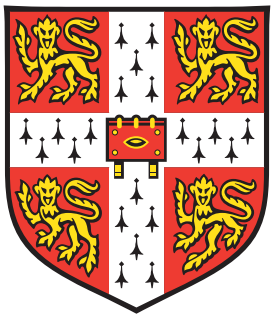
Model-Based Approaches to Support Process Improvement in Complex Product Development

A thesis submitted to the University of Cambridge
for the degree of Doctor of Philosophy



David Charles Wynn

Homerton College

February 28, 2007.

— To my parents —

Declaration

This thesis is the result of my own research and does not include the outcome of collaborative work, except where stated otherwise. The dissertation has not been submitted in whole or in part for consideration for any other degree.

This document contains fewer than 150 figures and fewer than 65,000 words.

David Charles Wynn

University of Cambridge

February 28, 2007.

Acknowledgements

I wish to express my gratitude to Professor P. John Clarkson for supervising this research and providing financial support from the IMRC block grant.

I would also like to thank all Rolls-Royce personnel who gave their time to support the project. Particular acknowledgement is due to Chris Powell and Neil Brown for their help during the eight-month case study.

Finally I would like to thank Seena Nair for her programming support and Dr. Claudia Eckert for her advice throughout the project.

Abstract

The performance of product development processes is important to the commercial success of new products. The improvement of these processes is thus a strategic imperative for many engineering companies — the aero-engine is one example of a complex product for which market pressures necessitate ever-shorter development times. This thesis argues that process modelling and simulation can support the improvement of complex product development processes.

A literature review identified that design process modelling is a well-established research area encompassing a diverse range of approaches. However, most existing tools and methods are not widely applied in industry. An extended case study was therefore conducted to explore the pragmatic utility of process modelling and simulation. It is argued that iteration is a key driver of design process behaviour which cannot be fully reflected in a mechanistic model. Understanding iteration can help select an appropriate representation for a given process domain and modelling objective.

A model-based approach to improve the management of iterative design processes was developed. This approach shows that design process simulation models can support practice despite their limited fidelity. The modelling and simulation framework resulting from this work was enhanced for application to a wider range of process improvement activities. A robust and extensible software platform was also developed. The framework and software tool have made significant contribution to research projects investigating process redesign, process robustness and process optimisation. These projects are discussed to validate the framework and tool and to highlight their applicability beyond the original approach. The research results were disseminated in academia and industry — 72 copies of the software were distributed following requests in the first three months of its release.

—— This page is intentionally blank ——

Contents

—— This page is intentionally blank ——

Chapter 1

Introduction

Today's multi-national manufacturing companies recognise that they must not only offer better products than the competition; they must also bring these products to market faster and more cost effectively than their rivals. For example, research from McKinsey and Company has indicated that releasing a product six months behind schedule can lead to the loss of over 30% of profits, whereas a product delivered on time but 50% over budget leads to losses of less than 5% (DTI, 1994; Figure 1.1). The scope of this study is limited to relatively simple products with short market lives. However, it does highlight that a company's understanding of the process by which products are developed and, ultimately, their ability to predict and control this process can have a substantial influence on profitability.

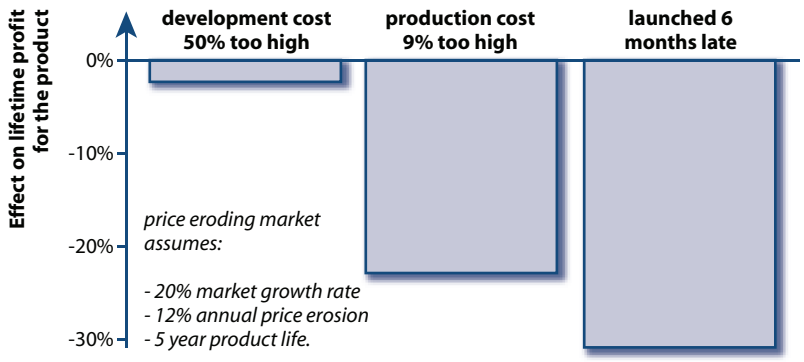


Figure 1.1 The timeliness of delivery to market can form a substantial influence on the lifetime profitability of a new product (DTI, 1994).

Bringing any new product to market is a challenging and expensive undertaking. For example, the design and development of a screwdriver requires the coordination of six people for twelve months (Ulrich and Eppinger, 2003). This is perhaps among the simplest of manufactured products, consisting of only a few components; near the other end of the spectrum, a helicopter comprises over 10,000 individual components (Eckert *et al.*, 2001). For engineered products of this complexity, development and manufacture requires the co-ordination of hundreds of companies and thousands of individuals. For example, Airbus maintains a procurement network of around 1500 suppliers in more than 30 countries, and development of the Boeing 777 passenger aircraft required the attention of around 17,000 personnel for up to four and a half years (Ulrich and Eppinger, 2003). These examples highlight that complex products tend to have complex development processes, which are consequently difficult to understand and improve.

This thesis is motivated by the commercial importance and practical difficulty of improving complex product development processes. It is argued that:

> The processes by which complex products are developed may be modelled using a formal approach. Such models may be used to support a range of process improvement activities.

This is based on the assumption that lessons learned from studying one product development process can be applied to improve another in a similar context, *e.g.*, the same company and product family. This assumption may be justified from practical experience and acceptance in the existing literature. Although meriting further research it will not be explored in this dissertation.

The argument is necessarily abstract to encompass the content of this dissertation. It is qualified further in the following four sections. Firstly, the research background is explored and its importance highlighted. Secondly, the argument is decomposed into two research questions and a main success criterion. Thirdly, the research methodology is discussed. Finally, the dissertation structure is summarised.

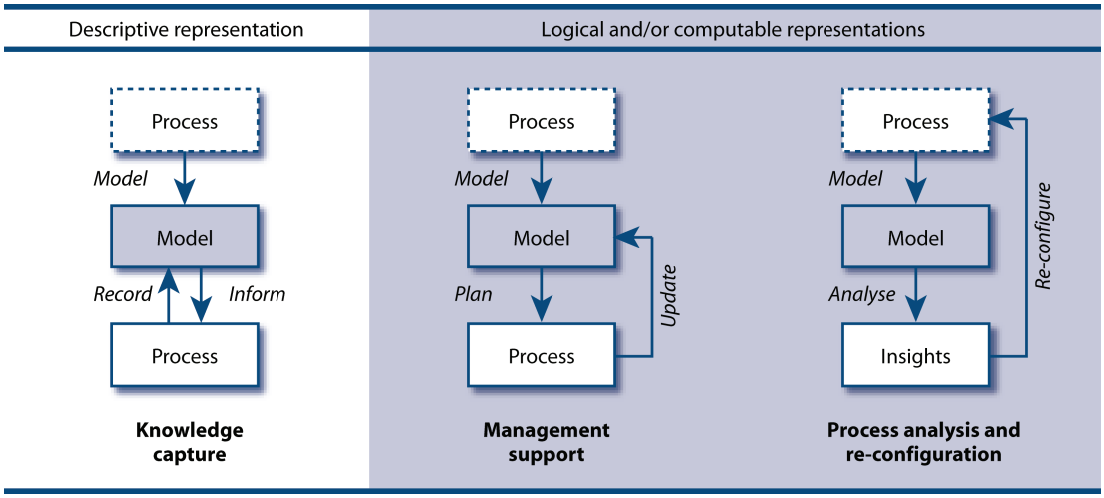


Figure 1.2 The applications of process modelling considered by this thesis. A dashed border indicates consideration of many instances and a solid border denotes a single instance.

1.1 Background

This section discusses the applications of process modelling which are considered in this thesis and introduces the key stakeholders in these activities. This allows more detailed discussion of the research motivation, *i.e.*, the novelty of the thesis and its importance to academic and industry practice.

1.1.1 Applications of process modelling

Process models may be constructed for many purposes and should be appropriate to their intended use (Browning and Ramasesh, 2007). This thesis concerns process modelling to support three forms of process improvement, focusing on the design aspects of complex product development (Figure 1.2). These are:

- **Knowledge capture.** Most organisations prescribe standard, high-level processes that aim to ensure good practices such as proper evaluation and review activities. Technical processes are often described using informal flowcharts, usually at a level of detail that fits on a presentation slide. Such descriptions highlight a particular perspective and understanding of the process and are of limited utility outside this context. Such documentation is nevertheless useful. It represents experts' process knowledge in a form which can support discussions, thus promoting co-ordination and develop-

ment of a shared understanding. It is often proposed that more integrated process capture could support co-ordination by making experts' knowledge more explicit and accessible (*e.g.*, Eckert and Clarkson 2003). Additionally, a specific design process model could document the design rationale which must be revisited if any re-design is necessary. This is a pertinent issue since many complex products have lifecycles measured in decades and must be supported after the original designers have left the company.

- **Management support.** In the context of this thesis, management is concerned with project planning, monitoring and control. Gantt charts are usually used to represent processes for these purposes. The utility of these documents can vary according to the process characteristics, company and industry sector. For example, processes driven by the iterative refinement of high performance components are characterised by uncertainty in task definition and ordering. They are thus difficult to represent in an acyclic Gantt chart. Planning documents typically show little detail in these cases. In contrast, Gantt charts representing more sequential processes can encompass thousands of tasks but provide little indication of the rationale underlying decisions such as task ordering and resource allocation. In both examples the inadequacy of existing representations can cause difficulties when reasoning about plans, schedules and risk. This dissertation will show how process models incorporating design iteration and uncertain events could form the basis of more effective management practice.

- **Process analysis and reconfiguration.** Companies must understand their design processes in order to identify and evaluate improvement opportunities. This could involve identifying where investment in new tools could provide most leverage, evaluating the impact of resourcing levels upon project delivery, or identifying configuration changes which could improve key measures of process performance. Such analysis can be difficult since individuals' perceptions of processes focus on their specific responsibilities and goals.

  Process modelling can support these improvement activities in many ways. For example, constructing a model helps develop a detailed, common understanding of the process and thereby allows the modeller to directly identify

problem areas (*e.g.*, Wynn *et al.* 2005). The structure of a process model may be analysed to highlight potential improvements in more complex cases (Eppinger, 1991). A simulation model capturing process behaviour can provide a 'virtual sandbox' in which alternative configurations may be explored and cost-effective improvements identified (O'Donovan, 2004). Simulation models may also be viewed as 'black boxes' for estimating the performance of a given process configuration. They could thus be used as the basis of a design process optimisation approach (Bell *et al.*, 2007).

1.1.2 Stakeholders

A number of stakeholders may be identified in the applications discussed above. The requirements of each group are aligned to the responsibilities and goals of that role:

- **Technical engineering and management personnel** responsible for the development and maintenance of design technology. These individuals own the detailed understanding of process and product characteristics; their buy-in is critical to the introduction of any model-based approach. They require modelling approaches to provide a detailed and convincing representation of the process.

- **Project management personnel** responsible for delivery of a project within time and budget constraints. Such stakeholders aim to achieve reductions in cost and duration of individual projects, as well as improvements in predictive and progress monitoring abilities. Since project managers are concerned with meeting immediate time and budget constraints they require pragmatic support methods which deliver short-term benefit.

- **Academic researchers** seeking to improve the effectiveness, efficiency, transparency or theoretical understanding of the design process. They require modelling approaches which are flexible to allow extension and application outside the original context, and which are implemented in robust tools which enable validation of research outputs in an industry context.

Primary application	Tool	Advantages	Disadvantages	Suitability (of 3)		
				Capture	Management	Analysis
Knowledge capture	General-purpose diagramming software (*e.g.*, PowerPoint, Visio).	Always available. Flexible.	Informal representation means no common language. All diagrams must fit on one sheet, so information is fragmented and inconsistent.	☐		
Management support	Traditional project management tools (*e.g.*, MS Project).	Low cost. Well known. Training easily accessible. Perceived as safe option.	Poor for managing dynamic and iterative processes. No simulation capability or reasoning about risk. No explicit definition of information - hence complex networks can be difficult to interpret and validate.		☐	
	General-purpose simulation-based planning tools (*e.g.*, PERTMaster, Risky Project).	Limited representation of uncertainty. Planning-specific analyses are possible, *e.g.*, critical path and resource levelling.	Underlying description is often unsuitable for design processes. The simulations cannot account for iterative or adaptive process behaviour, hence are not widely accepted in this domain.		☐ ☐	☐
	Specific design project management tools (*e.g.*, PlanWeaver, ProModeller, Plexus Modeller).	Targeted at design process planning. Some account of iteration. More suitable than general planning tools for this domain.	High purchasing cost - often including consultancy fees. Customised to specific industries (*e.g.*, construction), less successful outside these areas. Modelling frameworks are customised for planning and do not support knowledge capture *etc.*, hence it may be difficult to justify the modelling investment. Technologies are still unproven and not yet accepted into general practice.		☐ ☐ ☐	☐ ☐
Process analysis	Business process modelling and workflow tools (*e.g.*, ARIS).	Implement established methodologies for business process modelling and re-engineering.	Description/simulation is unsuitable for design. Ability to manipulate, visualise and query large models may be limited. Logical constraints and consistency checking are not enforced or not appropriate to design.	☐ ☐		☐ ☐

Figure 1.3 A summary of the advantages and disadvantages of modelling tools which have been employed to support process improvement in industry.

