task.
4.3 Reduce availability	Currently employed resources are removed from a task.
4.4 Adjustment of future planned profile	The planned resource profile of a task and/or of the project is changed.
5. Processes	
5.1 Engineering	
5.1.1 Change inter-task concurrency levels	The work requirements of a task from predecessor tasks are changed. This either increases or restrains potential work accomplishment.
5.1.2 Change intra-task concurrency levels	The intra-task work requirements are changed. This either increases or restrains the potential work accomplishment.
5.1.3 Eliminate tasks	One or more engineering tasks are removed from the product development process.
5.1.4 Introduce tasks	One or more engineering tasks are introduced in the product development process.
5.1.5 Eliminate dependencies	Work dependencies among engineering tasks are removed.
5.1.6 Change process timings	Timings regarding the technical development processes within an engineering task are changed (e.g. duration of QA review periods).
5.2 Management	
5.2.1 Change monitoring delays	The time-delays used in monitoring progress status are changed.
5.2.2 Change control policies	The decision-rules used to generate corrective actions are changed.

Table 6.10 – List of elementary management decisions that can be considered in a management task

The design element to be produced in this step of formal model design consists of a matrix mapping the list of elementary control decisions into the tasks in the model architecture. This can be implemented through a matrix where each cell specifies whether a decision will be consider in each task, and in which of three possible ways described above – i.e. endogenous (ED), transmitted (T), or exogenous (EX). Table 6.11 provides an example of such matrix. For example, it specifies that decisions regarding works-cope changes can be generated only within the highest level project management task. This implies that no other management task will have the power of changing the scope of the work being accomplished in the engineering tasks (e.g. "Management component A" cannot reduce the scope of "Detailed Design Comp A", by its own initiative). Consequently, scope reductions can either be determined endogenously by the project management task (for example, as a reaction to schedule pressure), or they can be imposed exogenously in this task by the user. The management tasks "Management Comp A" and "Management Comp B" can receive a transmitted decision of scope change from the project management task (hence a "T" in the cells). When this happens, these tasks will have to decide how to dis-aggregate the decision down to the engineering tasks. In this example matrix, it can also be seen that all management tasks have the power to adjust the schedule of the engineering tasks below them, possibly within a time-range specified and adjusted by the higher-level project management task. Schedule adjustments are considered as being totally endogenous decisions, and so not even the project management task can receive such decision as an exogenous input.

Reading through this type of table, the modeller will be able to know for each management task: (i) which decisions will have to be generated endogenously, (ii) which decisions can be received from an upper-level management task and hence need to be dis-aggregated, and (iii) which decisions can be received as an user exogenous input. This information has important implications to the structure and calibration of the SD sub-structures that will model each task. For example, in many cases, two management tasks will be classified as being of the same type (see task classification matrix in table 6.7), and hence will modelled by a common SD sub-structure. However, if different sets of decisions are considered for each task in table 6.11, then the "switch parameters" should be implemented to enable/disable decisions within the generic SD sub-structure which will model that type of tasks.

It is important to note that the information specified in this design element must be consistent with the model architecture and with the information captured within the SD dependency links, as specified in the previous steps of formal design. For example, if a management task is specified in table 6.11 as possibly receiving schedule adjustments decisions from an upper level management task, then it will have to be linked to a higher management task in the model architecture and the control information link between them must contain this decision. Of course, the information of this design element should also be consistent with how the management structure of the project works in the real world.

Mapping of causal structure to model architecture

A SD project model can contain numerous feedback loops. Some will take place within individual tasks, while others take place across several tasks. In a complex model, it may not be practical, or even useful, to identify *all* the feedback loops contained within the model structure. Identifying all these loops before the model has been implemented at the equations level is even harder. However, there are some important categories (or types) of feedback loops that dominate the project behaviour. These loops should have been identified and classified in the initial stage of causal analysis (stage 0), based on the proposed high level classification framework. Most of these loops are generic and take place in various areas of the project. For example, the feedback effect representing how work progress is reinforced by natural staff learning, is likely to be applicable in all engineering tasks.

When the generic sub-structures for the SD-Tasks in the model architecture are developed and linked, it is necessary to ensure that all the relevant feedback loops have been captured as desired, within and across all tasks. In some cases a specific feedback effect should be captured in certain tasks, but ignored in others due to low relevance. The dependency links between the SD-Tasks must ensure that the relevant loops are captured across the various project tasks as desired. Exogenous factors are also an important part of the feedback structure identified in stage 0. Some exogenous factors may need to be present in more than one task. Therefore, in order to develop the SD sub-structures some mapping of the feedback loops and exogenous factors to the model architecture is required.

Identifying all the feedback loops that will appear in the model structure at the equation level is not feasible. In fact, some loops will be discovered while the model structure is being build (i.e. a learning process). The aim of this step is therefore to map the feedback loops and exogenous factors identified in stage 0, into the tasks of the model architecture developed in the current stage. This mapping will be useful in the implementation of the SD model: it will let the modeller know which loops and exogenous factors should be considered in each task, and across tasks in the model architecture. Throughout the implementation process this design element will be updated, helping the modeller to keep track of which loops are being captured where, within the whole and possibly complex structure of the SD model.

Mapping the tasks against feedback loops and exogenous factors can be implemented using a matrix. Conceptually, it is possible to define a single matrix crossing all the tasks in the model architecture against all types of feedback loops and exogenous factors identified in the causal structure of stage 0. However, in practice this can lead to a large and complex matrix difficult to represent. On the other hand, there are some feedback effects which can only take place within certain types of tasks. Furthermore, it is also necessary to identify unambiguously those loops that cross various tasks. An alternative and more practical way of representing this mapping, is therefore to decompose this into several matrices. In SYDPIM, the following set of matrices is proposed:

(1) *intra-task engineering matrix* – maps the feedback loops internal to the engineering process, as defined in stage 0 (i.e. E1, E2, and E3 types), against the engineering tasks. These loops take place only within a single task;

(2) *inter-task engineering matrix* – maps the feedback loops internal to the engineering process, as defined in stage 0 (i.e. E1, E2, and E3 types), against the engineering tasks. These loops take place across several tasks;

(3) *intra-task management matrix* – map feedback loops internal to the management process, as defined in stage 0 (i.e. M1, and M2 types), against the management tasks. These loops take place only within a single task;

(4) *inter-task management matrix* – maps the feedback loops internal to the management process, as defined in stage 0 (i.e. M1, and M2 types), against the management tasks. These loops take place across several tasks;

(5) *engineering-management matrix* – maps the feedback loops that cross through both engineering and management processes, as defined in stage 0 (i.e. EM1,

EM2, and EM3 types), against all tasks in the model architecture. By definition, these loops take place across several tasks;

(6) *engineering exogenous factors* – maps exogenous factors internal to the engineering process, as defined in stage 0 (i.e. ExE1, and ExE2 types), against the engineering tasks;

(7) *management exogenous factors* – maps exogenous factors internal to the management process, as defined in stage 0 (i.e. ExM1, and ExM2 types), against the engineering tasks.

An example of an "intra-task engineering matrix", mapping E-type of feedback loops into engineering tasks, is provided in table 6.12 (exogenous factors can be mapped in the same way). The first column presents the three generic intra-engineering feedback loops proposed in stage 0 (E1, E2, and E3; see figure 6.4), decomposed into some generic sub-categories. These loops are mapped against all engineering tasks of the model architecture in figure 6.6.

This design element provides important information for the implementation of the model. It informs the modeller about the relevant feedback effects that must be captured explicitly in the SD sub-structures which will model each of the SD-Tasks in the model architecture. For example, in table 6.12 it is clear that in "Systems Requirements", the feedback effect of error re-generation, where undiscovered errors contribute to the generation of more errors, will not be considered. This may be due to the fact that in this early stage of software development, the "intra-work" dependencies are not so relevant as in the later development stages. This effect is also not relevant in the testing tasks, perhaps because no "product creation" is undertaken in these tasks. It is also possible to see that the effect of work complexity is considered in coding of component B, but not in coding of component A, perhaps because the first involves highly interconnected code. Other modelling assumptions can be read in this matrix, which is therefore a rich design element. Each cell in the matrix makes explicit important assumptions about the real world which will be considered in the model.

Mapping feedback loops and exogenous factors into the tasks in the model architecture imposes important characteristics on the SD sub-structures to be developed. This information needs to be related to the task classification matrix, a design element developed in a previous step of this stage (see table 6.7). This classification matrix identifies the SD-Tasks that fall in a same category and hence will be modelled by a same generic sub-structure (e.g. ETPD – engineering task of product development). However, as previously stressed, each individual task may require some specific changes to this generic structure. The matrices developed in this step, mapping feedback loops and exogenous factors to the tasks, like the matrix in table 6.12, provide a rigorous way to identify some of the important differences among the tasks in a same category. For example, the development tasks "Systems Requirements", "Designing", and "Coding", are all classified as ETPD in table 6.7. However, while most of their SD structure will be the same (i.e. the generic ETPD structure), changes must be made to ensure that the feedback effects in table 6.12 are considered as desired – a practical way of implementing these differences is through the use of on/off switches to (de-)activate feedback loops.

Inter-task feedback loops can be of any of the eight types identified in figure 6.4. Because these loops cross several tasks, their chains of cause-effect will be implemented through both tasks' internal structure and dependency links specified in tables 6.8 and 6.9. An example of an E1 inter-task feedback effect, is the quality of the work of a predecessor task reducing the quality of the work being done in a successor task; as errors are discovered in the successor task, corrections may also be required in the predecessor task, thereby delaying overall progress. For inter-task feedback loops a simple "yes/no" specification as in table 6.12 is not sufficient. The modeller must identify through which tasks the feedback loops cross. This can be done using a simple table with two columns, one identifying the feedback loops and the other identifying the sets of tasks through which these loops take place. Table 6.13 illustrates this concept.

Once more, it is important to note that the purpose of this table is not to identify all the specific feedback loops that will take place in the model at the equations level. Instead, the aim is to identify, at the appropriate level of detail, the most relevant inter-task feedback loops that must be captured explicitly in the final model structure. The feedback loops identified in this type of matrices must be consistent with other two design elements specified in previous steps: (i) the model architecture (see figure 6.6), and (ii) the management and engineering dependency links (see tables 6.8 and 6.9). For an inter-task feedback loop to take place between a set of tasks, the two following conditions must be respected: (1) there will have to be links specified in the model architecture that allow for such loop to take, and (2) the types of links and/or their information contents must allow for the rewired effects to take place. For example, the "Inter-task work availability" effect between the tasks {Requirements Specifications, High-Level Design}, as identified in table 6.13, is possible because: (a) there is an engineering dependency link between these two tasks (see figure 6.6), and (b) this dependency considers the "technical work requirements" effect (see table 6.8). However, an "Inter-task QA effect" feedback loop would not be possible to take place because this dependency does not consider "QA effects".

Types of Feedback Loops	Sets of Tasks
E1 – work state ↔ work rate	

Inter-task work availability	{Requirements Specifications, High-Level Design}
	{High Level Design, Detail Design Comp A}
	{High Level Design, Detail Design Comp B}
	....

Inter-task Quality impacts	....
Inter-task QA effects	....
Inter-task rework accomplishment	....
E2 – work state ↔ staff state	
...	....
E3 – staff behaviour ↔ staff state	
...	....

Table 6.13 – Example of an engineering matrix mapping inter-task E-type of feedback loops into the engineering tasks of the model architecture

An important element required to develop the seven matrices proposed in this step of formal design is the list of feedback loops and exogenous factors to be considered in the model. As already mentioned, in SYDPIM these are classified according to the framework proposed in stage 0. At the end of that stage, the model's feedback structure was specified and therefore an initial version of this list is available. Throughout model design, it is perfectly normal that, as new design elements are specified (e.g. model architecture), the modeller may identify the need to extend or change this list. In this continuous process of updating the feedback structure, it is important to involve managers and the staff working in the project, to ensure that only the relevant effects are being considered. Some examples of the eight types of feedback loops and four types of exogenous factors were described in stage 0 (also in tables 6.12 and 6.13), but many others can be considered. Good sources of information are the general literature in Project Management, and the past applications of System Dynamics reviewed in chapter 2.

The matrices developed in this step of formal design provide a rigorous mapping of the qualitative causal structure of the project to be captured in the SD model, into its formal architecture of tasks. This is an important element of formal design, because it provides the modeller with rigorous information about the feedback requirements of the quantitative SD model. These requirements will tailor the SD sub-structures in the model, as well as the links among them. These matrices can also be very useful in supporting model the practical use of the model. They establish a "bridge" or relationship between the qualitative and quantitative structures. When interpreting the results produced by the model, these matrices will support the analyst in relating the feedback loops of the causal structure to the outcomes observed in each task and in the whole project – which loops are the driving forces of certain behaviours. This relationship is important in order to understand the causes of behaviour and thereby to all aspects of practical model usage.

Summary

The stage of formal model design just described provides a rigorous and flexible specification of the project model to be developed. The model design produced consists of a set of formal design elements. These elements incorporate all the critical *assumptions* upon which the model application will be considered, and upon which the model validity can be judged. This includes the level of aggregation, the main interconnections in the project, the core feedback loops, the scope and structure of the project management process (e.g. decisions considered, management hierarchy).

This stage of formal design builds upon the previous stages of causal analysis (stage 0) and requirements specification (stage 1). The causal analysis stage provides the specification and classification of the key feedback loops to be captured in the model. The requirements specification stage identifies what the model has to be able to accomplish (i.e. range of analysis, input and output). Both of these elements are considered in this stage of formal model design.