1.1.3 Research motivation

Figure 1.3 summarises the advantages and disadvantages of some existing tools employed to support knowledge capture, process management and process analysis in industry. This brief analysis illustrates that existing modelling approaches do not address all the applications considered above. The motivation for a more integrated approach is threefold:

- **Reduce the cost of modelling** by allowing adaptation of models to new purposes. In current practice, process improvement activities are usually undertaken by different personnel using separate representations. Enhancing existing descriptions of the design process for simulation would reduce the cost of such analysis and help ensure the knowledge of domain experts

was incorporated. Similarly, a computable model used for management support would require little enhancement prior to additional analysis.

- **Support communication between stakeholders** using a modelling technology which provides a common conceptualisation of processes and the terminology for describing them. This would support development and maintenance of process modelling expertise within a company. A common language would also enhance the dialogue between industry and academia.
- **Support development of novel approaches.** The process of developing, evaluating and deploying any model-based approach requires a great deal of implementation effort. However, in many cases such approaches are derived from existing modelling concepts. An extensible software platform suitable for a range of modelling applications and stakeholders could reduce implementation costs and thereby support academic research.

1.2 Research questions and success criterion

To recap, this dissertation will argue that:

> The processes by which complex products are developed may be modelled using a formal approach. Such models may be used to support a range of process improvement activities.

Based on the previous section this argument may be decomposed into the following research questions:

- What attributes are necessary in a design process modelling framework used for knowledge capture, management support and process analysis?
- What are the requirements for design process modelling and simulation software to support academic research and industry practice?

This research project was intended to contribute to both academic knowledge and current practice regarding design process improvement. The success criterion was therefore that results should contribute to other academic research projects and should be applied in industry.

1.3 Research methodology

In overview, the research questions were addressed by developing a process modelling framework and software tool. The answers were validated by applying the new approaches to support each application discussed above.

This section discusses the research methodology under four headings: research paradigm; research phases; research methods; and research project.

1.3.1 Research paradigm

It is important to define whether the research is considered scientific (quantitative) or qualitative in nature, since these traditions place different demands on methodological rigour. In scientific research, rigour is often *"reflected in narrowness, conciseness and objectivity, and leads to rigid adherence to research designs"*. In qualitative research, by contrast, *"rigour is associated with openness, scrupulous adherence to a philosophical perspective, thoroughness in collecting data, and consideration of all the data in the development of a theory"* (Cope, 2002).

Furthermore, whereas scientific research emphasises objectivity and repeatability, in qualitative research the researcher becomes the primary instrument of investigation. The process by which research is conducted should be carefully selected to minimise the effect of the researcher's bias on their conclusions. A thorough description of the process is usually considered necessary to aid interpretation and critique of the work (*e.g.*, Crilly 2005).

This research project incorporates elements of both scientific and qualitative paradigms. This was necessitated by the different forms of knowledge sought in each research phase in order to address the research questions. Combining these paradigms is a challenge common to much design research, which aims to both *describe* and *improve* the effectiveness of the design process. (Eckert *et al.*, 2004a).

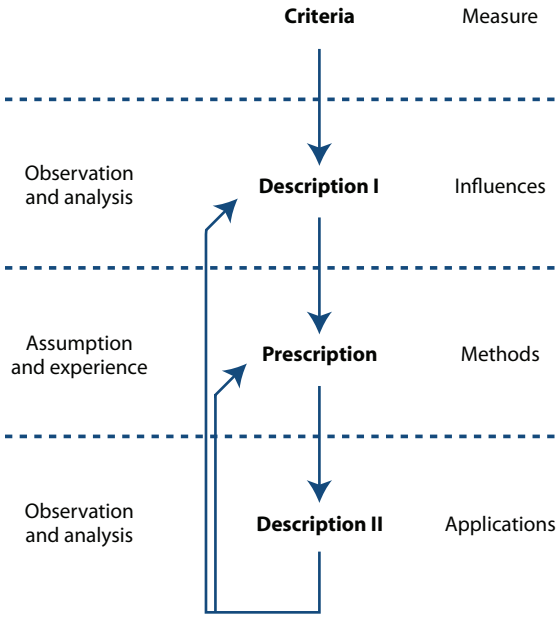


Figure 1.4 A methodology for conducting design research (Blessing *et al.*, 1995).

1.3.2 Research phases

The design research methodology proposed by Blessing *et al.* (1995) was selected as the structure for this research. Blessing's approach reflects the received scientific view of knowledge acquisition and consists of four main phases (Figure 1.4):

- **Criteria,** in which the goals of the research are established and criteria formulated by which the results will be evaluated.
- **Description I,** in which a detailed understanding of the problem is formulated through literature review and exploratory case studies. By generalising from concrete examples of the engineering design process, the phase should result in an expanded problem definition and a description of the context for which the research results may be considered valid.
- **Prescription,** in which theories describing the nature of the problem, methods which distill the theories into pragmatic approaches or descriptions of best practice, and computer tools to support application and validation are developed.
- **Description II,** in which the approach and tools are validated against the criteria through additional case studies.

1.3.3 Research methods

This section discusses the methods applied during each phase of the research. Key methodological considerations will be revisited throughout the dissertation.

1.3.3.1 Criteria: Literature review

The research questions and success criterion of Section 1.2 were developed at the outset of this project and refined as literature was reviewed (Chapter 2).

Although Blessing's approach indicates that research results should be evaluated against pre-defined criteria, it is usually acknowledged that the ideal evaluation is not possible due to time, repeatability and access constraints (Blessing *et al.*, 1995). Indeed, many philosophers of science reject the existence of testing criteria altogether (Reich, 1995). In this research, detailed testing criteria proved difficult to define due to the breadth of potential applications. The criterion discussed above was therefore viewed as the guiding principle for a primarily discursive evaluation.

1.3.3.2 Description I: Empirical study

The design process should be viewed not only as a system to be modelled and improved, but also as the context into which support solutions must be deployed. An eight-month on-site case study was therefore undertaken at Rolls-Royce to ensure the research output was sensitive to contextual requirements.

An empirical and theoretical understanding of design process modelling was developed from this study (Chapter 3). A model-based approach to support project management was proposed, thereby providing context for the main research contribution (Chapter 4).

1.3.3.3 Prescription: Modelling framework and software development

A task-based, parameter-driven process modelling framework termed the *Applied Signposting Model (ASM)* was developed to address the first research question (Chapter 5). The framework was implemented in an extensible software platform called *P3 Signposting* (Chapter 6). This addressed the second research question and enabled validation of the research via application in industry.

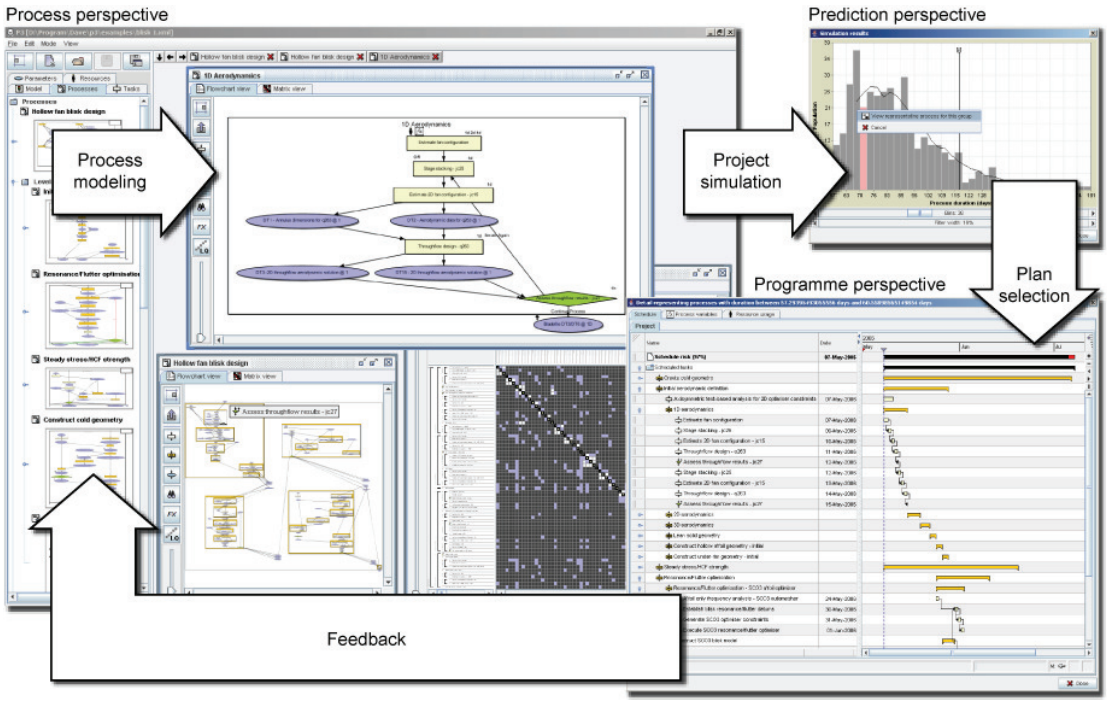


Figure 1.5 The *P3 Signposting* software for design process modelling and simulation was developed as part of the research project (Wynn *et al.*, 2006a).

From the earliest stages of the research the modelling framework and software implementation were developed concurrently through a process of iterative refinement. Following the case study discussed above, refinement of the approach was guided by frequent interaction with users in industry and academia. Each user provided ongoing feedback regarding both the modelling framework and the software. In addition to regular contact by telephone, email and in person, users were encouraged to log feature requests and bugs using a web-based issue tracking system. This regular interaction ensured the appropriateness and stability of the software implementation.

1.3.3.4 Description II: Introduction into practice

The modelling framework and software tool were validated by application to three industry-related projects and two Ph.D. projects which extended the approach as part of novel research (Chapter 7). This also illustrates their use in each of the three applications discussed above.

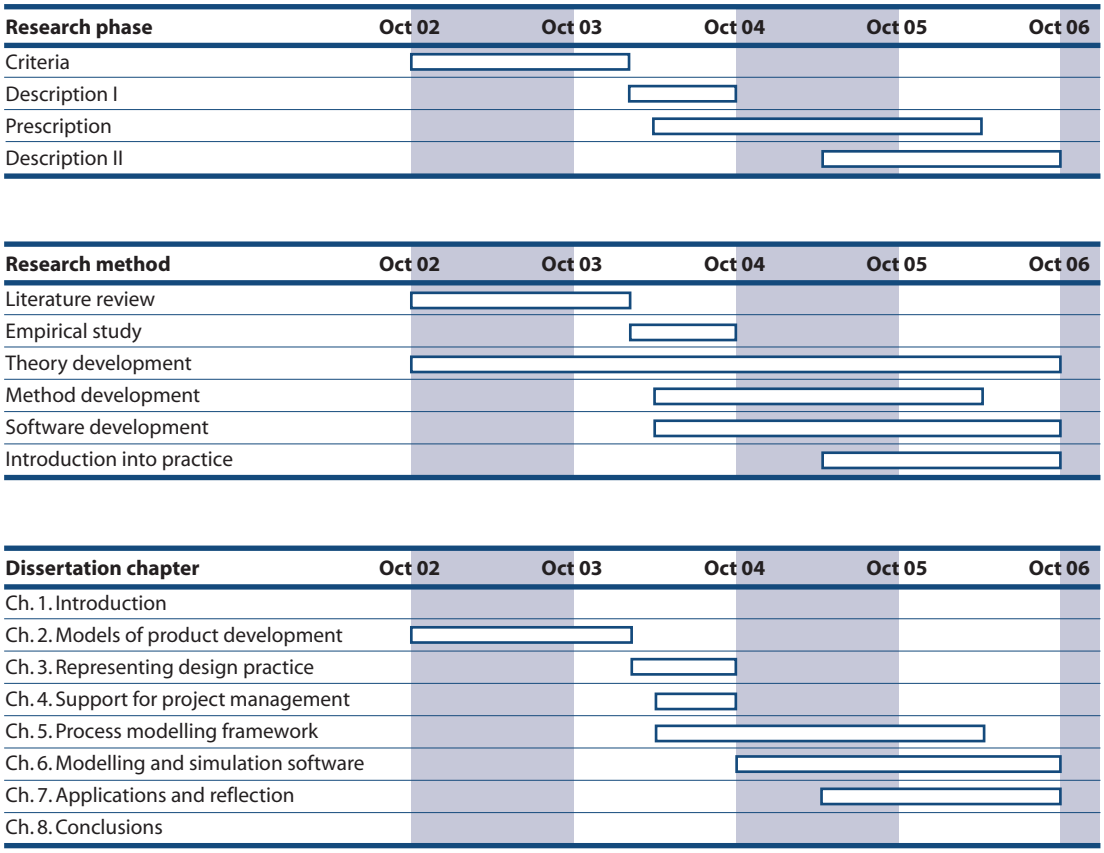


Figure 1.6 The research phases, methods and corresponding dissertation chapters.

In each case, a different user applied the approach over several months to support a process improvement project in collaboration with Rolls-Royce. It has thus been possible to evaluate the approach within its intended context of application. In addition to many informal discussions, a number of users were interviewed to explore their experience with the approach. This was combined with a more objective evaluation in which usage of the software was automatically recorded over several months' modelling activity. The resulting data detail over 100 hours' continuous usage. These data were analysed to indicate the time expended in modelling and highlight the rework-intensive nature of the modelling activity.

1.3.4 Research project

The four research phases overlapped significantly during this project; a common occurrence when following Blessing's methodology (Eckert *et al.*, 2004). Figure 1.6 illustrates the timing of each phase together with the methods applied and corresponding dissertation chapters.

1.4 Thesis summary

This dissertation proceeds in eight chapters:

1. **Introduction.** The thesis is introduced, research questions and success criteria are defined and the research methodology is discussed.

2. **Models of product development.** Published models of the design and development process are reviewed and it is argued that no single modelling approach provides an adequate description of practice. This highlights the need for additional research to identify the most appropriate basis for addressing the research questions.

3. **Representing design practice.** Drawing on an extended case study, design iteration is identified as a key influence on process behaviour and on the uncertainty surrounding project planning and management. It is argued that the behaviour of iteration is influenced by context-specific factors and shown that each of the modelling frameworks reviewed in Chapter 2 is better suited to representing certain aspects of iteration. The chapter concludes that the most appropriate modelling framework should thus be selected by considering the characteristics of iteration in the process to be modelled, as well as any specific requirements of the modelling application.

4. **Support for project management.** A model-based approach to support project management was developed to address requirements identified in Chapter 3 and thereby to provide context for addressing the research questions. The approach uses process modelling and simulation to improve the fidelity of plans and reduce the cost and frequency of re-planning.

5. **Process modelling framework.** The *Applied Signposting Model (ASM)* is introduced to address the first research question. The ASM is a task-based, parameter-driven modelling framework which provides a flexible infrastructure for design process modelling and simulation. It is based on a prototype modelling approach which was developed as part of the management support method discussed in Chapter 4.

6. **Modelling and simulation software.** A software platform entitled *P3 Signposting* is introduced to address the second research question. In addition to implementing the ASM, P3 Signposting enables rapid development of new modelling frameworks. The requirement for this functionality arose from a methodological observation which emerged during the research reported in Chapter 4.

7. **Applications and reflection.** Application of the modelling framework and software tool to support process improvement projects in industry and academia is discussed, thereby validating the research contributions with respect to the success criterion discussed above.

8. **Conclusions.** The research questions are discussed in light of the dissertation. Research contributions are reviewed, opportunities for further work are highlighted and conclusions are drawn.

Chapter 2

Models of product development

This chapter sets the thesis in context by reviewing published models of the design and development process. A classification scheme is developed to encompass the wide range of models which are discussed. This reveals the diverse range of models and modelling approaches, none of which is agreed to adequately reflect all aspects of product development. It is therefore concluded that empirical research is necessary to identify the most appropriate basis to support process improvement in a given context.

The body of literature relevant to this review is expansive and incorporates a broad range of perspectives. Browning and Ramasesh recently published a comprehensive study of task network approaches to modelling product development processes, identifying around 400 publications relevant to this sub-set alone (Browning and Ramasesh, 2007). It is not feasible to exhaustively review the literature which influenced the research project; therefore, this chapter provides an overview of design process models prior to focusing on those which have greatest relevance to this dissertation.

2.1 Overview

Many authors have proposed theories, models and methods to explain or improve upon aspects of design practice. This body of literature is known as *design methodology* and is concerned with:

> *"...the study of how designers work and think; the establishment of appropriate structures for the design process; the development and application of new design methods, techniques and procedures; and reflection on the nature and extent of design knowledge and its application to design problems."* (Cross, 1984).

Despite extensive research undertaken in this area since the 1950s, no single theory or model is agreed to provide a satisfactory description of the design process (Bahrami and Dagli, 1993). Likewise, there is no 'silver bullet' method which can be universally applied to achieve process improvement. Most methods have a well-defined and often relatively narrow focus, ranging from the generation of mechanism concepts (*e.g.*, Pahl and Beitz 1995) to the management of project risk (*e.g.*, Baxter 1995).

In this chapter, some popular approaches to modelling the design process are presented and their practical relevance is discussed. Throughout the chapter, a classification framework is developed to support the discussion and to relate the diverse range of forms exhibited by these models.

2.2 Classifying models of designing

A theme of this thesis is to highlight the difficulty of developing a satisfactory model of the design and development process. It is an equally challenging task to describe the relationships between models concerned with various aspects of product development — although authors such as Bayazit (2004), Cross and Roozenburg (1992) and Blessing (1994) have used a variety of approaches to frame discussions of such literature, these frameworks can seem as diverse and difficult to relate as the models they describe.

Commonly cited factors which differentiate the literature include the discipline of origin, in which architectural literature has emphasised the creativity at the heart of design activities, whereas engineering tradition has taken a more rationalist, scientific approach as exemplified by authors such as Hubka and Eder (1980) and Pahl and Beitz (1995). Bayazit discusses design methodology from the perspectives of nationality of origin and the historical development of form (Bayazit, 2004).

The following sections briefly discuss three perspectives which can be useful in highlighting issues of practical relevance and which clarify the positioning of the work presented in this thesis. These inter-related perspectives are: *stage vs. activity-based models* of the design and development process; *problem vs. solution-oriented strategies* for solving design problems; and *abstract vs. analytical vs. procedural approaches* to modelling and/or improving the design process. Preceding a discussion of the models themselves, these three perspectives will be introduced in the following sections.

2.2.1 Stage- vs. activity-based models

Blessing (1994) classifies models of designing using the four categories shown in Figure 2.1. This framework is based on the earlier theorising of Hall (1962), who proposed a two-dimensional perspective of development projects in which the stage-based structure of the project life-cycle lies orthogonal to an iterative problem-solving process which takes place within each stage. Asimow (1962) further developed this theory, transferring Hall's ideas from the domain of systems engineering to that of design. Asimow described the essentially sequential, chronological structure of the project as the *morphological dimension* of the design process, and the highly cyclical, rework-intensive activities characteristic of the designer's day-to-day activities as the *problem-solving dimension*.

Blessing refers to those models concerned with Asimow's morphological and problem-solving dimensions as *stage-based* and *activity-based* respectively (Figure 2.1). She also notes the existence of combined models which prescribe well-structured, iterative activities within each stage (*e.g.*, Hubka, 1982); by comparison, purely stage-based models indicate only the possibility of rework using

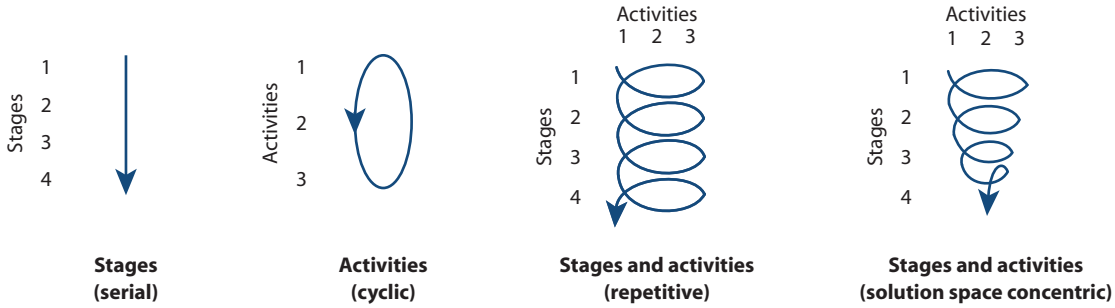


Figure 2.1 Stage- *vs.* activity-based models (Blessing, 1994).

feedback loops between stages (*e.g.*, French, 1999). Some combined models illustrate convergence on a design solution by proposing progressively more concrete activities in each stage (*e.g.*, Evans, 1959). It will be seen that models with a stage-based component can be easier to interpret and apply than their purely activity-based counterparts.

2.2.2 Solution- vs. problem-oriented strategies

Another commonly used scheme places literature into either of the following two categories, according to the strategy the author proposes is used to reach the design goal (*e.g.*, Lawson, 1980; Birmingham *et al.*, 1997):

- **Problem-oriented** in which emphasis is placed upon abstraction and thorough analysis of the problem structure before generating a range of possible solutions;

- **Solution-oriented** in which an initial solution is proposed, analysed and repeatedly modified as the design space and requirements are explored together.

Observing graduate students of architecture and science asked to solve a simple design problem, Lawson (1980) concluded that the strategy chosen in practice is determined by training and background; designers appeared to prefer the more creative 'try it and see' solution-oriented approach, while the scientifically trained focused on unravelling the problem before attempting to synthesise solutions. Lawson went on to describe the interlinked and subjective nature of problem specifications and design solutions, a persuasive argument supported by many other authors (*e.g.*, Jones, 1970; Cross, 1994), and concluded that real design

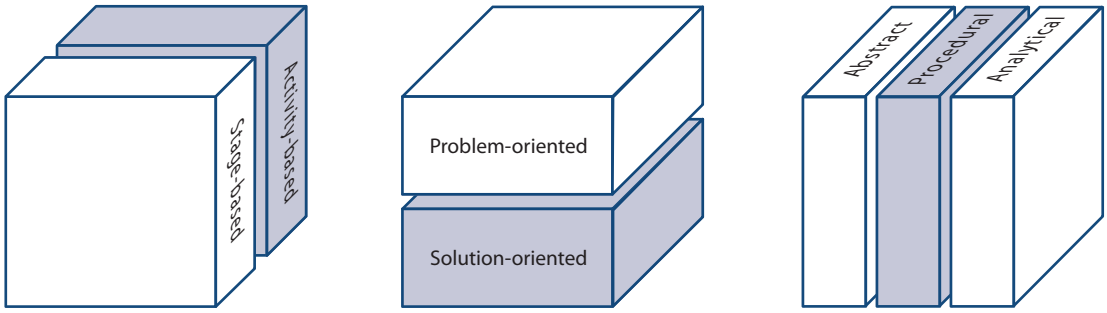


Figure 2.2 A framework for organising models of the design and development process (Wynn and Clarkson, 2005).

problems cannot be solved in a purely problem-oriented fashion. In fact, it is generally recognised that designing requires application of both of these strategies at one point or another, according to the individual nature of each problem the designer encounters (Frost, 1992). To summarise, it may be seen that stage-based models typically adopt a problem-oriented strategy, whereas activity-based models may be either problem or solution-oriented in nature.

2.2.3 Abstract vs. procedural vs. analytical approaches

This dissertation focuses on model-based approaches which are relevant and readily applicable to support process improvement. With this in mind, a third set of categories is proposed here to position the research against existing literature (Figure 2.2):

- **Abstract approaches** are proposed to describe the design process at a high level of abstraction. Such literature is relevant to a broad range of situations but does not offer specific guidance useful for process improvement.

- **Procedural approaches** are more concrete in nature and focus on a specific aspect of the design process. They are less general than abstract approaches, but can be more relevant to practical situations.

- **Analytical approaches** are used to describe particular instances of design processes. Such approaches consist of two parts: a modelling framework used to describe aspects of a process; and techniques, procedures or computer tools which make use of the representation to support investigation or improvements to that process.

To relate this typology to the previous discussion, abstract models are usually activity-based in nature — although this relationship is not clear in many cases — and thus may adopt either a problem- or solution-oriented strategy. Procedural models are problem-oriented in nature and always include a stage-based component. Analytical models may fall into any category in the previous schemes according to the purpose for which the model is constructed.

The framework depicted in Figure 2.2 is further developed in Sections 2.3, 2.4 and 2.5 through more detailed discussion of abstract, procedural and analytical approaches respectively.

2.3 Abstract approaches

A commonly held theory of designing is that designers can and should resist bringing their own preconceptions to bear on a problem. Models proposing this problem-oriented strategy are essentially sequential in nature. This is typified by Jones' model (1963), in which the design process comprises the three stages of *analysis*, *synthesis* and *evaluation*. The analysis stage involves consideration of the problem and its structuring into a set of objectives. Synthesis involves generation of a range of solutions, and evaluation requires the critical appraisal of the solutions against the objectives.

Cross (1994) proposed a four-stage variant in which the designer first *explores* the ill-defined problem space before *generating* a concept solution. This is then *evaluated* against the goals, constraints and criteria of the design brief. The final step is to *communicate* the design specification either for manufacture or integration into a product or subsystem. Since generation does not always result in a satisfactory solution, Cross includes a dependency from the evaluation to the generation stage (Figure 2.3).

These problem-oriented models are based on the premise that designers are capable of formulating a solution-neutral problem statement, and propose that the final design should be more dependent upon logical reasoning than prior experience. This assumption, common to all problem-oriented literature, forms the basis of the procedural design models introduced in Section 2.4.

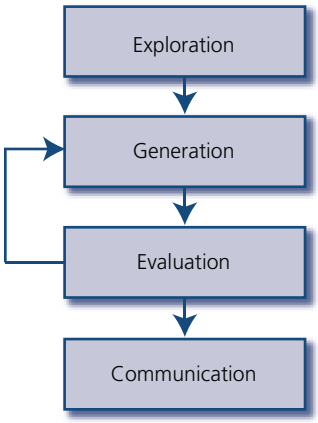


Figure 2.3 Cross' model of the design process (Cross, 1994).