The design elements developed in this stage will guide the model implementation phase at the equation level. These elements address the most important questions that the modeller will have to consider during implementation:

(1) how does the model structure represents the whole project? What are the work phases, intermediate schedules, effort, and resources breakdown considered? How is the management structure captured?

(2) which tasks in the model structure represent similar processes and hence can be modelled by a similar SD sub-structure? How many of these generic sub-structures are needed?

(3) how are the project tasks interconnected in the model? How do they impact one another? What are the effects considered?

(4) What control decisions are considered in the model? Which of these decisions are considered in each task, and how?

(5) How does the project causal feedback structure (developed in stage 0) relates to the model structure? Which loops are captured within and across which tasks?

The design elements developed are summarised as follows:

1. *Model architecture (SD-TNet)* – a visual network of interconnected tasks of engineering, management, and human resource management. It provides a high level representation of the product development process and of the project control process. It identifies: engineering tasks, management tasks, human resources tasks, and the links among these tasks (see figure 6.6).
2. *Task classification* – a matrix mapping the tasks in the SD-TNet against generic task categories. Each task category will be modelled by a common SD sub-structure. This matrix identifies the generic task categories, and classifies the individual model tasks accordingly (see table 6.7).
3. *Task dependencies* – two matrices specifying (i) the effects considered within the links between engineering tasks, and (ii) the information contents within the dependency links between management tasks and between management and engineering tasks (see table 6.8 for engineering links, and table 6.9 for management links).

4. *Control decisions* – two matrices, one identifying the set of elementary control decisions to be captured in the model as decision roles or as exogenous inputs, and the second mapping these decisions against the management tasks in the model (see table 6.10 for list of elementary control decisions, and table 6.11 for mapping against management tasks).
5. *Mapping of causal structure to model architecture* – set of seven matrices mapping the feedback loops (intra-task and inter-task effects) and exogenous factors in the causal structure (stage 0), into the tasks in the model architecture (see table 6.12 for an example of mapping intra-task engineering type of loops; see table 6.13 for an example of mapping inter-task engineering type of feedback loops).

When specifying all these design elements, the modeller must ensure that the information they incorporate is consistent one another. While they are primarily used as the basis for implementing the model at the equation level, they will be used throughout the whole model development process as an important source of information, and should be updated whenever changes are required or errors are identified.

The next phase of the SYDPIM development process addresses model implementation. This consists of developing the SD project model at the equation level, based on the design. This phase includes the development of the model sub-components (i.e. SD sub-structures), their integration, and model validation.

6.6.4 Phase B – Implementation and validation

After the SD project model has been designed, the next step is implementation, using a System Dynamics simulation language and tool. The process of transforming the model design into a quantified computer model is here referred to in SYDPIM as "model implementation".

For this second phase of the development process, SYDPIM proposes a structured framework for model implementation and validation, which is described in this section. First, the implications of the specific simulation language to be used are discussed. A structured framework for model implementation and a structured framework for model validation are then presented separately. This is followed by a description of how these two frameworks are used together into a single integrated process of model implementation and validation. The SYDPIM Model Development Method proposes that implementation and validation are two processes closely interrelated. This implies that the model is not be implemented first and only then tested for validation. Instead, validation will be carried out progressively during implementation.

The simulation language

In practical terms, the specific simulation language used to develop the model may have a considerable impact on the implementation process. The following factors can be critical:

(1) *availability of visual interface* – the first SD simulation language used to build models was Dynamo (Pugh 1985). This language had the great disadvantage of not supporting a visual interface from where the model could be developed. Later, other SD languages emerged in the market as user-friendly software tools which supported this feature, such as Powersim, iThink/Stella, and Vensim. The visual interface allows the model to be developed using a small set of visual objects that can be created and linked directly on the screen, using a mouse. These basic objects typically include in these visual languages are: "Level"/"Stock", "Flow-Rate", "Auxiliary", and "Information Flow", but can include more. Although the model development method here proposed does not depend on the availability of this visual feature, it will be assumed that a visual interface is available to the modeller. At present, the more commonly used languages / tools support this feature;

(2) *replication of sub-structures* – a critical issue in the development of SD models which have potential to become complex (like project models), is the possibility of sub-structures being easily replicated. Typically, a SD project model will include several repetitions of a same sub-structure, which will differ only in terms of parameter calibration. Most SD modelling languages provide this feature, more or less comprehensively, and hence it will be here assumed that this is not a constraint to the model implementation process;

(3) *mathematical features* – the availability of pre-defined mathematical formulations can affect the way in which the model is formulated. When a certain formulation

needed is not available as a "built-in" function, the modeller will have either to dis--aggregate the model, or to use abstract sub-structures to produce an equivalent formulation. It will be assumed that the more common formulations will be available to the modeller;

(4) *performance* – when a model has potential to become very complex, time-performance can be a problem. While this can be a serious restriction in practical terms, this will not be considered as a limitation. To the author's experience, the present versions of the most popular SD tools perform within reasonable time-scales.

SYDPIM assumes that a reasonably powerful simulation tool is available.

Basic principles

In the design phase, the project model was decomposed into sub-tasks as specified in the model architecture (i.e. the SD-TNet). While there are dependency links among these tasks, each will have its own core independent structure. The approach proposed in SYDPIM to implement these core structures is based on a set of principles.

The first principles is that each of the project tasks can be implemented and (partially) validated individually. The project tasks will then be progressively integrated and further validated into the final model.

The second principle is that some project tasks can be modelled by a common generic structure (which can save a considerable development effort). In the model design phase, the tasks are grouped into categories. Each category it will be modelled by a generic SD structure – these are hereafter referred to as "super-structures". The super-structures required are identified, in the task classification matrix developed in the design phase (see table 6.7). After a super-structure is developed, it is then replicated and tailored to model each of the specific tasks in the project architecture according to the classification matrix.

The implication of using super-structures is that each one will have to accommodate the requirements of several project tasks. For example, according to the classification matrix of table 6.7, the super-structure MCMT (control of several engineering tasks) will have to capture the control decisions required for the three project tasks: "Management of Comp A", "Management of Comp B" and "Project Management". While most of the requirements are likely to be common to the all tasks in a certain category, some may be unique to each task. Therefore, the super-structures must be developed in a way which allows an easy tailoring for each of the individual tasks in the category.

Super-structures will always be of one of the three elementary types proposed in the model architecture: engineering (ENG), management (MAN), and human resource management (HRM). The third principle for the proposed framework is that each type of super-structure calls for a slightly specialised implementation approach.

The fourth principle is that the conceptualisation of any super-structure is carried out progressively, based on the generic feedback structure proposed in figure 6.4. The various feedback loops are progressively captured in the SD structure and are tested for behavioural impacts. This is perhaps the more important principle of the SYDPIM Model Development Method. It will also be the basis to guide the stages of model integration and final validation.

The fifth and final principle is that the implementation and validation of a super-structure requires calibration to a certain scenario. By definition, a super-structure is generic and hence this scenario could refer to any of the specific project tasks to be modelled by the super-structure. However, it is not desirable that the development of a super-structure should is biased by any of these specific tasks – the danger being an excessive tailoring of the super-structure towards the specific characteristics of one task, in prejudice of the others. Instead, a fictitious or historical scenario should be preferably used. This scenario should have two main components of data: (1) the input characteristics (of the situation being simulated; e.g. product size, planned budget and schedule, productivity), and (2) the output behaviour to be reproduced (e.g. cumulative work accomplished, actual staff profile). The first component is essential to calibrate the super-structure. The second is not mandatory, but if available can be used to support the validation process. This scenario will be hereafter referred to as "super-structure reference scenario".

Likewise, the tailoring and calibration of the super-structure to the specific project tasks will also require a "task reference scenario".

In summary, the proposed framework specifies that the model development should start with the implementation of the required super-structures, through a series of steps, progressively incorporating more feedback loops. The development of a super-structure is based on the calibration for a reference scenario. Depending on the type of super-structure (i.e. ENG, MAN, and HRM), slightly different implementation steps are followed. Once developed, the super-structures are then replicated and tailored to model each of the project tasks in the model architecture. This tailoring is based on task-specific reference scenarios. During the whole process, the SD structures are continuously subjected to validation tests (this will be described in the validation framework). The model is then progressively integrated and further tested for validation. When the model is fully integrated it is then subjected to final validation tests.

The life-cycle process

As previously described, overall the SYDPIM method considers a life-cycle of three major implementation stages: (3) component development, (4) integration, and (5) validation. This is illustrated in figure 6.12, where stage (3) is split into the two steps of (i) development of super-structures and (ii) tailoring of super-structures to project tasks.

The process is iterative and hence it can cycle-back to previous stages. Validation is carried out continuously throughout the process at the end of each stage. The formal elements of model design provide the required input information to the implementation stages, in particular the project feedback structure and the model architecture.

The implementation of this life-cycle process is illustrated in figure 6.13, where a simple project model with four tasks is considered as an example. This figure shows that the integration stage progresses through several steps, until the whole model is assembled. The model is progressively validated throughout the integration process, where the emphasis of the validation tests will naturally shift into the new structures added to the model in each integration step. In the final validation stage, the model is calibrated to reproduce the project reference scenario. This figure also highlights the use of the design elements as sources of information during the model implementation process. These elements are essential to identify the structural requirements of the SD structures being developed, tailored and integrated. The three types of reference scenarios to be used are developed in the beginning of the stage, based on information from the real project and from the formal design elements.

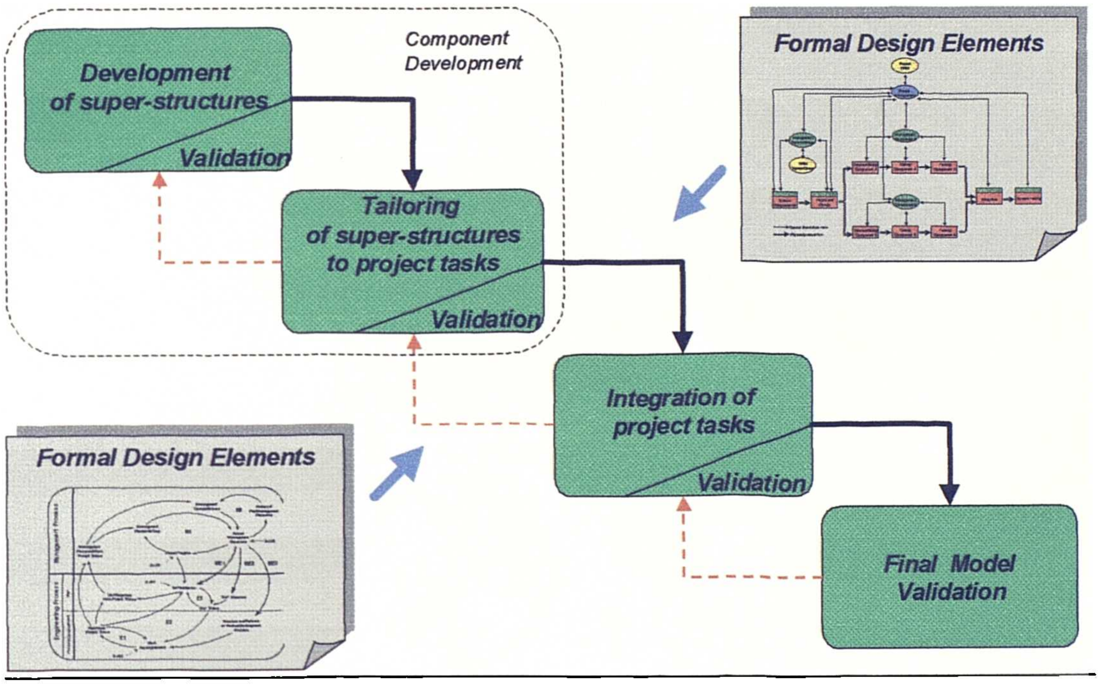


Figure 6.12 – Life-cycle process of SYDPIM model implementation

Note that the SYDPIM implementation process assumes that the generic structures will be tailored to the specific tasks *before* they are integrated into the final model. There are two main reasons for this approach: first, it helps to keep discipline throughout the process. In particular, otherwise the criteria to decide about which sub-networks of generic structures to be build first, and about their "ideal" size to start tailoring, would be subjective. Secondly, the validation of an assembly of sub-structures is more complex and again more subjective. If there is a specific case where the modeller feels that assembling a specific set of generic-structures is a more appropriate approach, then such a strategy can be followed by considering such an assembly as a generic-structure itself. The implementation and validation process can then be implemented as proposed in SYDPIM, helping to maintain the overall desired discipline .