In opposition to this problem-oriented perspective, Hillier *et al.* (1972) proposed the *conjecture-analysis* theory to reflect the belief that a designer would pre-structure a problem in order to solve it; that is, that existing knowledge and previous experiences form an unavoidable influence on the nature of the solution. This concept forms the basis of the solution-oriented models of design, which are commonly considered to provide more realistic descriptions of the designer's thought process than their problem-oriented counterparts.

One example of a solution-oriented model was proposed by Darke (1979) following observations of architectural design practice. Darke argues that the designer does not start by studying an explicit list of problem factors and objectives to be met by the design, but rather tries to reduce the set of possible solutions to a smaller class which is more manageable. To achieve this, a subset of the objectives is chosen, based on prior experience of similar problems and subjective judgement. Darke terms this subset the *primary generator*, consideration of which leads to a possible solution or conjecture being produced. This enables further clarification of the design requirements, against which the solution is tested and further improvements are made. This model implies a serial, depth-first process of designing rather than the parallel, breadth-first search of the problem space described by problem-oriented models.

March (1984) proposed a solution-oriented model of reasoning in design termed the *production-deduction-induction* model. Drawing on the philosophy of Peirce (1923), March argued that the two conventionally understood forms of reasoning, *i.e.*, *deduction* and *induction*, can only describe the evaluative and analytical

aspects of design respectively. He proposed that Peirce's third type of reasoning, termed *abductive* (or *productive*), is responsible for the essential creative activities. From this he developed the triple activity model described below.

In the first phase of *productive reasoning* the designer draws on the vague problem statement and his or her existing knowledge to conceive a candidate solution. In the second phase, *deduction* is used to analyse or predict the system behaviour based on an understanding of key physical principles. In the third phase, *inductive reasoning* is used to identify possible means of improving performance by altering the design. In common with other solution-oriented models, the iterative nature of designing is given primary emphasis.

Protocol studies have been used to support these descriptive theories of design thinking (*e.g.*, Cross *et al.* 1996). Other authors have questioned the validity of these results, describing how continuous verbalisation can affect the mental imagery which is considered central to many design activities (Eckert, 1997).

2.3.1 Discussion

The theories and models discussed above provide high-level, generic descriptions of design practice. As such, abstract approaches do not explain the process of designing in detail. They are characterised by a small number of stages or activities and do not describe the specific steps or techniques which might be used to reach a solution. The practical applicability of such approaches is described rather colourfully by Lawson as:

> *"About as much help in navigating a designer through his task as a diagram showing how to walk would be to a one year old child"...*
>
> *"Knowing that design consists of analysis, synthesis and evaluation will no more enable you to design than knowing the movements of breaststroke will prevent you from sinking in a swimming pool."* (Lawson, 1980)

Nevertheless, these abstract models can provide useful insight since they concern the iteration which is ubiquitous in design. This will be revisited in Chapter 3.

2.4 Procedural approaches

Procedural approaches are more concrete in nature than the abstract theories and models discussed above, typically incorporating a larger number of phases and focusing on a specific audience and/or industry sector. Such literature is commonly categorised as follows (Finger and Dixon, 1989):

- **Descriptive approaches** result from investigation into actual design practice. Processes and procedures observed in industry form the basis of texts which are used primarily for teaching, training and research purposes. The abstract theories and models introduced in the previous section are descriptive in nature.
- **Prescriptive approaches** are distillations of best practice intended to improve effectiveness or efficiency in some aspect of the design project. Such procedures are usually targeted toward a particular audience (for example, student, design engineer or manager) and domain (for example, industrial or mechanical design).

Prescriptive approaches recommend or prescribe guidelines, stages or techniques which, if implemented correctly, are thought to improve performance in specific aspects of the product or project. To illustrate, a procedure may be intended to improve product reliability, or to improve visibility of the design process to its participants. Hubka (1982) expresses a commonly held view by recommending such procedures when searching for solution concepts in order to cover a wider search space, and also suggests that following a systematic approach can be particularly beneficial in all review and revision activities. Archer (1965) proposes that systematic approaches are particularly useful under one or more of three conditions: when the consequences of being wrong are grave; when the probability of being wrong is high (for example, due to lack of prior experience); and/or when the problem is complex, characterised by many interacting variables.

Aspects of both descriptive and prescriptive approaches may be found in most literature. To illustrate, consider the following distinction of scope in procedural approaches:

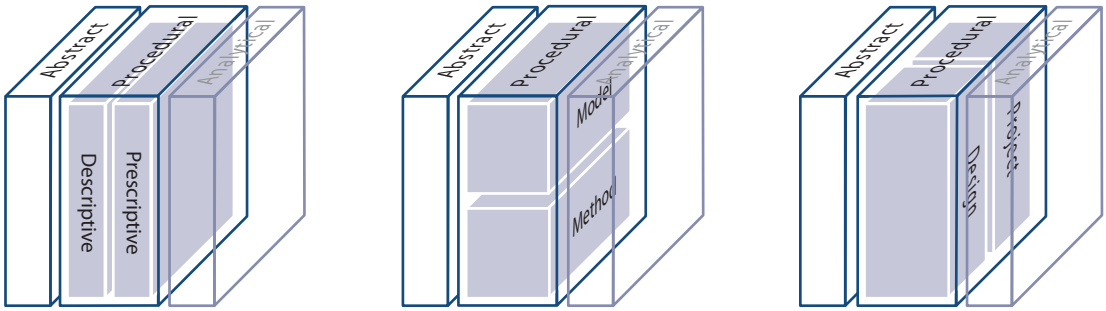


Figure 2.4 A framework for organising procedural models of the design and development process (Wynn and Clarkson, 2005).

- **Models** refer to a description or prescription of the morphological form of the design process.
- **Methods** prescribe systematic procedures to support the stages within a model.

Much procedural literature combines collections of prescriptive methods, such as brain-storming, synectics, functional analysis and morphological combination, with a problem-oriented model to illustrate the context of each method. Furthermore, models and methods are often intertwined, with the stages of each model being dependent upon the methods from which it is composed. The schemes of prescriptive *vs.* descriptive and model *vs.* method will not be discussed further. Instead, the discussion will consider approaches according to their *focus*, which falls between the following two extremes:

- **Design-focused approaches** support the generation of better products by application of prescriptive models and methods to the design process (*e.g.*, Pahl and Beitz, 1995).
- **Project-focused approaches** advocate methods to support or improve management of the design project, project portfolio or company (*e.g.*, Hales, 2004).

A summary of the framework for organising procedural approaches is depicted in Figure 2.4, which illustrates the dimensions of *descriptive vs. prescriptive*, *model vs. method* and *design-focused vs. project-focused*. The following sections introduce some well-known procedural approaches, beginning with models focused on the process of design.

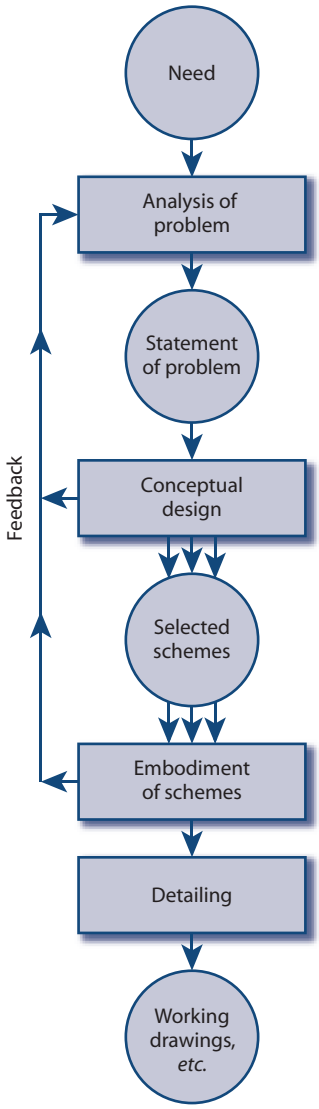


Figure 2.5 French's model of the design process (French, 1999).

2.4.1 Design-focused models

Many procedural models present design as a series of stages, each of which is visited only once by the ideal process. A typical example proposed by French is based on design practice observed in industry (Figure 2.5). The model consists of four stages (French, 1999):

- The process begins with the observation of a *market need* which is *analysed* leading to an unambiguous *problem statement*. This takes the form of a list of requirements which the product must fulfill.

- During the *conceptual design* stage several *concepts* are generated, each representing a set of physical principles for solving the problem. These schemes are transformed into a more concrete representation to allow assessment and comparison. The resulting concepts are evaluated and one or more are chosen to form the basis of the final solution.

- The chosen architecture is then solidified in the *embodiment phase*, where the abstract concept is transformed into a *definitive layout*.

- Finally, the remaining *details* are added to remove all ambiguity from the solution, leading to release of *manufacturing instructions*.

French accounts for the disordered activity sequences observed in practice by describing his model as hierarchical in nature; in other words, a project may encompass several stages of the model according to the completeness of each aspect or module of the design.

Perhaps the most well known of the stage-based models was proposed by Pahl and Beitz (1995) for mechanical design (Figure 2.6). Each of the four prescribed phases consists of a list of *working steps* which they consider to be the most useful strategic guidelines for design. They propose that following these steps ensures nothing essential is overlooked, leading to more accurate scheduling and resulting in design solutions which may be more easily reused.

Although many more design-focused models may be found in the literature, Cross and Roozenburg (1992) describe how most have converged upon the general *consensus form* exhibited by the models of French and of Pahl and Beitz. Other examples may be found in the work of Dym and Little (2000), Ullman (2003), Pugh (1991) and Roozenburg and Eekels (1995). The consensus model characterises iteration as an undesirable or failure-driven phenomenon. Iteration is usually indicated by incorporating 'feedback' between stages, which also indicates the transfer of insights between projects. A different perspective was offered by Evans (1959), who proposed a combined stage and activity model which explores the iterative nature of the design process. Noting that one of the most fundamental problems of design lies in making trade-offs between many interdependent factors and variables, Evans argues that design cannot be achieved by following

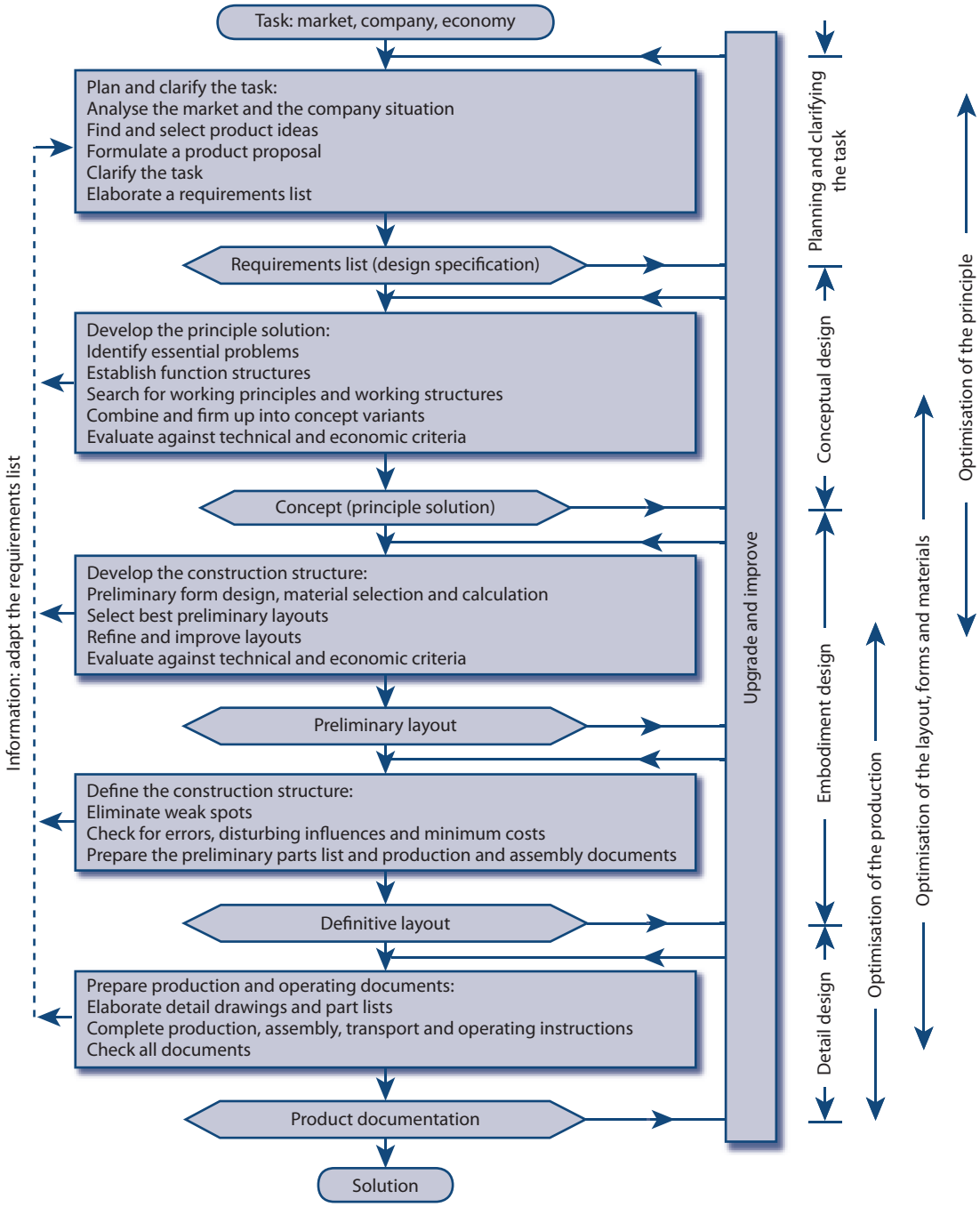


Figure 2.6 Pahl and Beitz' model of the mechanical design process (Pahl and Beitz, 1995).

a sequential process. He demonstrates this using the example of bridge design, where the structure must be chosen to support the dead weight of the material, but the weight is not known until the structure has been defined. According to Evans such interdependencies are characteristic of design problems, a view which has become ubiquitous in modern thinking about design. Evans proposes that an iterative procedure is adopted to resolve such problems; early estimates are

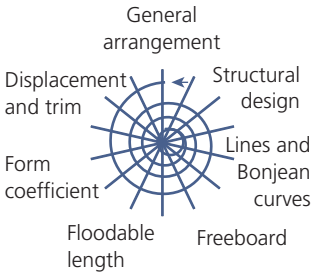


Figure 2.7 A summary of Evans' model of the ship design process (Evans, 1959).

made and repeatedly refined as the design progresses, until such time as the mutually dependent variables are in accord. Based upon this principle he proposed the prescriptive model for ship design shown in Figure 2.7.