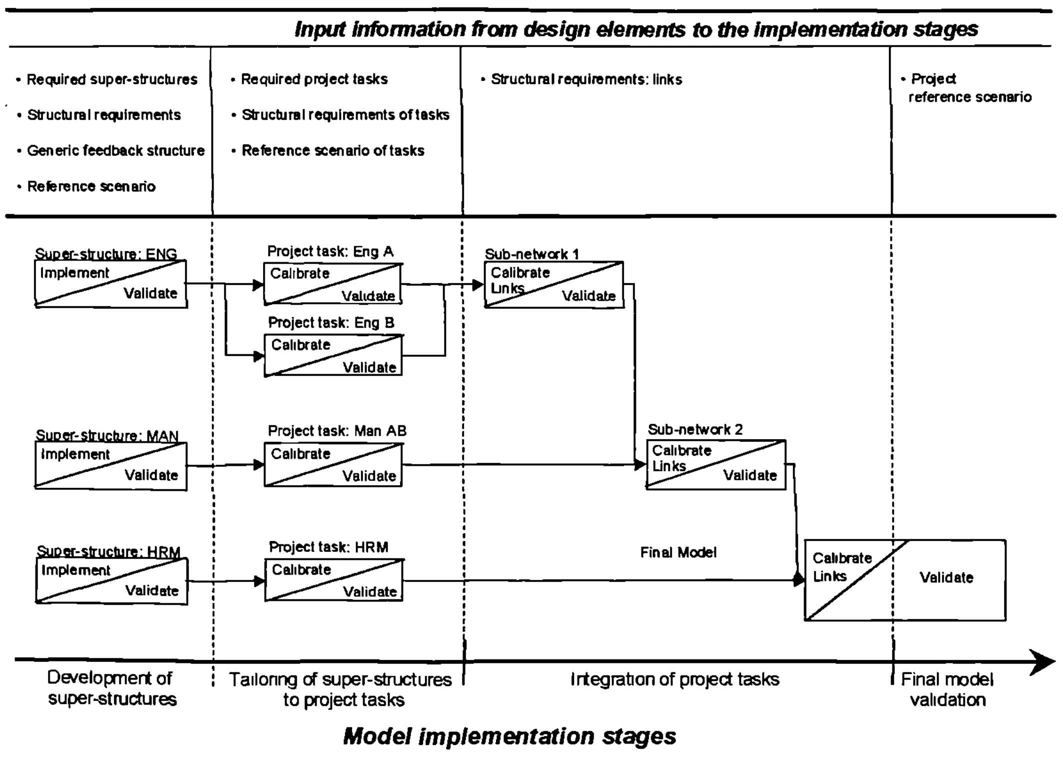


Figure 6.13 – Overview of the proposed implementation framework

The implementation of the life-cycle process of SYDPIM implementation can be described formally as shown in algorithm 6.1.

This algorithm describes the model implementation as a logical step-by-step process. The general process and steps to be implemented within each of the three stages are described, wherein the formal design elements are used as a critical source of information.

The validation activity at the end of each step is based on a reference scenario, which the sub-structures and model will have to reproduce. In order to understand the causes for the behaviour produced, design elements describing the feedback loops contained within the sub-structure are used. Validation also takes place during the implementation steps and is here not restricted to reproducing the reference scenario. SYDPIM proposes a structured validation framework to be integrated within the implementation process. This is described in the next section.

```
Algorithm: {SYDPIM model implementation process}
    STAGE 3a: {component development: development of super-structures}
    FOR each <super-structure> in <task classification matrix (table 6.7)> DO
        Produce <super-structure requirements> from:
                <task classification matrix (table 6.7)>,
                <inter-task physical dependencies (table 6.8)>
                <inter-task information control flows (table 6.9)>
                <tasks management decisions (table 6.11)>
                <intra-tasks feedback loops (table 6.12)>
        Specify <super-structure reference scenario>
        Implement <super-structure> using:
                <generic feedback structure (figure 6.4)>,
                <super-structure requirements>
                <super-structure reference scenario>
        Validate <super-structure> using:
                <super-structure reference scenario>,
                <super-structure requirements>
    ENDFOR

    STAGE 3b: {component development: tailoring of super-structures to project tasks}
    FOR each <super-structure> in <task classification matrix (table 6.7)> DO
        FOR each <project task> to be modelled by <super-structure> DO
            Specify <task reference scenario>
            Produce replication of <super-structure>
            Tailor and calibrate <super-structure> to <project task> using:
                    <task management decisions (table 6.11)>
                    <task internal feedback loops (table 6.12)>
            Validate <project task> using:
                    <task reference scenario>
                    <task internal feedback loops (table 6.12)>
        ENDFOR
    ENDFOR

    STAGE 4: {integration of project tasks}
    FOR each <link> in <model architecture (figure 6.6.)> DO
        Integrate <project tasks> in <link> using:
                <inter-task physical dependencies (table 6.8)>,
                <inter-task information control flows (table 6.9)>,
        Validate <task sub-network> using:
                < reference scenarios of tasks in sub-network>,
                <inter-task feedback loops (table 6.12)>
                <generic feedback structure (figure 6.4)>
    ENDFOR

    STAGE 5: {final model validation}
    Specify <project reference scenario>
    Validate <project model> using:
        <project reference scenario>
        <generic feedback structure (figure 6.4)>
```

Algorithm 6.1 – Formal description of the SYDPIM model implementation process

Model validation

Overview rationale

In SYDPIM, the validation of a System Dynamics model is considered in two main dimensions: (i) the concept (i.e. what it means), and (ii) the process (how it is done). In terms of concept, the ideal of validation means that the model provides a true representation of how a system works and behaves in the real world. However, as discussed in chapter 2, models of social systems inevitably represent a "biased" view of those involved in the modelling process. Such models therefore cannot be seen as an independent representation of reality. While the "valid" model must certainly comply with many aspects of reality, it does so by incorporating the various mental models of the participants in the modelling process. In this context, the ultimate requirement of the "valid" model is its ability to produce consistent scenarios of behaviour (in the past and in the future), both to explain the outcomes and to deliver achievable solutions (see earlier discussion in chapter 2). Furthermore, the validity of a model should not be divorced from its purpose and usefulness.

In terms of process, validating a model ideally consists of conducting a well defined set of tests (or any other type of verification activity), which, if passed, ensures that a model "truly" represents reality. However, as discussed in chapter 2, in System Dynamics there is simply no single test that, *per se*, can ensure model validity in this way. Furthermore, there is currently no established formal process through which a SD model can be validated. In practice, validation tends to be carried out on an *ad hoc* personal basis, although established confidence testes and quantitative techniques can be used. The absence of a formal process for model validation raises potential difficulties in practical applications, because the user audience will mostly likely question the conceptual truth of the model. To the author's knowledge, the models reviewed in this chapter were "validated" using personal such approaches. In the general practice of SD, there is no or very poor detailed documentation description of how the validation process is carried out (Peterson and Eberlein 1994).

In the absence of such a vital requirement for SD project modelling, a structured framework is proposed in the SYDPIM method. This framework is specialised in project models and builds upon the more relevant validation work currently available in the literature. This includes, primarily, the generic validation approach proposed by Barlas (1996, 1985), and the early work of Forrester and Senge (1980). The important concept of "partial model testing", proposed by Homer (1983), is also considered. A brief summary review of these pasts developments is here presented.

Review of existing frameworks

The first relevant contribution towards the establishment of a structured approach to model validation in System Dynamics was proposed by Forrester and Senge (1980). The authors proposed a set of confidence tests that should be carried out to validate any SD model. According to this work validation is proposed as a gradual process of building confidence in the model as these tests are carried out and passed. The set of tests is summarised in table 6.14.

The authors grouped the confidence tests in three main categories: (1) structural, (2) behaviour, and (3) policy implication. Structural tests question the underlying logic behind the model as a formal description of the real system (i.e. is this how the system operates in the real world?). Behaviour tests focus on questioning the model's ability to reproduce real scenarios, or scenarios consistent with mental models (i.e. is the model able to show what happened, or what is likely to happen?). Finally, policy implication tests question the model's consistency in providing policy recommendations (i.e. will the recommended policies lead, or have led, to improvements?).

Confidence Tests	Description
Tests of model structure	
• Structure-verification test	Verify and directly compare the structure of the model against the structure of the real system
• Parameter-verification test	Conceptual and numerical verification against knowledge in the real life system.
• Extreme-condition test	Test the equations that lead to the rates from the states. Check rate-values for extreme values of levels.
• Boundary adequacy	Check if further structural developments are required in order to test if existing structure is sufficient to meet the model purpose
• Dimensional consistency test	Check unit measures in both sides of equations
• Other tests	Statistical tests must be conducted with care.
Tests of model behaviour	
• Behaviour-reproduction test	How well the model generated behaviour matches observed behaviour in the real system
• Symptom generation	The model must reproduce the symptoms of difficulty that created the problem
• Frequency generation	Periodicity of fluctuation
• Relative-phasing	Phased relationships between variables
• Multiple mode test	Test if the same structure is able to reproduce more than one mode of observed behaviour
• Behaviour characteristic	The model must reproduce typical characteristics of the system behaviour in the real life
• Behaviour prediction	Test if the model generates qualitatively correct patterns of future behaviour
• Event prediction	The model must be able to predict the characteristics of sudden events
• Behaviour anomaly test	Identify and trace causes of produced behaviour not likely to occur in the real systems
• Family member	The model must stand transplantation reproduce the behaviour of other members of the family of system it models
• Surprise behaviour test	Check if aspects of the behaviour produced by the model can be uncovered as being also exhibited by the real system
• Extreme-policy test	Check if results produced with extreme condition policies are consistent with what would happen in the real world
• Boundary adequacy test	The same as structure but includes analysis of model behaviour
• Behaviour sensitivity	Identify sensitive parameters in the model and question whether the real system is also sensitive. Test if other combinations of likely parameters make the model fail other tests
• Other tests	Kalman tests (data vs. behaviour)
Tests of policy implications	Test if a change in policy in the real system causes the changes predicted by the model
• System improvement test	Test if policies found beneficial with the model also improve the real system (ideal but difficult to implement)
• Changed behaviour prediction test	Check if policy changes in the real system were as predicted by the model
• Boundary adequacy test	Check if major policy recommendation produced by the model change if the boundary of the model is changed
• Policy-sensitivity test	Check if recommended policies are sensitive to uncertain parameters within their plausible range

Table 6.14 – Confidence tests proposed by Forrester and Senge (1980)

The proposed confidence tests are not ready to be implemented in a given model. Instead, they identify *classes* (or types) of tests that can be conducted in any model. For each class, the authors provide a conceptual definition and discuss their nature and aims. No formal process is proposed, describing how specific tests within each class can be generated for a given model. In practice, the specific tests will depend on the type of model and problem being addressed. There is also no formal specification of the acceptance criteria for the tests (i.e. when are the results "good enough", so that a test can be considered as "passed"?). Equally, there is no specification of how test failures should lead to changes in the model. The key relevance of this work is that it provides a useful framework to identify and understand classes of validation tests, which can be used to generate and conduct specific tests.

Later, the work by Barlas (1985, 1989, 1996) proposed a more structured and formal validation approach. The major novelties of this work were: (1) a formal multi-step process for behaviour validation, (2) a formal validation phase at the end of model development, and (3) a general framework to apply the existing confidence tests within this validation phase. Regarding (1), this consists of a six-step procedure where quantitative indices are produced to assess whether the behaviour produced by the model is consistent with the observed behaviour in the real world. Regarding (2) and (3), the confidence tests proposed by Forrester and Senge (1980) were re-grouped as shown in table 6.15.

Confidence Test	Description
Structure validity	
• Direct structure	Direct comparison of each relationship with knowledge about the real system (no simulation involved). Very qualitative.
• Empirical tests	Comparison with qualitative or quantitative information extracted from the real system
• Structure confirmation	Form of the equation with the relationship that exists in the real system (mostly qualitative)
• Parameter confirmation	Value of constant parameter with knowledge about the real system
• Conceptually	Elements in the real system that correspond to the parameter
• Numerically	Estimate of the numerical value of the parameter
• Theoretical tests	Comparison with generalised knowledge existing in the literature
• Structure confirmation	Form of the equation with existing knowledge in the literature (mostly qualitative)
• Parameter confirmation	Value of constant parameter with knowledge in the literature
• Conceptually	Elements in the real system that correspond to the parameter
• Numerically	Estimate of the numerical value of the parameter
• Extreme condition	Each equation is tested for extreme values of input parameters
• Dimensional consistency	Checking of dimensional units in both sides of each equation
• Structure oriented behaviour tests	Assess the validity of the structure indirectly through behaviour tests. Involves simulation. Can be applied to the whole model or to sub-models. Can be formalised (e.g. Vensim reality check).
• Extreme condition (stress testing)	Assigning extreme values to the model parameters and compare behaviour with likely or observed behaviour of real system
• Behaviour sensitivity	Identify parameters to which the model is highly sensitive and question if that happens in the real world
• Modified behaviour prediction	Test if the structural modified model exhibits the behaviour of available modified version of the real system
• Boundary adequacy	Test if additional structure alters essential aspects of model behaviour
• Phase relationship	Compare phase relationships between pairs of variables in the model with the phase relationships observed/expected in the real system
• Qualitative features	Specify qualitative features of the "expected" behaviour of real system under certain conditions and check if the simulated behaviour also exhibits those features. Count the number of runs where expected and simulated features match.
• Turing test	Expert is shown a mix of simulated and real behaviour. It is statistically checked whether there was a distinction between the two types of behaviour.
Behaviour reproduction accuracy	Check how accurately the model reproduces the observed behaviour of the real system. The emphasis should be on pattern prediction and not on point prediction.
• Steady-state behaviour	
• Multi-test	Six-step statistical procedure (Barlas 1985, 1989)
• Statistics of goodness-of-fit	See Sterman (1984)
• Other tests	See Forrester and Senge (1980)
• Transient behaviour	No statistical tests should be applied
• Graphical/Visual measures	Compare graphical visual measures of most typical behaviour-pattern characteristics
• Quantitative features	Situation-specific formulas that quantify behaviour pattern characteristics

Table 6.15 – Confidence tests proposed by Barlas (1994, 1996)

According to this author, confidence tests can have two distinctive purposes: (i) test the structural validity of the model, and (ii) test the model's accuracy in reproducing behaviour observed in the real system. Structural tests may either question the model's logic directly without requiring simulation, or may be based on simulated behaviour. Barlas' framework differs from Forrester and Senge (1980) in the way in which the tests are grouped. Barlas gives special emphasis to the fact that behaviour tests can be used to question the model structure, without having to match any particular reference mode of behaviour. It also gives less emphasis to the explicit definition of policy oriented behaviour tests and further considers other tests inherited from other fields, like the Turing test (Turing 1950). Finally, Barlas asserts that the type of behaviour accuracy tests should differ, depending on whether the system exhibits steady-state or a transient behaviour. Barlas' framework can be summarised in the following main points:

(1) the formal validation phase starts as soon as the first version of the model is available;

(2) confidence tests should then be carried out in a sequence of three steps: (i) direct structure tests, (ii) structure-oriented behaviour tests, and finally (iii) behaviour accuracy tests. It is important to consider that behaviour accuracy tests are only meaningful if the model has passed the structural tests.