The radial lines show the aspects of the ship — the interdependent variables — which must be chosen for the design to be complete. As the project progresses, these variables are gradually refined by repeated attention until the ultimate, balanced solution is reached. At each iteration the manoeuvring room (*i.e.*, margins set aside for change) decreases as the interdependencies are gradually resolved, smaller modifications are required, and different methods may be applied to each problem. Evans notes that the effort required to improve the design increases as the solution converges, and that more and more resources may be applied as the project moves towards completion.

2.4.1.1 Design-focused methods

The methods accompanying procedural models are intended for use by engineers and designers to support the execution of individual stages or design steps. They typically concentrate on the early stages of the design process — Roozenburg and Cross (1991) question the existence of detailed procedures for the embodiment and detail design stages. As with models, methods may be focused or independent of discipline; for example, morphological combination is of limited use in the design of non-mechanical products such as microprocessors, whereas brainstorming and requirements analysis are applicable to many situations. According to Pugh (1991), successful design is subject to the integration of such general design methods with traditional engineering expertise.

2.4.1.2 Discussion

The design-focused models introduced above concentrate upon the technical aspects of solving design problems, describing or prescribing the steps thought necessary to progress from problem to solution. Some authors have developed models which offer general insights that have since been applied to many design disciplines — the form of Evans' spiral model, for example, appears in diverse fields from naval architecture (Rawson and Tupper, 1994) to software engineering (Boehm, 1988) more than four decades after publication. Other approaches, such as the mechanical design models exemplified by Pahl and Beitz, incorporate a strong focus on the details of product structure. These approaches have found fewer applications outside their original disciplines.

In practice, the applicability of many such models and their accompanying methods is limited by their product-focused perspective. This implies that the key difficulty in a design project lies in finding solutions to the technical problems. In reality, however, even the simplest design process is a highly complex socio-technical activity requiring a much broader range of skills, from marketing to human resource management. Furthermore, many authors describe how most complex design projects place strong limitations on early concept design, with constraints such as existing product platforms and legislative requirements often predetermining the form of the solution (Pugh, 1991). In such circumstances the primary difficulty design companies face lies in the integration of diverse methods, disciplines, tools and personnel (Andreasen and Hein, 1987). The design-focused approaches discussed above do not highlight these issues.

2.4.2 Project-focused models

The project-focused literature discussed in following sections emphasises the context of the design process, including cost-related activities such as product planning and risk management (*e.g.*, Baxter 1995). In other words, project-focused literature concentrates less on product design and more on product development, defined by Roozenberg and Eekels (1995) as the development of a new business activity around a new product. Understanding the interaction between developing new products and new business is often considered key to success in this.

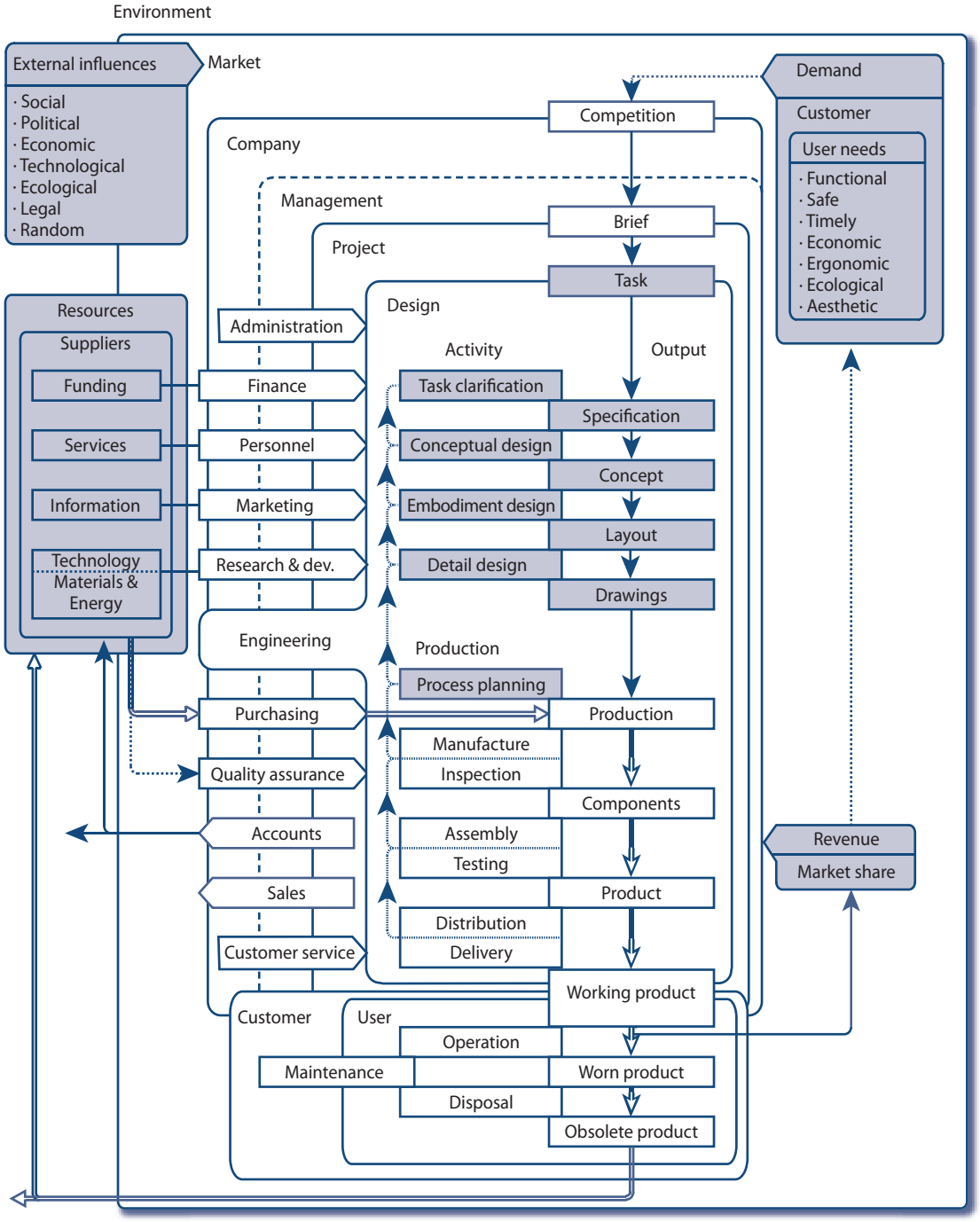


Figure 2.8 The design process in context (Hales, 2004).

2.4.2.1 Managing influences on the project

Projects are influenced by a large number of factors which have little relation to the design process. Such influences vary from project to project and ensure that each is unique. For example, individual projects must often compete for limited resources within a company. Furthermore, the development of complex

products typically requires the coordination of many organisations, of which the individual companies may have responsibilities ranging from subsystem design to component manufacture; in either situation, successful integration of inter-organisational processes is critical to prevent delays to the project. For example, specification errors can be extremely costly for an externally designed, long lead-time component. Other influencing factors are further removed from the project and cannot be directly managed or influenced. For example, changes in organisational structure, government legislation or available manufacturing technology may cause a project to fail or to be cancelled.

Management of these issues poses a challenge which Hales proposes is best resolved by promoting awareness of the influencing factors and their possible impacts on the project (Hales, 2004). He provides a comprehensive list of such influences at several different levels, including the macro-economic, micro-economic and corporate scales, summarised in diagram form in Figure 2.8. His method advocates the explicit consideration of each item on these checklists so that design managers can gain a broader perspective and make more informed decisions. The figure places the familiar stage-based view of the design process into its context within the project, company and market.

2.4.2.2 Integration of personnel and disciplines

Focusing on concept development, Ulrich and Eppinger (2003) include the following as key challenges in new product development:

- Recognising, understanding and managing product related trade-offs, such as weight *vs.* manufacturing cost.
- Working in an environment of constant change. As technologies and customer demands evolve and competitors introduce new products, there is a constant time pressure on all design and development activities.
- Understanding the economics of product development from marketing through to manufacture and sales, so that a return can be made on initial investments.

Ulrich and Eppinger propose methods intended to meet these challenges, facilitating problem solving and decision-making by integrating personnel from a variety of backgrounds and perspectives. Taking a similar viewpoint, Pugh writes that industry is primarily concerned with *total design*, *i.e.*:

> *"The systematic activity necessary, from the identification of the market/user need, to the selling of the successful product to satisfy that need — an activity that encompasses product, process, people and organisation."* (Pugh, 1991)

In other words, development of any product requires collaboration of personnel familiar with many different disciplines, including those of technical engineering, engineering design, and many other non-technical fields. This integration of disciplines requires that all participants have a common view of the total design activity and can thereby subscribe to a common objective with a minimum of misconceptions. Pugh believes that visibility of operational structure is key to this common understanding, so that *"everyone can find out what people are doing and why"* (Pugh, 1991). He proposes that a disciplined and structured approach is necessary to achieve this.

2.4.2.3 Process control and evaluation

The problem-oriented perspective of the design process as an ordered progression through a series of stages is popular in industry and has been adopted in a variety of forms by many successful companies, including DuPont, 3M, Hewlett-Packard, Procter and Gamble, ICI-UK, IBM, Polaroid, Black and Decker and Exxon Chemicals (Cooper, 1994). The *gates* between stages, through which each project must pass to continue, are a dual-purpose structure used both for rationalising decisions and for planning. The well-defined deliverables from each stage are convenient documents with which to assess whether a project is likely to succeed, and the timing of these milestones anchors the schedule of the overall development project. The artificial division of the process provides management with a quality control structure in which each gate represents an opportunity to recognise and halt a failing project; if the criteria for passing each gate are chosen wisely, following a prescribed process is one way of assuring the quality of the

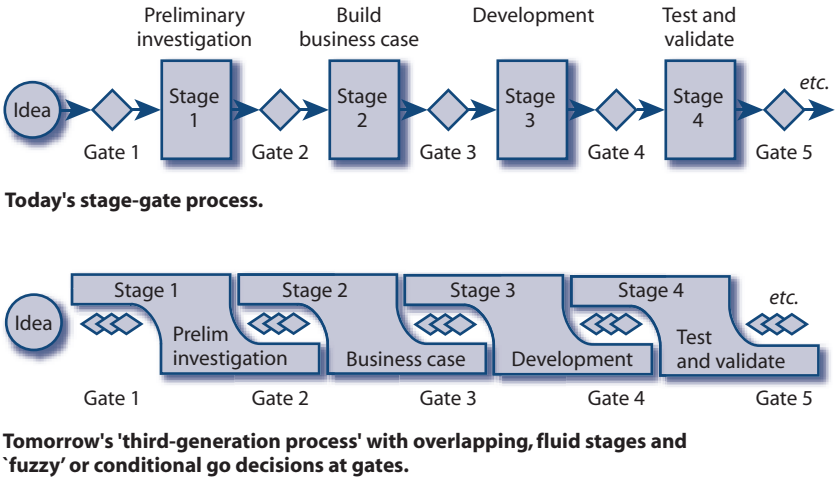


Figure 2.9 Stage-gate structuring (Cooper, 1994).

resulting product (Ulrich and Eppinger, 2003). Implementing such procedures allows a company to comply with quality standards such as ISO9000 (1994) or APQP (1995). This is obligatory for large engineering firms, as most European companies require their suppliers to gain such accreditation.

However, Cooper (1994) argues that there are many practical weaknesses to this form of gated process control. The system can be inefficient, since projects must wait at a gate until all necessary activities have been completed. The overlapping of stages is not permitted in most cases, although it is often desirable in the above situation. There can be high bureaucratic overheads at each gate, and the individual project perspective means there is little provision for managing the division of resources across a portfolio. Cooper proposes that these systems should be made more fluid and adaptable, should incorporate 'fuzzy gates' which are situational and conditional, should provide for sharper focus of resources and better management of the portfolio of products under development, and should be generally more flexible than the current stage-gate model (Figure 2.9).

2.4.2.4 Discussion

Many authors have proposed models and methods to support design project management. However, for each situation which may be improved by the application of such methods, there are many more which cannot; those problems highly dependent on human factors may be particularly resistant. Another common difficulty lies in the balancing of activities and resources across a portfolio of projects

Company structure	Manufacturing process	Suppliers
Establishment size Independence Centralisation	Process complexity Process flexibility Process constraint Production volume Internal span of process	Rationalisation Degree of control Collaboration Locality

Market/customer	Product	Local environment
Market type Market size and share Market complexity Market infrastructure Exports Number of competitors Competitive criteria	Product type Product variety/range Product complexity Product structure Product status Innovation rate Design capability	Local labour market Skills Training Local support infrastructure Financing and grants

Figure 2.10 Company classification dimensions and key factors (Maffin *et al.*, 1995).

under development; many methods are strongly focused on individual projects and offer little useful guidance in such a situation.

These models give a view of the design process as a series of stages and provide good visibility of the design goals. Although successful design practice often follows these patterns, the high level of abstraction does not provide much guidance towards practical implementation. The models are more indicative than directive of best practice; Blessing writes that they are seldom employed as methods for design process improvement (Blessing, 1994).

The methods range from broad but abstract through to concrete but limited in scope, and each design project represents a unique combination of a wide range of factors. It is clearly important to make informed decisions about the nature of the models to adopt in the context of a particular situation. Maffin *et al.* (1995) argue that, although most process models are too general in scope and prescriptive in nature for easy application, they can be interpreted for use in each design company (Figure 2.10). They propose that a set of critical factors which define the organisation and the product are influential upon the product development process, and that classifying companies according to this framework could form the basis for guiding the application of models in industry. Within a company, and for a particular line of products, many of these factors may be considered fairly constant.

	What? DATA	How? FUNCTION	Where? NETWORK	Who? PEOPLE	When? TIME	Why? MOTIVATION
SCOPE {contextual} Planner	List of things important to the business	List of processes the business performs	Locations in which the business operates	List of organisations important to the business	List of events/cycles significant to the business	List of business goals / strategies
BUSINESS MODEL {conceptual} Owner	*e.g.,* Semantic model	*e.g.,* Business process model	*e.g.,* Business logistics system	*e.g.,* Work flow model	*e.g.,* Master schedule	*e.g.,* Business plan
SYSTEM MODEL {logical} Designer	*e.g.,* Logical data model	*e.g.,* Application architecture	*e.g.,* Distributed system architecture	*e.g.,* Human interface architecture	*e.g.,* Processing structure	*e.g.,* Business rule model
TECHNOLOGY MODEL {physical} Builder	*e.g.,* Physical data model	*e.g.,* System design	*e.g.,* Technology architecture	*e.g.,* Presentation architecture	*e.g.,* Control structure	*e.g.,* Rule design
DETAILED REPRESENTATIONS {out-of-context} Subcontractor	*e.g.,* Data definition	*e.g.,* Program	*e.g.,* Network architecture	*e.g.,* Security architecture	*e.g.,* Timing definition	*e.g.,* Rule specification

Figure 2.11 The Zachman framework for classifying models of enterprise architecture (Zachman, 1987).

2.5 Analytical approaches

The models discussed above offer insights into the nature of development projects and design processes. However, they are often too general to advise planning activities or to guide the daily decisions which are made by design managers. This section reviews analytical approaches which aim to provide this lower-level support.

To recap, analytical approaches to process improvement involve the modelling and analysis of a specific situation and aim to provide insights and advice which may be operationalised. Many modelling frameworks have been proposed as the basis of such methods. The *Zachman framework* aims to provide a comprehensive picture of the possible approaches to modelling enterprise architectures, *i.e.*, the various structures found within an organisation. The framework considers that such modelling aims to address questions of *What? How? Where? Who? When?* and *Why?* posed by the stakeholders *Planner*, *Owner*, *Designer*, *Builder* and *Subcontractor* (Figure 2.11). Zachman proposes that every modelling approach can be uniquely categorised as a combination of one question and one stakeholder, and that the alternative perspectives provided by each approach are 'additive and complementary' (Zachman, 1987).

The following sections discuss approaches which may be applied to model design and development processes. Such models may be classified as follows:

- **Task network models** represent processes as aggregations of *tasks* which may be tackled individually.[1] These models assume that tasks are selected and attempted with the goal of driving the process towards completion (Browning and Ramasesh, 2007). They often describe project stages together with the iterative, problem-solving steps within stages. With regards to the typology discussed in Section 2.2.1 above (p. 17), most task network models may thus be considered as combined stage- and activity-based in nature.

- **Queueing models** consist of a number of independent processing stations which transfer discrete packets of information.[2] Such models incorporate dynamic simulations in which each station maintains an in-tray of information waiting to be processed. When a unit of work is completed, any resulting information is transferred to the in-tray of another station for further processing (Adler *et al.*, 1995). Work queueing approaches are most appropriate for studying processes whose connectivity is well understood, but whose dynamics exhibit complex, emergent properties. Manufacturing systems and supply chains are classic examples of such processes.

- **Multi-agent models** view processes as a collaboration of decision-makers whose actions are dependent upon information received from one another. The defining characteristic of a multi-agent model is the bounded rationality of decision-makers — in other words, each agent acts without knowledge of the entire system (*e.g.*, Olsen *et al.* 2006; Mihm *et al.* 2003).

- **System dynamics models** view processes as work-processing systems whose behaviour is governed by the feedback and feedforward of information about the current process state.[3] In contrast to other approaches,

[1] Task network models are described by Browning and Ramasesh (2007) as *Activity network models*. To avoid confusion with the terminology of later chapters, the term *Activity* henceforth refers to the actual work and *task* to an element of a model which represents this work.
[2] In this thesis the term *information* may refer to both data and physical material.
[3] Note the important distinction between the feedback which governs process behaviour and *iteration*, which may be viewed as facilitating feedback since an iterative process typically

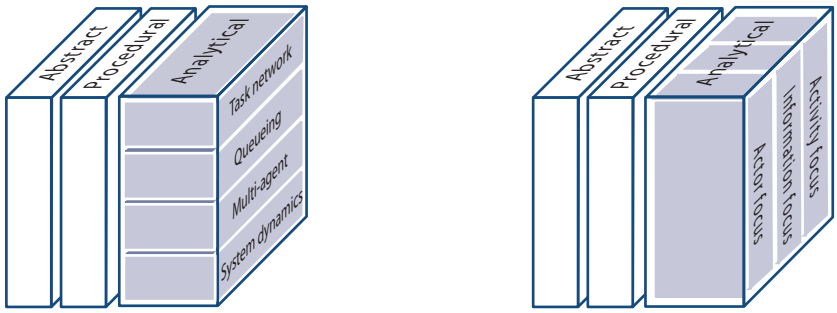


Figure 2.12 A framework for organising analytical models of the design and development process.

system dynamics models of design processes are abstract and consist of few elements. The equations governing feedback are of critical importance in determining the dynamical behaviour of the model.

Each model type provides a different perspective of process structure and behaviour. Each is thus suited to modelling systems with certain characteristics. For example, task network and queueing models emphasise that system complexity arises from the structural connectivity of many simple elements, whereas multi-agent and system dynamics models assign an important role to the behaviour of individual components. Another consideration guiding the selection of an appropriate framework is the purpose for modelling. For a modelling framework to satisfy its purpose it should have appropriate *focus*:

- **Activity-focused** frameworks emphasise individual tasks and their context within the process. Task network models are activity-focused and queueing models include an activity-focused component.
- **Information-focused** perspectives seek to represent the influential role of evolving information in processes. System dynamics, multi-agent and queueing models typically incorporate an information focus. Although task network models do represent information flows, they are treated as the basis for connectivity rather than as a primary driver of behaviour.

releases information more frequently, *e.g.*, following each attempt of an evaluation activity. Iteration is discussed further in Chapter 3.

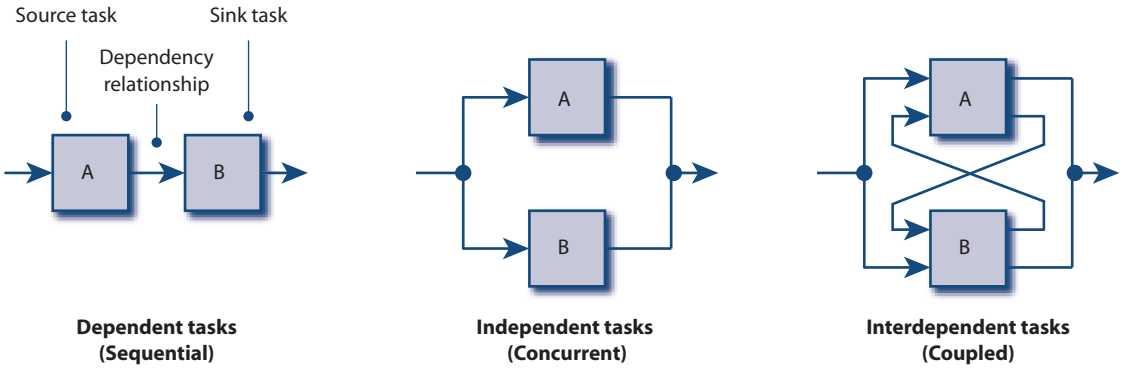


Figure 2.13 Three possible direct dependency relationships between two tasks (Eppinger, 1991).

- **Actor-focused** frameworks highlight the roles of individuals in determining process behaviour. Pulm describes such approaches by analogy to a 'beehive', exhibiting self-organised complexity in contrast to the structured 'relay race' implied by activity-focused models (Pulm, 2006). Multi-agent models are actor-focused and work-queueing models incorporate an actor focus.

Since the body of relevant literature is extensive, the remainder of this chapter focuses on those approaches which have been used to model design and/or development processes. The discussion allows the modelling framework introduced in Chapter 5 to be positioned with respect to existing literature. Task network models are most relevant to this objective and thus form the focus of the review.

2.5.1 Task network models

Task network models consider that the order in which tasks may be attempted is governed by the information required and generated by each task. For example, consider the three possible direct dependency relations between two tasks shown in Figure 2.13 (Eppinger, 1991). Capturing these relationships for all the tasks comprising a design activity allows the process to be represented as a task dependency network, through which there may be many possible routes from start to finish. The order of task execution can then have an important effect on the efficiency of a process; it is often proposed that achieving an appropriate ordering of the tasks is critical to successful product development (*e.g.*, Eppinger, 1991, Clarkson and Hamilton, 2000).

Many valid task decompositions may be developed for any design activity. Furthermore, tasks may be specified at varying levels of detail appropriate to the context in which they are executed and the purpose for which a model is constructed.

Task network models are especially appropriate in the modelling of adaptive or variant design processes, since in such scenarios the majority of tasks and parameters are well delimited and may be identified in advance (Pahl and Beitz, 1995). This is the case with the aero-engine design processes studied in this thesis, which — as will be discussed in Chapter 3 — may be described to a large extent in terms of the design and analysis tools used to perform most activities.

2.5.1.1 Classification of task network models

Browning and Ramasesh (2007) note that all process models are developed to meet an objective and structure their review of task network approaches accordingly. They cover the domains of Process *visualisation*, Project *planning*, project *execution and control* (referring to monitoring and re-planning) and process *development* (referring to knowledge management and other continuous improvement activities). Browning's review focuses on process management. An alternative set of modelling objectives is introduced here to better reflect the applications discussed in Chapter 1:

- **Description and documentation** purposes are usually addressed by models which provide a semi-formal, diagrammatic representation of process connectivity. Since process description often requires the manipulation of large volumes of data, descriptive frameworks usually provide hierarchical structures to support this. In common with other informal or semi-formal modelling approaches, the content of a descriptive process model is subject to interpretation by the reader. Few constraints are placed on the nomenclature and structure of such models, which are consequently not amenable to most computational analyses.

- **Simulation** uses a model to approximate the behaviour of a system upon which it is not possible or cost-effective to experiment directly. Although care must be taken to understand the limitations of insights gained through simulation, few alternatives are available to explore the behaviour of design processes. These systems are unsuited to direct experimentation due to complexity, access limitations and slow response to any changes which are introduced.

  Task network modelling approaches used for simulation must provide a formal, computable language for describing process structure and/or behaviour. In addition, many simulation algorithms require that models are well-structured to ensure termination or convergence. As a result, simulation models are often more rigorous than purely descriptive models.

  A number of techniques have been applied to simulate design processes based on task network representations, most notably *discrete event* and *time-stepping* Monte-Carlo approaches. In the former case it is usually assumed that process state evolves in discrete steps, *i.e.*, that tasks release information only when work is completed; in the latter, discrete time steps are used to approximate the continuous transfer of information during task execution. Most design process simulation literature focuses on analysis techniques and does not explicitly address the issues involved in constructing large models.