(3) if at any step the model fails to pass a test, then the process cycles back to review the model and is re-started again. If the test that failed was of behaviour accuracy and the revisions caused no changes to the model structure (i.e. only required changes in parameters), then the process can proceed directly to the behaviour accuracy tests (i.e. structure tests are skipped);

(4) the behaviour accuracy tests to be applied on the model depend on the type of behaviour exhibited by the system. If it exhibits a steady-state behaviour, then a particular quantitative multi-step procedure is proposed (Barlas 1985, 1989). In this case, statistics of "goodness-of-fit" should be used (see Sterman 1984). In case the mode of behaviour is transient, then these general statistics might not be appropriate and situation-specific formulas can be used instead to assess "goodness-of-fit". Visual and / or graphical measures of the more important characteristics of the behaviour patterns can also be used.

Once Barlas' framework builds upon the confidence tests proposed by Forrester and Senge (1980), this framework will be used as the main source of reference in the SYDPIM method.

Finally, the partial model testing technique proposed by Homer (1983), consists of testing and validating individual sub-structures of a model, against empirical exogenous input data and against empirical output data. This technique is particularly useful to estimate parameters or formulations when the required dis-aggregate information is not available within an acceptable level of confidence. Partial model testing advocates that individual pieces of a model can be tested and refined before the whole model is subjected to validation tests. Homer (1983) argues that this procedure may enhance significantly the quality to the whole model.

Principles of the SYDPIM validation framework

The validation framework proposed in the SYDPIM Model Development Method, builds upon and extends the SD validation work described above. According to SYDPIM, validation is integrated into the implementation process based on the following principles:

(1) confidence tests are applied throughout the implementation process, as the model gradually evolves towards a final structure;
(2) different types of confidence tests should be applied at different stages in the implementation process;
(3) the degree of tolerance to pass behaviour tests should decrease, as the model evolves towards its final structure (i.e. the acceptance criteria becomes more strict);
(4) some confidence tests are applied to individual super-structures in isolation, at the end and during its development;
(5) behaviour accuracy tests can only be applied to the final model, or to "endogenously self-contained" structures. In both cases, it is required that a formal description of a reference mode of behaviour is available (historical or likely extracted from mental models).

The validation framework is now explained within the context of the implementation framework previously presented.

The SYDPIM model validation framework

As just described in the implementation framework, the SYDPIM implementation process evolves from implementing individual super-structures, through a series of steps, to progressively integrating these into a final model. Therefore, during this process the possible SD sub-structures available for validation are:

(1) generic super-structures;
(2) super-structures tailored and calibrated to specific project tasks (i.e. task-structures);
(3) sub-networks of integrated task-structures;
(4) the final project model.

According to the SYDPIM method, validation tests are carried out during and after these structures are developed. Therefore, these tests can be carried out at any of the following stages:

(a) during any implementation step of a super-structure;
(b) at the end of each implementation step of a super-structure;
(c) at the end of the full development of a super-structure;
(d) at the end of tailoring and calibrating of a super-structure to a project task;
(e) during the integration of project tasks;
(f) at the end of integrating project tasks into sub-networks;
(g) after the full model has been integrated.

According to the proposed principles of SYDPIM validation, not all confidence tests are appropriate to be applied at each of these stages. Based on this, the following additional principles are proposed:

- *direct structure tests* – these tests do not use simulation and should be applied only at those stages where the model structured is being modified (e.g. new variables being created, parameters and input functions being estimated). This includes stages (a), (d), and (e) above. Although direct structure tests can be considered as part of structural conceptualisation, they should be carried out explicitly.
- *behaviour tests* – before the full model is assembled, these tests should be conducted with special attention because the *complete* feedback structure of the project is not yet captured in the model. The behaviour patterns produced by

"incomplete" structures cannot be judged as if the final model was being considered. SYDPIM proposes that applying behaviour tests to an incomplete structure can be considered in the following situations:

(1) if the model embodies a self-contained endogenous structure, then structure-oriented behaviour tests can be applied. If historical or known behaviour of the sub-system is available, then behaviour accuracy tests can also be conducted;

(2) if the incomplete structure is not endogenously self-contained, but the modeller can anticipate with confidence the impacts of the "structural incompleteness" on behaviour, then structure-oriented behaviour tests can be carried out. These tests require some degree of tolerance. On the other hand, behaviour accuracy tests should not be conducted;

(3) if the structure is not self-contained and the modeller cannot anticipate the impacts of this on behaviour, then the incomplete structure should be evolved until one of the previous cases is reached;

(4) behaviour tests for partial model testing (Homer 1983) can be applied to very small "pieces" of the model, like incomplete structures not endogenously self-contained. This technique should be used to validate unmeasured or intangible parameters and formulations.

The set of tests proposed in the SYDPIM validation framework is summarised in table 6.16. The first column identifies the validation stages throughout the implementation process, where confidence tests should be conducted. The other columns group the tests into three main categories, according to Barlas' framework: direct structure tests, structured oriented behaviour tests, and behaviour accuracy tests. In each implementation step, the essential tests are marked as "advised" in the sense that they conducted. Other helpful but not essential tests are marked as "useful".

During the steps of implementing super-structures, all direct structure tests should preferably be conducted. In particular, extreme condition tests and tests dimensional consistency tests should be carried out explicitly. Partial model testing should be conducted if intangible or unmeasured parameters and formulations need to be estimated (this procedure will require the description of empirical input and output behaviour to be available; this can be derived from hard data collected or from expert-opinion).

At the end of each implementation step of a super-structure, more feedback loops have been incorporated in the model. Therefore, further validation tests can be carried out to assess the behaviour impacts. While a super-structure is not a complete structure, it can be considered as being endogenously self-contained. Therefore, according to the SYDPIM principles above, the behaviour tests should only be structure-oriented. Qualitative features tests should be conducted to assess whether the behaviour is consistent with the new structural additions. Extreme condition tests can help to assess the robustness of the new additions to extreme situations. Boundary adequacy tests should question whether the new structural additions are the best alternatives. Given that the structure is evolving, the acceptance criteria of these tests should be tolerant. As a complement, behaviour sensitivity and phase-relationship can also be conducted, helping to assess the consistency of the behaviour currently being produced and to identify sensitive parameters at an early stage.

At the end of the last implementation step, the super-structure is complete, incorporating all the required feedback loops and exogenous factors. At this stage, the same structure-oriented behaviour tests should be conducted in a more exhaustive and strict manner. The qualitative features test should focus on the model's ability to reproduce the key characteristics of the output of the "super-structure reference scenario". In these tests, the model is subjected to the input conditions of the reference scenario, which may include not only constants, but also time-varying exogenous variables that will represent potential interactions with other model components – in case of non-self-contained endogenous structures, the qualitative features test can become similar to partial model testing applied to the generic structure. Phase-relationship and behaviour sensitivity are now required tests. The modeller should check the key behaviour relationships between certain sets of variables, and should identify for the future the sensitive parameters. All these tests should be conducted under a strict acceptance criteria. Alternative pieces of the model structure can also be hypothesised, depending on the tests results, although the structure is likely to be adequate at this stage.

The next step in the implementation process is to tailor and calibrate the super-structures to the project tasks. This step will be based on the initial definition of a "task-reference-scenario", which can be historical or anticipated (i.e. empirically derived). Tailoring the model will consist in many cases in tuning "on/off" structural switches, to ensure that the super-structure will incorporate the structural requirements of the specific task. The validation of this procedure is straightforward). Calibration of the model parameters is a more delicate procedure and should be subjected to the "parameter confirmation" direct structure test. This test will ensure the objective validation of the numerical values used for the parameters. Partial model testing should be used to estimate the remaining unmeasured or intangible parameters. In case historical scenarios are available for the specific project task (e.g. from very similar past tasks in the real world), then the structure can be calibrated to generate various additional scenarios and so Turing tests can be further conducted to reinforce validity. Finally, behaviour accuracy tests should be conducted to test if the structure is capable of reproducing well the reference scenario of the specific task.

Once all tasks in the model architecture are modelled by individual SD structures, the next step is to gradually integrate these into sub-networks, until the final model fully assembled. Integrating calibrated sub-structures consists mainly in specifying parameter values and equations to implement the links between the tasks. During this step, all direct structure tests should be carried out to ensure that the definition and calibration of the links is validated.

After a set of tasks has been integrated into a sub-network, behaviour structure tests should be conducted. Extreme condition and behaviour sensitivity tests should focus on the new parameters and equations added to the model to create the links between the tasks. Phase-relationship tests should focus on the relationship between variables from different tasks, in order to check whether the cross-task behaviour characteristics are consistent. In most cases, it is unlikely that there will be historical behaviour available for the specific sub-network of tasks being tested, and so behaviour accuracy tests are not appropriate. Qualitative feature tests can be used to assess the overall behaviour characteristics of the sub-network and should be carried out under a tolerant acceptance criteria.

Finally, after the full model has been integrated into a single network of tasks, behaviour accuracy tests should be conducted. If there is already historical behaviour available from the project, the accurate reproduction of this behaviour is of crucial importance. For the future segment of the project, the reference mode of behaviour to be reproduced is the one portrayed by the current project plan – as it will be discussed in the next chapter, *the model's ability to reproduce the planned behaviour is critical for the implementation of the SYDPIM Project Management Methodology.* As a complement to behaviour accuracy tests, hypothetical "what-if" scenarios for the current plan can also be tested.

An overview of the SYDPIM model validation framework is shown in figure 6.14. The proposed sets of confidence tests, grouped in table 6.16 for each validation stage, are mapped into the model implementation process. This way, the seven stages of the SYDPIM validation framework are integrated into the three SYDPIM implementation stages.

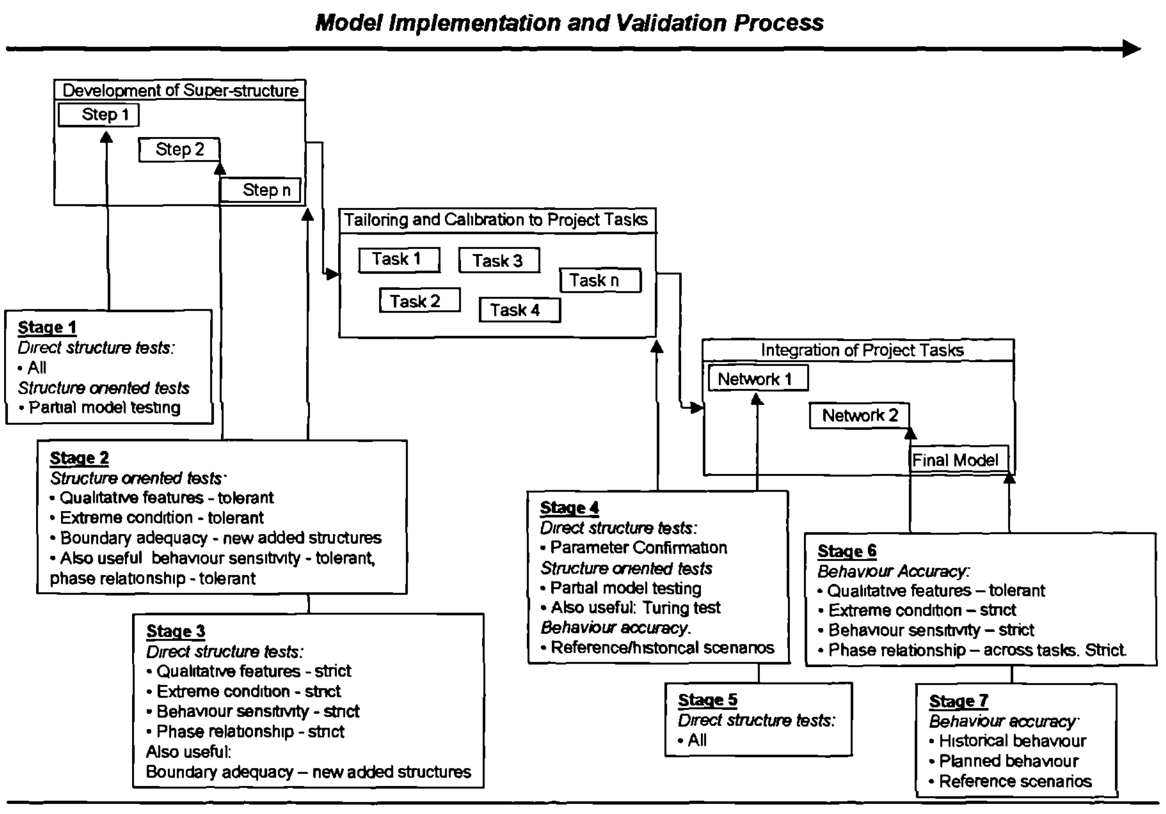

Model Implementation and Validation Process
Development of Super-structure
Step 1
Step 2
Step n
Tailoring and Calibration to Project Tasks
Task 1
Task 3
Task n
Task 2
Task 4
Integration of Project Tasks
Network 1
Network 2
Final Model
Stage 1
Direct structure tests:
• All
Structure oriented tests
• Partial model testing
Stage 2
Structure oriented tests:
• Qualitative features - tolerant
• Extreme condition - tolerant
• Boundary adequacy - new added structures
• Also useful behaviour sensitivity - tolerant, phase relationship - tolerant
Stage 3
Direct structure tests:
• Qualitative features - strict
• Extreme condition - strict
• Behaviour sensitivity - strict
• Phase relationship - strict
Also useful:
Boundary adequacy – new added structures
Stage 4
Direct structure tests:
• Parameter Confirmation
Structure oriented tests
• Partial model testing
• Also useful: Turing test
Behaviour accuracy.
• Reference/historical scenarios
Stage 5
Direct structure tests:
• All
Stage 6
Behaviour Accuracy:
• Qualitative features – tolerant
• Extreme condition – strict
• Behaviour sensitivity – strict
• Phase relationship – across tasks. Strict
Stage 7
Behaviour accuracy:
• Historical behaviour
• Planned behaviour
• Reference scenarios

Figure 6.14 – Integrating the SYDPIM validation stages into the implementation process

The SYDPIM integrated process of implementing and validating a project model can now is formalised in the algorithm 6.2. This is an extension of algorithm 6.1, integrated with the validation framework described in table 6.16.