- **Process integration and automation** requires logical models which can be used to directly drive computational design codes. Note that this category refers to automation of *design*, not business processes. Business process (workflow) automation considers the *availability* of information as driving process behaviour, whereas design process automation requires a more sophisticated representation to adequately control the iterative behaviour of most design processes. In such cases, it is necessary to interpret the *content* of information packets to decide which tasks are most appropriate to attempt and at what level of fidelity. Jarrett, for example, uses the accuracy to which design parameters are known to guide the step-wise selection of preliminary compressor design codes (Jarrett, 2000). His model assumes that accuracy is correlated to the codes used at each step in the process.

This thesis is concerned with process modelling to support both description and simulation. In forthcoming sections, discussion of task network models is therefore organised around the approaches' representation of task connectivity:

- **Task precedence models** capture interactions between tasks in terms of *information precedences* which imply temporal precedence.[4] A relationship between two tasks in such a model usually indicates that the sink task cannot be attempted until the source task has been completed.[5]

- **Task dependency models** capture *information dependencies* between tasks. A dependency is weaker than a precedence since it indicates only that the sink task requires information which is produced by the source task. Design tasks are commonly interdependent, each requiring information that the other produces. A dependency model describes such interdependencies but does not indicate how the problem should be solved. For example, an initial estimate might be made and the two activities repeated in a pattern of iterative refinement; or they might be undertaken concurrently, facilitated by frequent information exchange (*e.g.*, Eppinger 1991). Analytical approaches which incorporate dependency models are based on the premise that a process can be improved by studying the structure of the underlying information flows.

- **Dynamic task models** differ from precedence models in that task ordering is not explicitly captured, and from dependency models in that task connectivity is not directly represented. Instead, such models view design as a dynamic process organised around the changing state of the product (*e.g.*, Clarkson and Hamilton 2000). Unlike precedence and dependency models, dynamic task models are based on a solution-oriented perspective of the design process.

[4]The weaker information precedence implies that information produced by some iterations of Task A may be considered during some iterations of Task B, whereas the stronger temporal precedence implies that a particular instance of Task A occurs before a particular instance of Task B. Both forms of precedence indicate a sequence for attempting tasks.
[5]Some models allow tasks to overlap during execution.

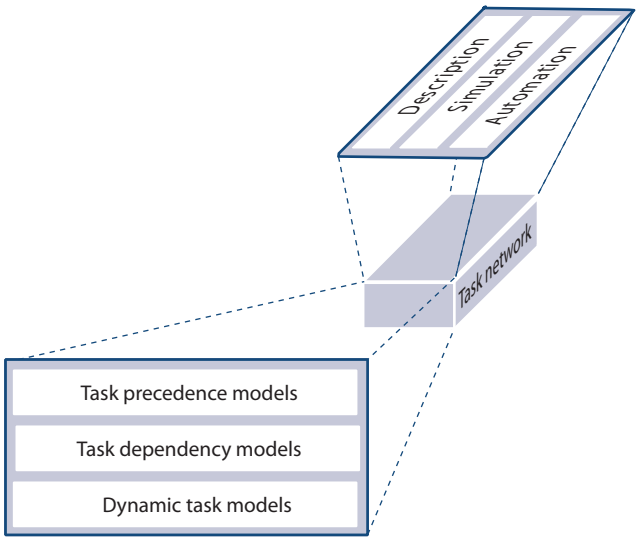


Figure 2.14 A framework for organising task network models of the design and development process.

The framework for organising task network models is illustrated in Figure 2.14. The following three sections review these approaches, focusing on task precedence, task dependency and dynamic task models respectively.

2.5.1.2 Task precedence models

The *IDEF3* process description capture method allows representation of a process using *Units of Behaviour (UOBs)* and *Junctions* to represent the flow of work, and *Object States* to represent the information transformed during the process (Mayer *et al.*, 1995).[6] The method incorporates two complementary diagrams: The *process flow diagram* (Figure 2.15), which illustrates a recurring flow of UOBs in a particular context, and the *Object State Transition Network (OSTN)*, which illustrates how the UOBs transform a given object between states (Figure 2.15). UOBs may be hierarchically decomposed into additional process flow diagrams. Multiple decompositions of any UOB are allowed, each representing the work flow from alternative perspectives. Since the IDEF3 approach was not originally intended for construction of computable models, multiple UOB decompositions need not be consistent.

[6]The ICAM DEFinition languages (IDEFs) are a family of modelling approaches which date from the 1981 US Air Force Program for Integrated Computer-Aided Manufacturing (ICAM). The family consists of 16 separate methods, of which the most relevant to this work are the *IDEF3 process description capture method* and the *IDEF0 function modelling method*. Software systems to support IDEF modelling are commercially available from *Knowledge Based Systems, Inc (KBSI)*.

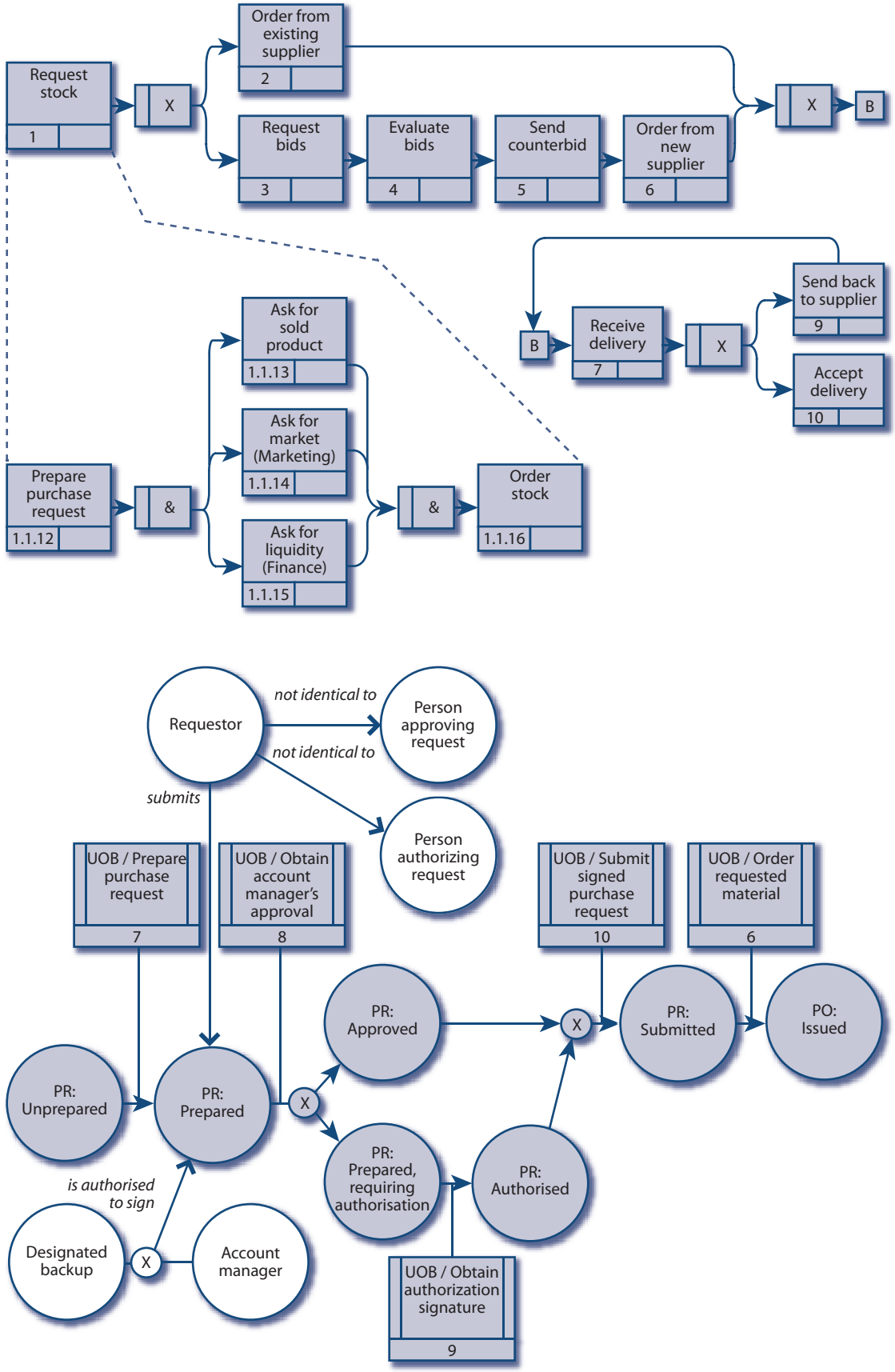


Figure 2.15 An IDEF3 process flow diagram (top) and object state transition network (bottom) (adapted from Mayer *et al.*, 1995).

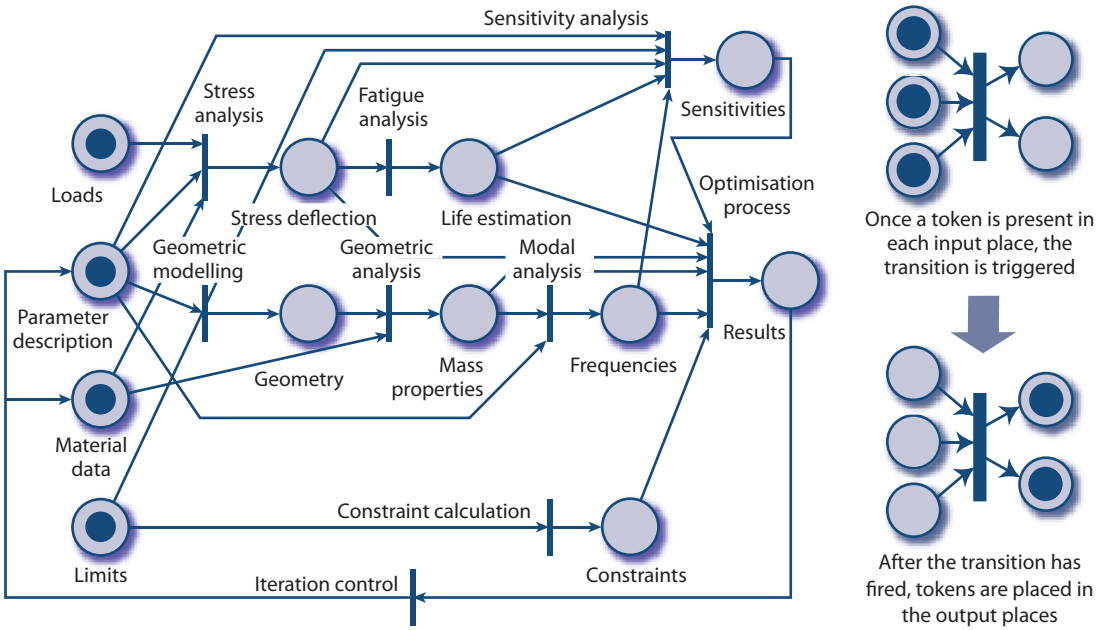


Figure 2.16 A Petri net model of a crankshaft design process (McMahon and Xianyi, 1996).

Petri nets are graphical representations of logic networks, consisting of *places* (circles) connected by *transitions* (vertical bars) via which *tokens* (solid dots) may move. In the basic form, a transition may fire when all its input places contain one or more tokens, at which point one token is removed from each input place and one token added to each output place. Extensions to the approach include logic gates to specify firing conditions and multiple outputs which are selected stochastically. *Coloured Petri nets* assign unique values to each token, thereby allowing information to be tracked through the network (Peterson, 1981).[7]

In many parametric design activities much human effort is expended in transferring and translating data between computer programs. To reduce this cost, McMahon and Xianyi (1996) propose that coloured Petri nets may be used for dynamic modelling of the information flows between design tasks. In this approach, tasks are modelled as Petri net transitions and places are used to represent *data states*. The scheme can represent serial, parallel and coupled tasks, the latter of which manifest as iterative loops in the net. This process model forms the basis of an automatic controller which directs a number of computer processes,

[7]The general form of Petri net should be categorised as a queueing model. However, in the application described here the net is structured such that each place contains at most one token at any time. Hence a queue of work can never form and the model is more appropriately described as a task precedence network.