The implementation stages

The generic modelling procedures to be followed within each implementation stage is described algorithm 6.2 above. In each stage, the formal design elements development in the previous phase the SYDPIM validation framework in table 6.16 are used as inputs to provide the required information. The actual implementation of the model consists in developing its structure at the equation level, which requires a modelling tool as previously discussed. In most tools this will consist in developing a "level/rate" diagram, quantifying the relationships through equations and calibrating the model parameters and initial values.

While much of the required creativity to develop a SD model was already employed during the design phase, the implementation process also requires this human input. Therefore, the model implementation process cannot be specified as a fully automated process of converting formal design into a quantified model. Instead, important modelling decisions regarding the representation of the real world in the model still need to be taken at this level. The best solutions will depend on the specific project and application of the model. Furthermore, in many cases there will be various technical alternatives to implement the design specifications. These alternatives have impacts on the way in which the model can be used in practice, and further refined or changed in the future. These are other important modelling decisions specific to the practical application which the modeller must take.

Some level of modelling discipline is important within modelling the process of each implementation stage. The known good principles of modelling should be applied (Richardson and Pugh 1981, Morecroft and Sterman 1994, Coyle 1996). In particular, under a SYDPIM perspective it is very important to make an effective use and articulation of the various SYDPIM formal design elements and of the validation framework.

```
Algorithm: {SYDPIM model implementation and validation process}
    STAGE 3a: {component development: development of super-structures}
    FOR each <super-structure> in <task classification matrix (table 6.7)> DO
        Produce <super-structure requirements> from:
                <task classification matrix (table 6.7)>,
                <inter-task physical dependencies (table 6.8)>
                <inter-task information control flows (table 6.9)>
                <tasks management decisions (table 6.11)>
                <intra-tasks feedback loops (table 6.12)>
        Specify <super-structure reference scenario>
        FOR each step in <super-structure implementation process> DO
            Implement and validate (stage 1) <super-structure> using:
                    <generic feedback structure (figure 6.4)>,
                    <super-structure requirements>
                    <confidence tests for stage 1 (table 6.16)>
            Validate (stage 2) <super-structure> using
                    <super-structure reference scenario>,
                    <super-structure requirements>,
                    <confidence tests for stage 2 (table 6.16)>
        ENDFOR
        Validate (stage 3) <super-structure> using:
                <super-structure reference scenario>,
                <super-structure requirements>,
                <confidence tests for stage 3 (table 6.16)>
    ENDFOR
    STAGE 3b: {component development: tailoring of super-structures to project tasks}
    FOR each <super-structure> in <task classification matrix (table 6.7)> DO
        FOR each <project task> to be modelled by <super-structure> DO
            Specify <task reference scenario>
            Produce replication of <super-structure>
            Tailor and calibrate <super-structure> to <project task> using:
                    <task management decisions (table 6.11)>
                    <task internal feedback loops (table 6.12)>
            Validate (stage 4) <project task> using:
                    <task reference scenario>
                    <task internal feedback loops (table 6.12)>
                    <confidence tests for stage 4 (table 6.16)>
        ENDFOR
    ENDFOR
    STAGE 4: {integration of project tasks}
    FOR each <link> in <model architecture (figure 6.6.)> DO
        Integrate <project tasks> in <link> and validate (stage 5) <link>_ using:
                <inter-task physical dependencies (table 6.8)>,
                <inter-task information control flows (table 6.9)>,
                <confidence tests for stage 5 (table 6.16)>
        Validate (stage 6) <task sub-network> using:
                < reference scenarios of tasks in sub-network>,
                <inter-task feedback effects (table 6.12)>,
                <generic feedback structure (figure 6.4)>,
                <confidence tests for stage 6 (table 6.16)>
    ENDFOR
    Specify <project reference scenario>
    Validate (stage 7) <project model> using:
        <project reference scenario>
        <generic feedback structure (figure 6.4)>
        <confidence tests for stage 7 (table 6.16)>
```

Algorithm 6.2 – The integrated SYDPIM process of model implementation and validation

The modelling work carried out in each implementation stage depends on the very specific aspects of the model application – e.g. the project, the modeller, the modelling tool, the management needs, effort and time available. The ideal to illustrate the SYDPIM implementation stage would be the full implementation of a project model based on a specific real project. Depending on the model complexity, this could be an lengthy and extensive description.

However, the SYDPIM Model Development Method was tested in the real case-study project which supported this research – i.e. the KDCOM project at BAeSEMA. The SYDPIM method was outlined in the beginning of the case-study as a conceptual framework, it was progressively revised into more detail refined during the case-study and finally extended to form a formal method as described in this chapter. While presenting the full modelling experience followed during the case would also be a considerably lengthy process, a simplified practical example of the implementation phase is now presented based for illustrative purposes. This example covers part of stage 3, where a generic super-structure for a specific category of engineering tasks is developed.

Illustrative example: implementation of a generic structure (stage 3a)

The starting point of the overall model implementation process is to develop the super-structures for each of the task categories in the task classification matrix. This is the first step of stage 3. As described in algorithm 6.2, the first task in this process is to identify structural requirements of each super-structure from the design elements. These requirements will then be the basis to "code" the super-structure into equations.

Specifying the structural requirements of super-structures

The logic of this task is described in algorithm 6.2 as follows:

```
FOR each <super-structure> in <task classification matrix (table 6.7)> DO
    Produce <super-structure requirements> from:
        <task classification matrix (table 6.7)>,
        <inter-task physical dependencies (table 6.8)>
        <inter-task information control flows (table 6.9)>
        <tasks management decisions (table 6.11)>
        <intra-tasks feedback loops (table 6.12)>
```

A "super-structure" must incorporate all the features required by all the specific project tasks in the model architecture that the super-structure will have to model. The set of structural requirements for each super-structure are derived from the design elements. These requirements can be grouped as follows:

(1) processes incorporated (engineering, management, human resources)
(2) links to other tasks (types, amount, information contents).
(3) control decisions captured (for management tasks only).
(4) internal feedback loops and exogenous factors captured.

Once derived from the design elements, these structural requirements can be compiled in a table. For example, let us consider the model architecture in figure 6.6. Five generic super-structures were identified in the classification matrix of table 6.7: ETPD, ETPI, ETPT, MCST, MCMT, and HRM. Table 6.17 shows some of the structural requirements for the super-structure ETPD (engineering task of product development), which were derived from the formal design elements, as identified in the last column of this table. For the structural requirements that are not present in all project tasks to be modelled by this super-structure, an input "On/Off" has been used. This suggests that the super-structure will contain a sub-structure implementing the particular feature, which can be activated or dis-activated through a switch parameter in the model. This modelling strategy is reported in recent System Dynamics literature (e.g. Coyle 1996).

Structural Requirements for Generic Structure	**ETPD**	**Design Element**
1. Processes incorporated		Table 6.7
(a) engineering	Yes	
(b) management	No.	
(c) human resources	No.	
2. Links to other tasks	—	
Engineering links (product flow)	—	Table 6.8
(a) One link to successor task containing	Always	<
(i) technical requirements	Always	
(ii) Work quality	On/Off	
(iii) QA effects	On/Off	
(iv) Rework	On/Off	
(b) One link from predecessor task containing	Always	
(i) technical requirements	Always	
(ii) Work quality	Always	
(iii) QA effects	On/Off	
(iv) Rework	On/Off	
Management links (control flow)	—	Tables 6.9/6.10
(a) one management-to-management link containing	No.	
... (contents of management link)	...	
(b) one engineering-to-management link containing	Always	
... (contents of management link) ...	...	
3. Control decisions		Table 6.11
... (list of control decisions considered)	...	...
4. Internal feedback loops and exogenous factors		Table 6.12
... (list of feedback loops and exogenous factors)	...	...

Table 6.17 – Example of a table specifying the structural requirements of a generic super-structure (ETPD), derived from the design elements.

The information in this table will guide the implementation of the ETPD super-structure. For example, the modeller can see that it will have to contain one engineering process of work accomplishment, which will simulate the product development process (by definition, ETPD refers to product development). It will always have "a link" to a successor task, specifying technical requirements available – note that the concept of "link" does not imply the presence of a specific "object" in the structure; rather, it is an abstract concept indicating that this task will produce information *to be stored in a variable*, which will be made available to the successor tasks. This link may or not contain information regarding the quality of the sub-product being delivered – the "On/Off" entry indicates that there will be a switch parameter activating or dis-activating this information content. The efficiency of the QA activity within this task may, in some cases, either affect or depend upon the efficiency of the QA activity of the successor task (e.g. discovering defects in the successor task may increase the QA efficiency of this task). Rework effects may or not be present in this link. By reading through this table, the modeller knows what are the effects that will have to be considered in the "link" to a successor task, and which of these will need to be selected using a switch parameter.

The purpose of this task is to produce a table like this specifying the structural requirements for each super-structure identified in table 6.7 – these requirements should be derived from and referenced to the formal design elements. Once these tables have been developed for each super-structure, the next task is to implement them individually.

Implementation of super-structures

The overall logic of this task is described in algorithm 6.2 as follows:

```
Specify <super-structure reference scenario>
FOR each step in <super-structure implementation process> DO
        Implement and validate (stage 1) <super-structure> using:
                <generic feedback structure (figure 6.4)>,
                <super-structure requirements>
                <confidence tests for stage 1 (table 6.16)>
        Validate (stage 2) <super-structure> using
                <super-structure reference scenario>,
                <super-structure requirements>,
                <confidence tests for stage 2 (table 6.16)>
ENDFOR
Validate (stage 3) <super-structure> using:
        <super-structure reference scenario>,
        <super-structure requirements>,
        <confidence tests for stage 3 (table 6.16)>
```

The implementation of a super-structure is considered as being a sequence of steps. The process is supported by validation tests, which are carried out at the end of each implementation step (i.e. validation stages 1 and 2, in table 6.16) and at the end of the process (validation stage 3, in table 6.16). These tests are presented in table 6.18 (extracted from table 6.16).

Validation stage	Direct structure tests	Structure oriented behaviour tests
1. During the implementation steps of a super-structure	*Advised:* • All	*Advised:* • Partial model testing
2. At the end of each implementation step of a super-structure	NA.	*Advised:* • Qualitative features – tolerant • Extreme condition – tolerant • Boundary adequacy – new added structures *Useful:* • Behaviour sensitivity – tolerant • Phase relationship – tolerant
3. At the end of the full development of a super-structure	NA.	*Advised:* • Qualitative features – strict • Extreme condition – strict • Behaviour sensitivity – strict • Phase relationship – strict Useful: • Boundary adequacy – new added structures

Table 6.18 – Validation tests to be carried out during the implementation of a super-structure, in stage 3 of model implementation

The example here presented will not include the quantification and calibration of the SD structure, and so the super-structure reference scenario is not proposed here. Also for the sake of simplicity, specific validation tests are not presented in detail.

The implementation of a super-structure includes: (i) creating and linking the various model's variables, and (ii) formulating mathematical equations for each of the variables. Unless the modeller is simply re-using structures previously developed, this process cannot be automated. Instead, it will be a creative process where the modeller will often use subjective input (e.g. from interviews with managers and staff involved in the project). The structured approach here exemplified is aimed at illustrating how the SYDPIM method can be used in order to bring discipline into this process.

The approach consists of a sequence of steps based on the generic project feedback structure described in figure 6.4. The steps to follow will vary slightly for each of the three types of elementary tasks proposed in the SYDPIM method:

(1) *engineering tasks* – these structures simulate product development type of work, being accomplished by staff allocated to the task. The feedback loops that take place within these structures are of the types E1, E2, and E3. The last two loops

represent staff adjusting their behaviour to try accomplishing the work within the specified targets. No managerial decisions are generated within this structure, although the managerial feedback loops ME1, ME2 and ME3 will pass through it.

(2) *management tasks* – these structures "mimic" the management activities of progress monitoring and generation of reactive control decisions. The feedback loops that take place within this task are of the types M1 (target adjustment) and M2 (policy adjustment). The control decisions generated within these structures will originate the feedback loops of types ME1, ME2, and ME3, when they are linked to other engineering tasks.

(3) *human resource management tasks* – these structures "mimic" the processes of staff hiring, firing, and other types of staffing activities with the outside world. Although the generic feedback structure of these tasks is not explicitly represented in detail in figure 6.4, the process of changing the staff level in the project is incorporated in the concepts "Actual Management Decisions" and "Staff Allocation". System Dynamics structures of staff hiring/firing applied to projects and to other organisational systems abound in the literature (e.g. Wolstenholme 1990, Coyle 1996, Abdel-Hamid 1989, Richardson and Pugh 1981). The main type of feedback loops that take place within these structures consist of adjusting the physical staff levels to the desired targets.

Implementing each of these three types of elementary SYDPIM tasks, based on the structural requirements produced, will call for slightly different approach, given the different nature of what is being modelled in each task. In this example, an engineering task is being considered.

Engineering tasks simulate staff carrying out the following main four activities: (1) work accomplishment, (2) monitoring progress against exogenous targets, (3) reporting progress to external management, and (4) adjusting their behaviour in order to achieve the targets. The particular way in which this generic process takes place in the real world depends on the specific project situation. In order to implement the SD structure that simulates these activities, a sequence of three steps is proposed:

(1) implementation of physical processes;
(2) implementation of E1 type of feedback loops and ExE2 exogenous factors;

(3) implementation of E2 and E3 type of feedback loops and ExE1 exogenous factors.

This sequence of steps is based on the generic project feedback structure proposed by SYDPIM (figure 6.4).

In each of these steps the following modelling activities take place: conceptualisation of the "level/rate" structure, quantification and validation tests (some supported by simulation runs). For the sake of simplicity, in this illustrative example only the conceptualisation of the of "level/rate" structure is presented. The implementation of each of these steps is now described separately.

(1) Implementation of physical processes

All management systems comprise a ground of "physical" processes upon which management control is implemented. This typically consists of flows of physical or abstract entities, moving throughout a sequence of states (i.e. possibly referred to as "entity life-cycle"). In an engineering task, the physical processes typically consist of inter-related flows of work units and of other work-related entities (e.g. defects). In the SD conceptualisation process the relevant entities are first listed, and then their life-cycles are identified and linked (Wolstenholme 1990) – ideally, the structure should not include any feedback loop; in practice, the exception may be the basic controls necessary to preserve non-negative stocks. The structure will include mainly "level" and "flow-rate" variables. Where required, exogenous parameters are used temporarily as auxiliaries, to specify the values of the rates. Other auxiliary variables can also be used to establish the "physical links" between the life-cycles of the entities. The simulation of this structure will implement a "feedback-absent" process of work being accomplished – somewhat like a pure mechanical system. The variables in the model represent the open-loop elements of the concepts "real work project status" and "work accomplishment", in the SYDPIM generic feedback structure (figure 6.4).

A simple example of implementing this step is shown in the figure 6.15. This SD structure captures the physical processes that take place at the core of an engineering task. Two flows of "tasks" and of "defects" are considered, each of these two entities having its own life-cycle. These life-cycles are inter-related through the use of auxiliaries and direct relationships between rates and levels. Four exogenous parameters are used temporarily as auxiliaries: "likely work rate", "likely review and rework rate", "actuality quality of work", and "actual quality of reviews". These are *temporary* exogenous variables because it is clear that they will not represent factors exogenous to the project in the final model. Instead, they are likely to be replaced by "endogenous" variables in the following implementation steps.

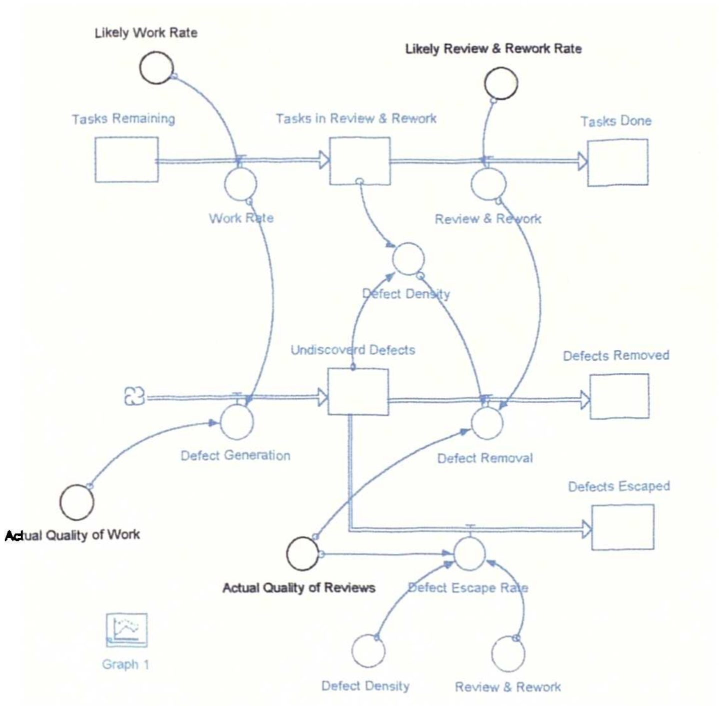


Figure 6.15 – "Physical processes" of an engineering task super-structure at the end of the first step of implementation

(2) Implementation of E1 type of feedback loops and ExE2 exogenous factors

The first level of feedback in an engineering task is of "non-human" nature. The physical state of the system naturally affects its own rate of change through "natural rules". These rules are "intrinsic" to the system: for example, the more the work complete, the more the work that can be started. These loops were classified in figure 6.4 as E1-type of feedback loops. They can be identified in table 6.17 where the requirement so the super-structure are specified. In order for these feedback loops to be captured, the "levels" (i.e. the "system state") will have to affect the "rates". The links may be implemented directly, but often intermediate endogenous auxiliary variables are used, for the sake of clarity. These auxiliaries represent indirect information about the "system state", derived from the information in the levels (i.e. the "raw information"). Furthermore, often the effects of the system state on the rates is more conveniently quantified through "table-functions", rather than through analytical formulations – in SD models, table-functions tend to play an important role in calibration; for this reason, and when appropriate, they will be regarded as exogenous inputs.

The figure 6.16 shows the result of imposing E1-type of feedback loops over the "physical processes" structure developed in the previous step – the new additions are in red. Three feedback loops were introduced:

- work completed determines how much work can proceed. This models the technical feasibility issues of the development process;
- the defect density of the work completed affects the quality of the work being done (both development and reviews). This reflects poor quality of work completed misguiding accomplishment of new work.

The implementation of these loops will introduce new endogenous auxiliaries and new relationships in the model structure. Some of the temporary exogenous variables from the previous step will be replaced by these auxiliaries – for example, the exogenous variables "likely review and work rate" was replaced by the endogenous variable "potential review and work rate". New exogenous parameters and table-functions will also be created – for example, " nominal effort required per defect" and "effect of defect density on quality". At the end of this step, exogenous parameters can be of three types: (i) temporary "place-holders " to be replaced by endogenous auxiliaries (to be introduced in later steps as new feedback loops are captured; e.g. "potential work rate"), (ii) links to other project tasks (these will become endogenous when the full model is assembled), or (iii) factors genuinely exogenous to the final project model structure (these can represent either external factors or intrinsic properties of the system; see chapter 2 for a discussion). The last category represents the ExE2 type of exogenous factors identified in figure 6.4 and are to be identified for the super-structure being developed in table 6.17. Exogenous variables in (2) should also be verified against the links identified in this table. At the end of this step, the model structure will capture the physical process of work accomplishment ruled by E1-type of "non-human" feedback, possibly influenced by some exogenous factors.

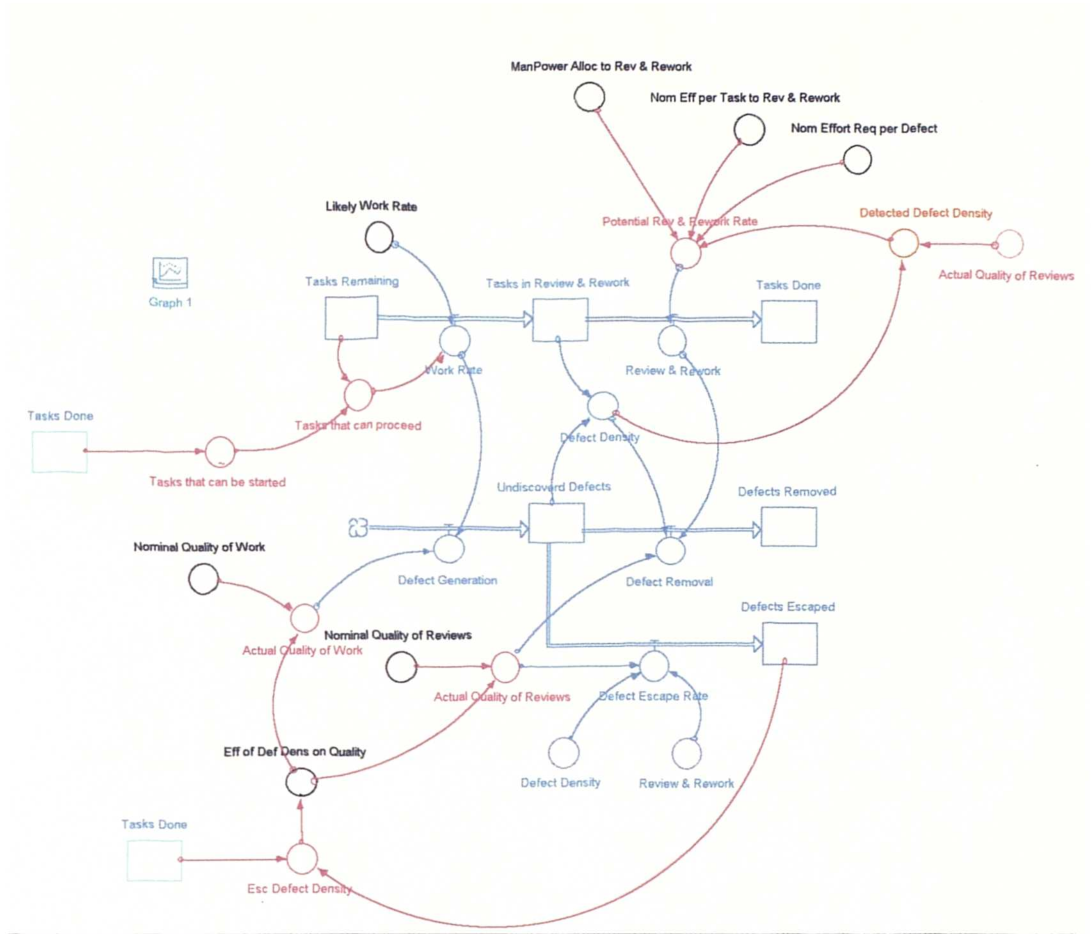


Figure 6.16 – Adding "non-human" feedback to an engineering task at the end of the second step of implementation

<u>(3) Implementation of E2 and E3 type of feedback loops and ExE1 exogenous factors</u>

The second level of feedback in an engineering task is of both human and physical nature, but still "non-managerial". This type of feedback results from the interactions between the staff and the physical state of the system. Staff will react to the way in which the work progresses, but without generating explicit management decisions. For example, the staff will naturally spread their working-hours throughout several working activities, depending on the work available to be done in each activity – e.g. if a large amount of design documents is awaiting review, designers will naturally tend to dedicate more hours to this activity. These loops were classified in figure 6.4

as E2-type of feedback loops: the work status affects the staff behaviour, which in turn affects the staff status; the staff status will then affect the rate of work accomplishment which in turn affects the work status. Again, this type of loops to be captured in the super-structure being developed can be identified in table 6.17.

Figure 6.17 shows the result of implementing E2-type of feedback loops over the structure from the previous step – new additions are in green. The feedback loops added considers that staff will split their effort between the two activities "development" and "review and rework", depending on how many tasks are available for development ("work to be done that can proceed") and on the priority given to this activity ("weight to work to do") – note that for a matter of simplification it is being assumed in this structure that the staff adjusts instantaneously their allocation of effort between these two activities; in reality, and depending on the situation, a delay may be considered.

In order for these feedback loops to be captured in the structure, new endogenous variables may need be introduced in the model to represent: the staff perceptions of the system status (which can differ from the "real" system status), the staff behaviour changes and the staff status. Once more, these and other variables will possibly replace existing exogenous variables being used temporarily (e.g. "likely work rate" replaced by "potential work rate").

The staff behaviour changes are not driven only by the need to accomplish work. As the staff change their status, over-time this will affect their own tendency to further change their behaviour in the future. For example, working over-time in order to accelerate the progress rate will also increase "staff fatigue" and may reduce "staff morale" – these type of staff conditions are referred to in figure 6.4 as "Staff Status". As the fatigue level increases, staff is less likely to work further over-time. This type of feedback loops, representing the interactions between staff behaviour and staff status, are classified in figure 6.4 as E3-type of loops. This is the third and final level of feedback that takes place within an engineering task, and is solely of human nature. The introduction of this type of feedback loops in the SD structure follows naturally from the E2 type of feedback loops, by linking back the effects of staff status on their "proneness" to change behaviour. The introduction of E3-type of feedback loops may lead to the creation of new variables that represent aspects of the staff behaviour and staff status. Occasionally, this may in turn lead to more E2-type of feedback loops being introduced.

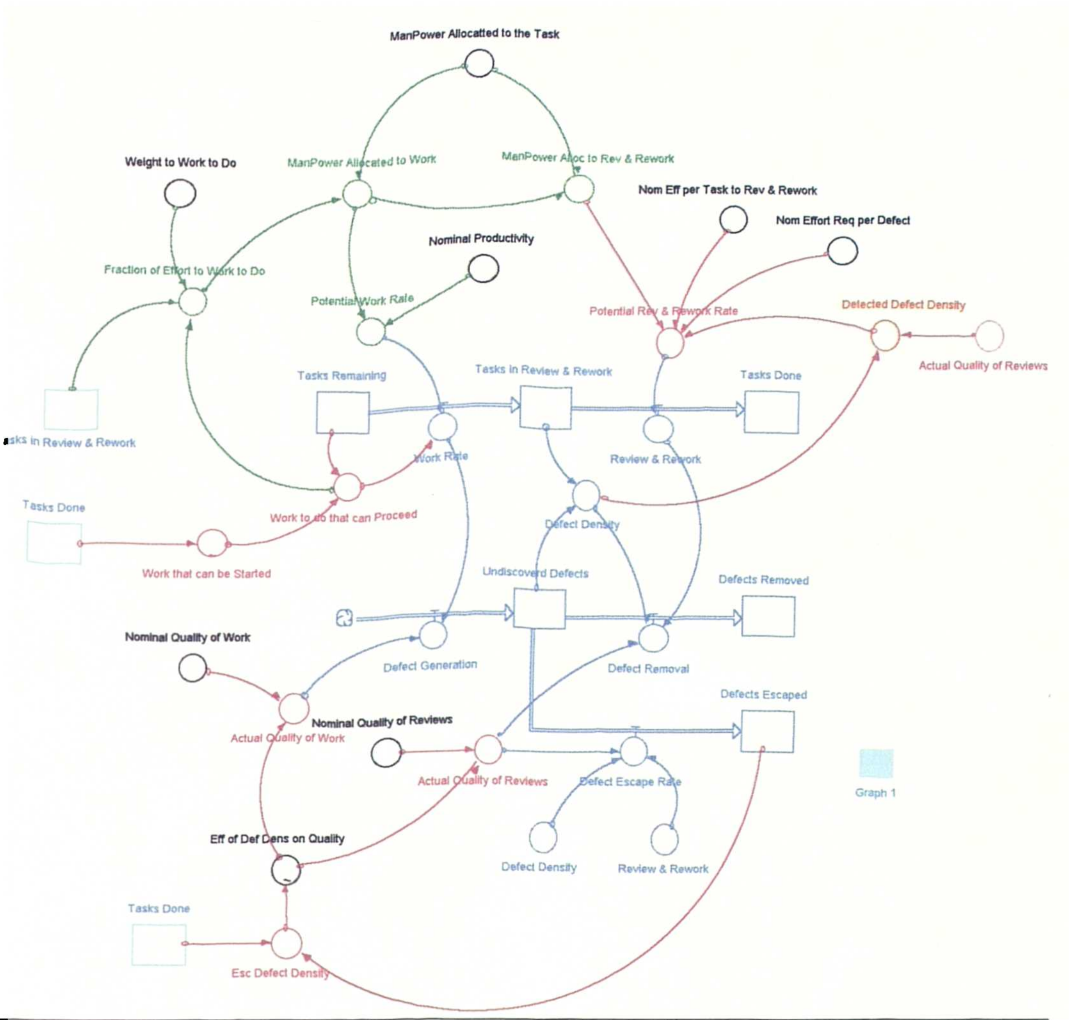


Figure 6.17 – Structure of the engineering task at the end of introducing E2 type of feedback loops

Figure 6.18 shows the result of adding more E2-type of feedback loops (e.g. the staff use of over-time as a reaction to schedule delay; the fatigue from over-time affecting the quality of the work being accomplished), and adding E3 feedback loops (in pink): for example, the use of over-time causes fatigue, which in turn reduces the staff willingness to use more over-time.

With the introduction of E2 and E3 types of feedback loops, exogenous variables from the previous step may be replaced and once again new exogenous parameters and table-functions may be created. However, at the end of this step the exogenous parameters in the model should only be of the two following types: (1) links to other project tasks (these will become endogenous when the full model is assembled) (e.g. variables in grey in figure 6.18 above), or (2) factors genuinely exogenous to the final project model (e.g. variables in black and preceded by "Ex" in figure 6.18 above). The exogenous factors created in this step should be of type ExE1, representing external factors or intrinsic properties of the system affecting staff behaviour – again, these can be identified for the super-structure being developed in table 6.17. The exogenous variables of type (1) should also be verified against the links identified in this table, as they must correspond to the information contents of these links.

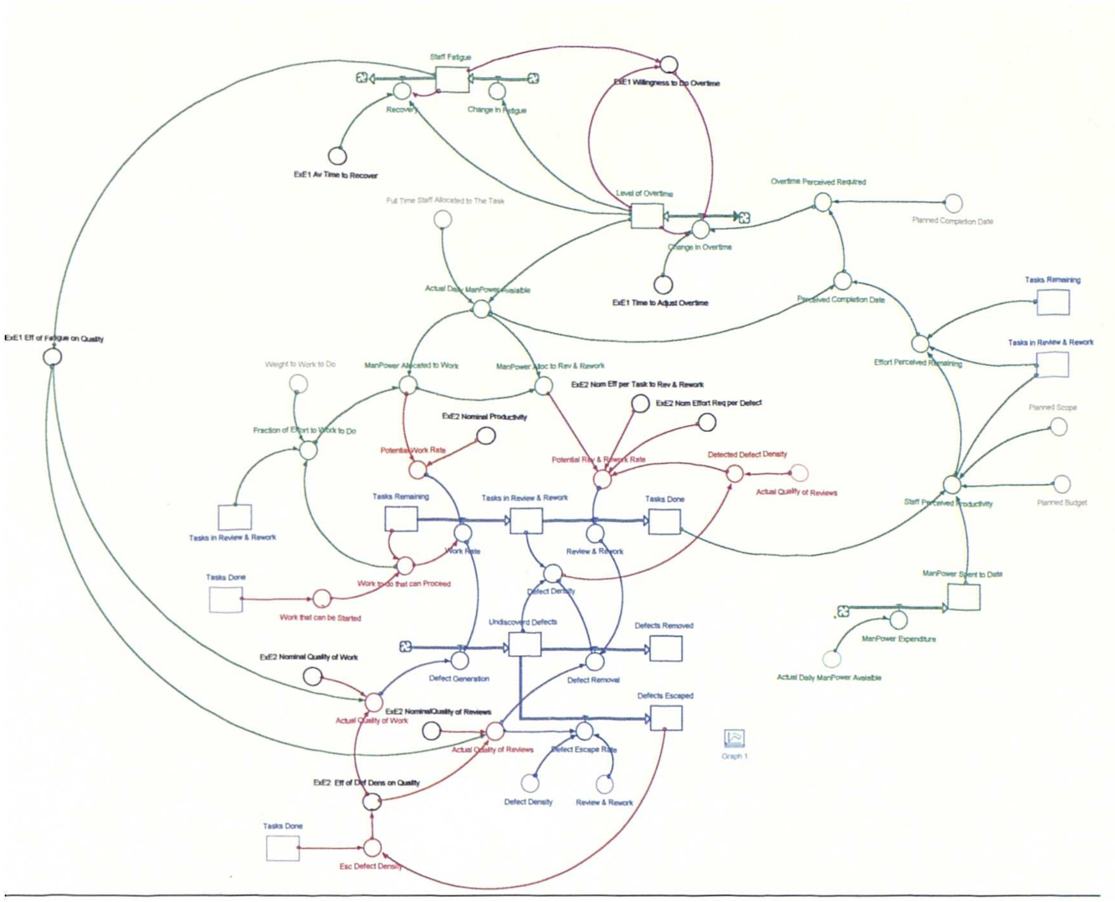

Staff Fatigue
Recovery
Change in Fatigue
ExE1 Willingness to Do Overtime
ExE1 Av Time to Recover
Full Time Staff Allocated to The Task
Level of Overtime
Overtime Perceived Required
Planned Completion Date
Change in Overtime
ExE1 Time to Adjust Overtime
Actual Daily ManPower Available
Perceived Completion Date
Tasks Remaining
ExE1 Eff of Fatigue on Quality
Tasks in Review & Rework
Weight to Work to Do
ManPower Allocated to Work
ManPower Alloc to Rev & Rework
ExE2 Nom Eff per Task to Rev & Rework
Effort Perceived Remaining
ExE2 Nom Effort Req per Defect
ExE2 Nominal Productivity
Planned Scope
Fraction of Effort to Work to Do
Potential Work Rate
Potential Rev & Rework Rate
Detected Defect Density
Actual Quality of Reviews
Tasks Remaining
Tasks in Review & Rework
Tasks Done
Staff Perceived Productivity
Planned Budget
Tasks in Review & Rework
Work Rate
Review & Rework
Tasks Done
Work to do that can Proceed
Defect Density
ManPower Spent to Date
Work that can be Started
Undiscovered Defects
Defects Removed
ManPower Expenditure
ExE2 Nominal Quality of Work
Defect Generation
Defect Removal
Actual Daily ManPower Available
Defects Escaped
ExE2 NominalQuality of Reviews
Actual Quality of Work
Actual Quality of Reviews
Defect Escape Rate
Graph 1
ExE2 Eff of Def Dens on Quality
Defect Density
Review & Rework
Tasks Done
Esc Defect Density

Figure 6.18 – Example of an engineering task super-structure at the end of the third and last step of implementation

At the end of this step, the super-structure will capture all the endogenous aspects of work accomplishment and staff behaviour within an engineering task. The structure will also contain exogenous factors that represent forces external to the project, intrinsic properties of the project system, and linkages to other engineering or management tasks.

Overview of validation

The aim of the SYDPIM implementation approach is to evolve the model structure gradually while being continuously tested for validation. As previously mentioned, in this example no specific validation tests were illustrated and the super-structure was not quantified. However, in the actual implementation of the SYDPIM method in the real world, validation tests would be carried out during and at the end of implementing a super-structure, according to the SYDPIM validation framework proposed – see table 6.18. While the specific tests which would be appropriate to conduct would depend on many factors specific to the situation, some guidelines are here presented, based on the SYSPIM validation framework, to illustrate how this should be implemented in practice.

As direct structure tests, at the end of each implementation step it is important to check whether the structural requirements identified in table 6.17 are being captured in the structure being developed. Of particular importance is the inclusion of all feedback loops. In checking these feedback loops, it is also important to note that model implementation takes place at the equation level, which in most situations will be more dis-aggregated than the one assumed in the feedback loops identified in design. Therefore, the loops identified in table 6.17 will aggregate and will be captured through various loops at the equation level in the model structure.

Regarding the behaviour tests, at the end of each implementation step structured oriented behaviour tests are carried out. These tests are based on the specification of a "super-structure reference scenario", against which the SD structure is being calibrated. Preferably, this scenario should specify: (i) initial conditions (e.g. work to be accomplished, man-power available over-time), (ii) targets for the expected outcome (e.g. duration, cumulative man-power), (iii) a set of possible modes of behaviour derived empirically, which *must* include a steady mode of behaviour, where progress evolves steadily towards the planned targets, and (iv) some required estimates for process metrics (e.g. productivity). The confidence tests to be carried out will question the three main dimensions of model validity: (a) the structural logic, (b) the mathematical formulations of the relationships in the model, and (c) the values of the exogenous parameters. The causes for the model's inability to pass behaviour reproduction tests can be found in any of these three areas. It is suggested in SYDPIM that these dimensions should be questioned in this sequence, although the process may iterate several times. As a guideline to conduct the behaviour tests based on a reference scenario, the following sequence of steps is proposed:

(1) *reproduction of reference scenario* – adding new variables and relationships forces the whole structure to be re-calibrated. This is done by adjusting the values of existing exogenous parameters (including table-functions), and by estimating new values for the new parameters. Where the calibration for the reference scenario requires unlikely parameter values, or an acceptable match cannot be achieved, the current structure is not passing this test and needs to be revised;

(2) *test minor variations* – once the reference scenario is being reproduced well, in particular the steady mode of behaviour, the next step is to subject the super-structure to changes in the parameter values. Changes of great magnitude can lead to dramatic changes in the outcome produced, and the modeller may have difficulty in understanding the reasons. Therefore, in this second step, minor parameter changes are tested and checked against variations in the new results produced – i.e. what is causing what to change? To carry out this diagnosis, the modeller should consider to the list of feedback loops already implemented in the structure. If plausible explanations cannot be found to justify the behaviour changes, then this test is not passed and the structure needs to be revised;

(3) *"Hyperbolic" testing* – at this stage the modeller should understand how the various exogenous parameters influence the feedback loops that drive behaviour. This final step is aimed at questioning the logic of the feedback structure, in a more radical manner. Extreme but yet theoretically possible values are considered for the parameters (both above and below likely values). It must still be possible to explain the new behaviour produced by the structure in a logical and consistent manner. This is, if these extreme conditions would ever occur, the likely dramatic outcome would probably be the one produced by the SD structure. If the structure is not capable of explaining this extreme scenario in a plausible manner, then it is likely to have flaws and needs to be revised.

These behaviour tests are being implemented into an engineering super-structure. This means that the behaviour being produced by this structure represents work being carried out in conditions of no managerial control – i.e. status is not being monitored against targets and no corrective actions are being generated. This must be taken into account when assessing the plausibility of the behaviour produced in the three steps above – deviations from targets are not "corrected", attenuated or aggravated by management actions. In step (1), the calibration for the steady mode of behaviour represents a situation where targets are achieved with no need of managerial corrections.

At the end of each implementation step, new variables and relationships are added to the model. While at any step the confidence tests will always question the whole model structure, the focus should be on the new elements added during the last step.

Some considerations for implementing management and HRM super-structures

The example here presented illustrated how the SYDPIM method can be used to develop an engineering type of super-structure (first step of stage 3; see figure 6.12). The process followed was based on the SYDPIM generic feedback structure (figure 6.4), where the various types of feedback loops were progressively incorporated into the model structure.

A similar approach can be taken to implement the super-structure of management tasks. In this case, the following sequence of steps is proposed, where the formal design elements would specify the specific loops to be captured (summarised like in table 6.17):
(1) implementation of M1 loops and EXM1 and EXM2 factors;
(2) implementation of M2 loops.

The validation of a management super-structure based on behaviour tests focuses on the ability of the structure to reproduce management reactive decisions in face of perceived deviations from targets. It is therefore easier and more appropriate to validate this type of structure by linking it to an engineering task. This task will provide continuous status information and will react to management decisions. To implement this approach, an engineering super-structure must be available. The same sequence of steps proposed above for an engineering super-structure can be followed, this time focusing on the behaviour of the management process. In steps (2) and (3), disturbances are introduced so that deviations from targets occur and are detected by the management component. This tests how this component perceives the deviations (i.e. magnitude, delay), and how the corrective decisions generated, such as using schedule pressure, re-adjusting the schedule, using over-time, or skipping QA. The behaviour produced by the model should reflect the managerial policies: for example, if a policy of minimising schedule slippage was considered then the model should react to delays by using schedule pressure and over-time.

Regarding HRM type of tasks, the generic feedback structure of figure 6.14 does not specify in detail the various types of feedback loops that can take place within these tasks – although E3-type of loops are very likely to take place. However, this type of SD structure is very common in organisational systems and it has been widely modelled in many types of SD models. Various existing SD structures of this type can be found in the literature (e.g. Abdel-Hamid and Madnick 1991, Ford 1995). The validation of this type of structure based on behaviour tests can also be enhanced by linking it to a management task, which in turn is linked to an engineering task. The three steps proposed for the engineering tasks can also be followed. The "targets" of an HRM task can be considered as a pre-planned staff profile.

Stages 3b, 4 and 5: tailoring of super-structures, integration and final validation

The example presented does not cover the full life-cycle of the SYDPIM Model Development Method. As shown in figure 6.12, once all the required super-structures identified in table 6.7 are developed, the second step in stage 3 is it to replicate and tailor these to represent the various tasks in the model architecture. In the following stage 4, these SD structures are integrated into the final model. SYDPIM proposes a progressive integration and validation process, as shown in figure 6.14. As management tasks and engineering tasks are integrated, ME1, ME2 and ME3 type of feedback loops start being created within the model structure.

Once more, the generic feedback structure of figure 6.14 can be used to guide this process. After the model is integrated, the final stage 5 five is exclusively aimed at validation on the basis of behaviour accuracy tests, preferably using real project scenarios (e.g. from the project past, if available).

Summary

In this section the second major phase of the SYDPIM Model Development Method was presented. In this phase, the SD project model is implemented and quantified at the equation level. In practice, this will be done using a SD modelling tool, where typically a "level/rate" type of diagram is developed to represent the model logical structure.

While the specific simulation language / tool used can restrict the features that can be incorporated in the model, the most popular modelling support the requirements assumed in the SYDPIM method: visual interface to develop a "level/rate" diagram, replication of sub-structures, basic mathematical features (e.g. delays functions) and performance.

The SYDPIM Model Development Method proposes a life-cycle process of continuous implementation and validation. This process is based on five major principles: (1) project tasks can be implemented and partially validated individually, (2) some project tasks are similar and hence can be modelled by a common generic structure (referred to as super-structure); these generic structures are replicated tailored to the specific task, (3) generic super-structures can be of three types, ENG, MAN and HRM, each type calling for a slightly different implementation process, (4) the conceptualisation of the SD structures is carried our progressively, incorporating more feedback loops, based on the generic project feedback structure proposed (figure 6.4), and (5) the validation of the generic super-structures, project tasks and of the whole project is based on the specification of reference scenarios.

The life cycle process of SYDIPM implementation comprises three major stages: component development (stage 3), model integration (stage 4) and final model validation (stage 5). The first stage includes two different steps: implementation of generic super-structures (stage 3a), and tailoring of these structures to the specific project tasks in the model architecture (stage 3b). Model integration takes place progressively, where intermediate sub-networks of tasks evolve towards the final model. The final model validation stage focuses on behaviour accuracy tests, where the model will have to reproduce project scenarios. Validation also takes place continuously throughout the life-cycle. The formal design elements developed in the previous phase are used in three stages to provide important input information and to support modelling decisions.

In order to integrate validation throughout the implementation process, SYDPIM proposes a validation framework. It is assumed that not all confidence tests are appropriate to all stages of the implementation process. Therefore, the SYDPIM framework proposes specific types of confidence tests for each stage of the implementation process. As a result, seven validation stages are proposed. This framework builds upon the most relevant existing validation approaches available in the literature. The specific tests to be carried during the actual implementation of a project model depend on the specific aspects of the modelling situation (data available, level of detail considered in the model, model structure). The SYDPIM validation framework therefore does not proposes specific tests. However, it provides a structured process and identifies the specific types of tests to be carried out, which should be used to generate the specific tests.

The integration of the validation framework into the implementation life-cycle process is presented as a structure and formal process for this phase of the SYDPIM Model Development Method. This formal process is specified through a logical algorithm, which should be used to guide the implementation of this phase in practice. The design elements developed in the previous phase play a key role in the process. The actual SYDPIM implementation and validation of a project model depends on many specific aspects of the project and modelling situation. For illustrative purposes, a generic example is provided, showing how a generic super-structure of an engineering task can be developed: its "level/rate" structure was progressively developed, increasingly incorporating more feedback loops; this process is based on the generic project feedback structure proposed by SYDPIM. An example covering the whole implementation life-cycle of the SYDPIM method would be overly extensive and as complex as developing a SD project model for a real project. The generic example here provided is illustrative and covers only part of stage 3. It results from a simplification of work carried out in the case-study project of this research. Some considerations of how validation would be carried out, about the other two types of generic super-structures (i.e. management and HRM) and about the other stages of the SYDPIM implementation process are presented.

While in practice modellers often tend to start the modelling process in this phase, the SYDPIM method argues in favour of a good model design, which will then support the whole implementation process. This approach helps in validation and provides a robust basis from which the modeller can understand the model and diagnose the causes of behaviour. The aim is to achieve a well documented quality project model, which can be applied in practice with a full understanding of the estimates and insights it provides. It is argued in this research that a well structured process like this is essential to achieve this goal, which is critical to the practical credibility of the approach and overall SYDPIM methodology.

6.7 Final overview discussion

The SYDPIM Model Development Method provides a new formal framework for the development of a SD project model aimed at being used in practice. In particular, the method is intended to ensure that the model gathers the requisites to be used within the SYDPIM Project Management Method, which will be described in the next chapter.

The SYDPIM Model Development Method is grounded on the good principles of the life-cycle approach, assuming a well ordered and logical sequence of phases, as well as continuous development activities. While a structured approach, the method is also flexible and can be implemented in an iterative manner, if required.

Some of the elements incorporated in the SYDPIM method were present in the past developments of SD project models reviewed in this chapter, although in most cases not in an explicit manner. An exhaustive analysis of these past developments was carried out to extract the most valuable elements that have proven of practical use. The SYDPIM method also proposes new elements not attempted before. The major novelties of the method are: (1) an explicit design phase, with the development of formal design elements prior to model quantification, (2) a generic project feedback structure, which is used to guide the overall modelling process, (3) a structured validation framework, comprising various stages and integrated into the implementation process, and (4) an explicit overall life-cycle process, comprising a sequence of well defined phases and stages.

The primary aim of the SYDPIM Model Development Method is to support the full implementation of the SYDPIM methodology, delivering the required SD project model. Nevertheless, the method can also be used to develop a project model to be used for another purpose (e.g. post mortem analysis of a past project).

Like any other formal technique, there are critical issues of practical implementation, which must be carefully addressed by any organisation intending to implement the method. The major potential difficulty has to do with the effort required to implement the design phase of the method, developing the formal deign elements, and then ensuring that these designs are kept updated throughout the implementation and validation phase. Ultimately, this problem has to do with the overall need to maintain the discipline required and imposed by the method (certainly, an issue critical to the implementation of any formal structured method). On the one and, if too much design effort is required, this can compromise the timeliness and usefulness of the practical application of the model. On the other hand, if the design elements are poorly specified or are not kept updated, they can mislead the implementation phase and might not be of any practical help. In both cases, the situation may lead to the method being abandoned half-way.

There are three issues that must be taken into account in order to approach this potential problem: (1) the availability of a technological platform, (2) the gradual implementation of the method at different levels of formality, and (3) the explicit assumption of a trade-off between the model quality and the extra development effort and time required. A software based technological platform has a strong potential to accelerate the implementation of the method, reduce dramatically the effort required, and reduce the number of possible mistakes. A gradual implementation process allows for an organisation to progressively learn and gain experience with the overall method and associated critical issues. This allows for the filtering of what is relevant and for the identification of problems and respective solutions. Ultimately, this continuous learning process may lead to a method tailored to the specific needs, characteristics and constraints of the organisation. As mentioned, while the method is structured, it is essential to consider its implementation in a flexible and adaptable manner. This implementation must also be supported by the explicit assumption that more effort and time is being dedicated to the process, in order to achieve a higher quality model.

The ideal of a technological platform to implement the SYDPIM method, would be a SD modelling tool capable of supporting the implementation of the design phase in an integrated manner with implementation. The features of this tool would include the creation of the formal design elements during the design phase, as well as their update throughout the implementation phase. The consistency among the design elements would be automatically checked and detected by this tool. In some cases, inconsistencies could even be automatically eliminated. Similarly, the consistency between the quantified SD model and the design elements would also be checked automatically by the tool. Possibly, some types of changes in the SD model would be automatically updated in the formal design elements, and vice-versa. In other cases, inconsistencies would be detected and the update would require human intervention.

Unfortunately, such a comprehensive tool does not exist in the market. Such development was not the purpose of this research, but it certainly constitutes a most useful next step for the future.

It is here recommended that a gradual implementation of the SYDPIM method is followed, in order to introduce the method in an organisation. It is important to note that while the method was tested and inspired by a real project environment, its characteristics are influenced by the specific organisation wherein the case-study project took place. Therefore, the implementation of the method in another organisation will probably require some adjustments. Further experiences with implementing the SYDPIM Model Development Method in real projects are in fact desirable, in order to improve its practicability and make it more generic and thereby easier to tailor to specific project organisations.

A gradual implementation implies that in early implementations, less formality is imposed in the modelling process and that not all of the elements of the method are considered. For example, the formal design elements "task classification", "control decisions" and "mapping of causal structure to model architecture" may not be considered. In this case, only the design elements "model architecture", and "task dependencies" would be implemented, identifying the project tasks and the way in which these would be linked. The SYDPIM method does not provide specific levels of formality nor a list of what elements should be implemented at each level. However, some guidelines are here provided. The level of formality has to do mainly with the strictness with which the rules and principles of the method are followed. For example, in a less formal implementation the model architecture may not consider an explicit classification of the project tasks into "engineering", "management" and "HRM". Similarly, instead of generic structures being developed and then being tailored to specific tasks, the SD structures for the project tasks in the model architecture would be developed directly. The decision of which rules and principles are to be followed depends on the specific aspects of the modelling situation (e.g. manpower available for the modelling effort, level of SD expertise, modelling experience, data / information availability, time required for the model to be ready to use, among others). The elements of the SYDPIM method which should be included in the modelling process are related to the level of formality. This decision also depends on the complexity of the model, and should also comply with the validity requirements of the model application. The more elements included, the more updating effort will be required in the implementation and validation phase. On the other hand, a much deeper understanding of the model will be achieved. A complex model to which a high level of validity is required will benefit from the implementation of all of the design elements. A gradual approach to implementing the SYDPIM method will allow the modeller to learn about the trade-off between the benefits and the extra effort required for higher levels of formality, and thereby to take these decisions on a practical basis.

Implementing the SYDPIM method implies a decision to allocate more effort and time to the modelling process, prior to using the model in practice. It is important that this decision is made in an explicit manner. The expected benefits are a greater quality of the project model, and should outweigh the "costs". The main benefit of a higher quality model is, literally, to prevent the disadvantages of a poor quality model. These disadvantages can be tremendous, when a model is required to address the practical needs of project management in the real world. A poor quality SD model will quickly provide unreasonable answers to real life scenarios. In a modelling context of low quality, "quick-fixes" tend to be used to solve the problem. The consequence is that, over-time, the model will evolve towards a cumbersome and confusing structure, which can hardly provide valid and useful insights – the author calls this the "a thousand equations fixing the other thousand" syndrome. Unfortunately, these situations do occur in reality, leading to models which are either abandoned or can only succeed if their practical application does not demand a certain level of validity (e.g. playing with SD modelling to raise awareness of the methodology). On the other hand, the extra time and effort required to implement the SYDPIM method must comply with the practical restrictions of the model application. In case time and effort are scarce resources, it is recommended that the scope of the model is adjusted to accommodate the implementation of the SYDPIM method. Prototyping and incremental development are appropriate solutions for this type of situation, where a model needs to be made available in a short period of time.

The SYDPIM Model Development Method was developed with the aim of supporting the case-study of this research, which in turn was aimed at field-testing and improving the SYDPIM Project Management Method. In the first place, a SD project mode was required. Secondly, the author realised that the practical usefulness of SYDPIM would be seriously constrained by an organisation's ability to develop the required project model. Since there was no structured framework or method available to support the development of such model, this has motivated the author to develop a structured method, as an integrative part of the SYDPIM methodology. An initial high level framework was outlined and was further refined throughout the case-study, having led to the structured method described in this chapter. The next chapter describes the major component of the SYDPIM methodology: the SYDPIM Project Management Method.