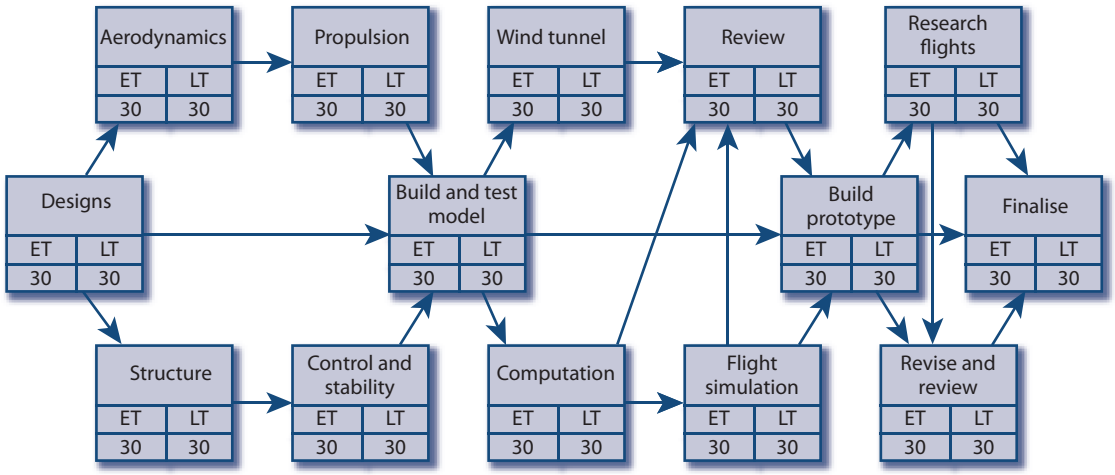


Figure 2.17 A PERT chart.

each performing a design task such as geometric modelling or analysis. These are co-ordinated using the Petri net model to solve parametric design problems in a predetermined manner. The framework is hierarchical in that transitions may represent a lower-level logic network with equivalent input and output places. McMahon and Xianyi apply this approach to automate engine crankshaft design (Figure 2.16). They propose that it may also be useful for process documentation, and suggest that re-using sections of existing networks could facilitate this (McMahon and Xianyi, 1996).

Critical path analysis refers to a set of techniques for analysing a process to determine the ultimate effect of delays in individual tasks. Most simply, the critical path is defined as the set of tasks for which any delay will propagate to the end of the project. Conversely, faster completion of non-critical tasks will not affect the schedule. Critical path analyses may thus be applied to determine where effort and/or resources should be concentrated for greatest effect.

The simplest form of critical path analysis is known as the *Critical Path Method* (PMI, 2004). This requires the process to be modelled as an acyclic network of task precedences and durations. It is assumed that every task in the network is attempted exactly once; the critical path is then the route from source to sink with longest total duration. The *Program Evaluation and Review Technique (PERT)* extends this approach by accounting for both the expected time (ET) and longest time (LT) of each task (Figure 2.17).

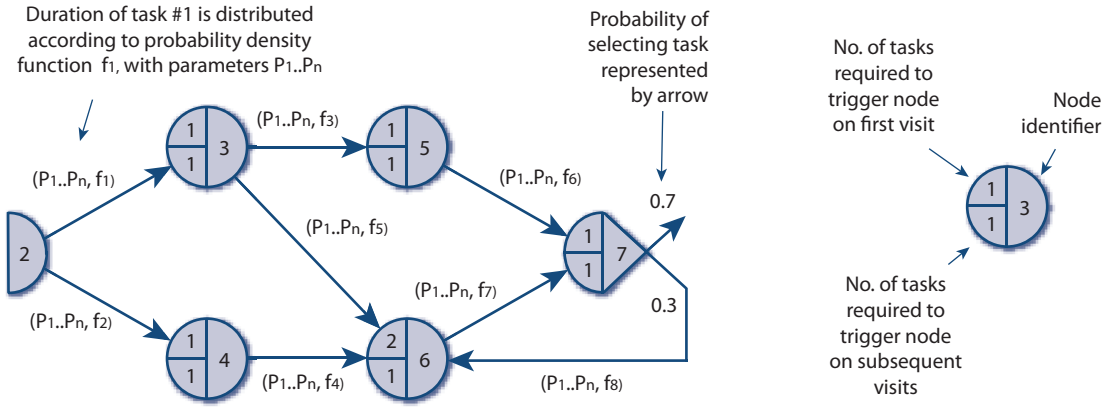


Figure 2.18 A GERT activity-on-arrow diagram (Adapted from Douglas, 1978).

Graphical Evaluation and Review Technique (GERT) was developed by Pritsker to address limitations in PERT modelling (Pritsker, 1966). GERT models may include tasks with multiple outcomes that are selected stochastically, thereby allowing non-essential tasks to be modelled and the possibility of test failures causing iteration to be incorporated. Task durations may be specified in terms of probability density functions and iterative cycles may be represented. The number of predecessors required to start a task may be specified for the first and subsequent iterations, thereby incorporating additional logic (Figure 2.18). Pritsker argues that analysis of this stochastic model can provide insights into process behaviour. For example, Douglas describes a stochastic critical path technique which aims to determine the criticality of tasks in a model which accounts for uncertainty. In such cases, tasks' criticality may be conditional upon the state of the process when the task is attempted (Douglas, 1978). Andersson *et al.* (1998) apply Monte-Carlo simulation to analyse an extended GERT-style model in which the durations and likelihoods of success of each task are specified using functions of the number of previous attempts. They argue that these non-linear elements allow more accurate modelling of iteration by representing the effect of *in-situ* learning on reducing task duration.

A key benefit of graphical task precedence approaches such as IDEF3, PERT and GERT is their intuitive flowchart-style notation. However, they can be unwieldy if a model's structure is complex or incorporates many concurrent tasks.

ProModeller is based on the configuration and hierarchical combination of *process elements* drawn from a standard library comprising around 50 objects

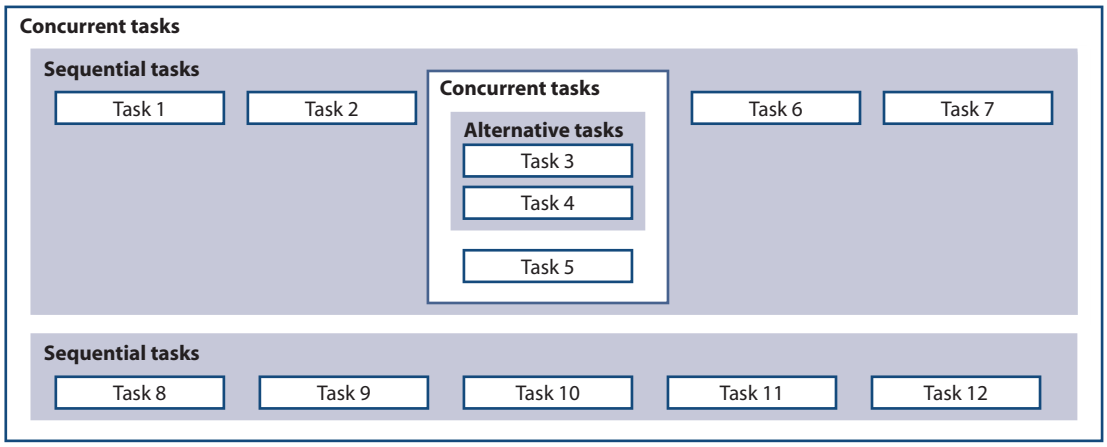


Figure 2.19 The ProModeller user interface (adapted from Vajna, 2005).

(Friesleben and Vajna, 2002). Process elements may represent technical and business activities in terms of cost, resources and working steps, and are also used to define model structure. Each structural element indicates that its nested tasks are attempted in a specified fashion: sequentially; in a cycle of iterative refinement; concurrently; or one is selected from a set of alternatives (Vajna, 2005; Figure 2.19). The reflection of process behaviour in structure ensures that all models are logically correct; unlike the simulation models discussed above it is not necessary to validate network structure prior to analysis. This facilitates the distribution of modelling effort among many process participants. However, care must still be taken to ensure that numeric values such as task durations are appropriately calibrated, which may be difficult if no-one has sufficient overview of the model. Additionally, since the user interface is logical rather than diagrammatic it may provide a less intuitive representation than the graphical task precedence approaches discussed above.

Business process management (BPM) and *business process re-engineering (BPR)* methodologies aim to support the continuous development of business processes, and consist of three complementary activities: *process design*, *process implementation*, and *process control*. A number of commercial software suites such as IDS Scheer AG's *ARIS platform* are available (Scheer, 2000), many of which are based on task precedence models. These approaches are oriented toward business processes and are less appropriate to model technical design activities. Many other workflow packages focus on visual representations and are not based on logical models which could be used for analysis (Basu and Blanning, 1997).

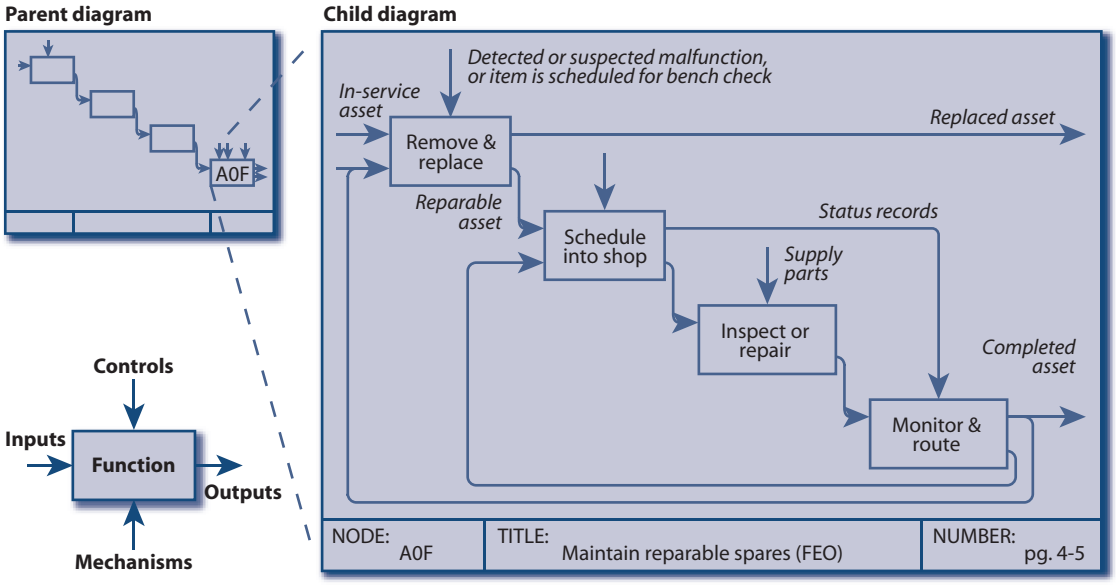


Figure 2.20 An example IDEF0 function model (NIST, 1993).

2.5.1.3 Task dependency models

To recap, task dependency models represent the information flows between tasks rather than the procedure of attempting them. In contrast to the precedence models discussed above, such models are often underconstrained – in other words, a range of alternative task sequences could be used to complete the process.

IDEF0 is a function modelling approach designed to capture the decisions, actions, and activities of an organisation or system (NIST, 1993). The approach consists of a formal graphical language for representation and a methodology for modelling. Functions are structured hierarchically and interconnected to constrain how they are triggered and controlled (Figure 2.20).

Functions in IDEF0 are specified by their *inputs*, which are transformed by the function to produce *outputs*; *controls*, which specify the conditions required for, or constraints upon the function's operation; and *mechanisms*, which represent the resources that execute the function. These relations are represented by arrows and are differentiated by the sides of the function boxes to which they connect. The full syntax includes notation to link functions across and between levels in the hierarchy. Kusiak *et al.* discuss application of IDEF0 to support re-engineering of design and manufacturing processes (Kusiak *et al.*, 1994).

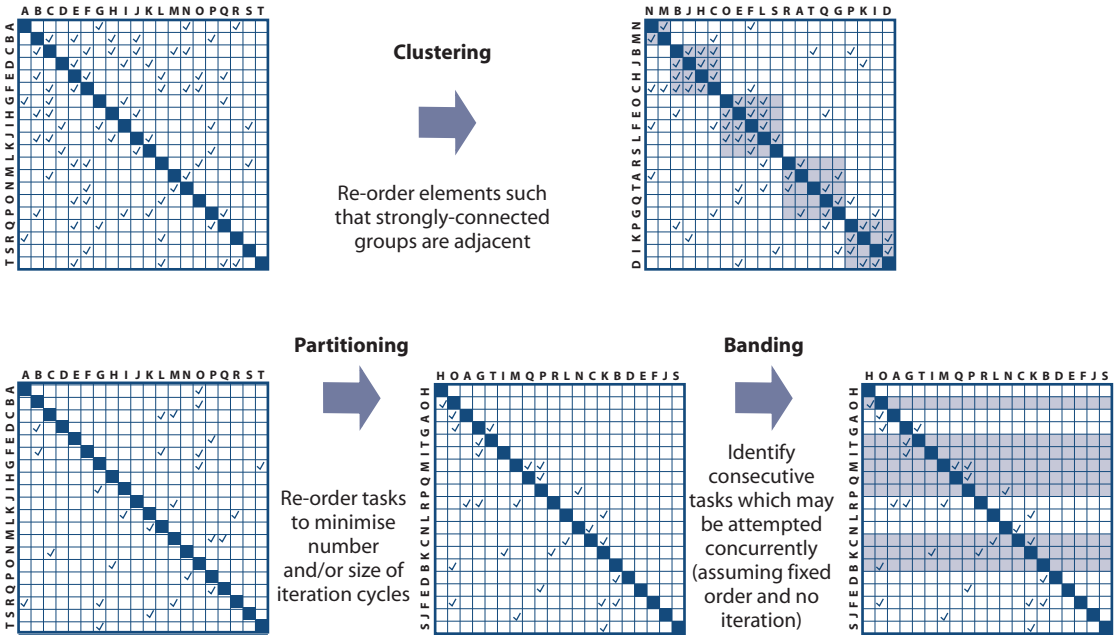


Figure 2.21 The dependency structure matrix provides a concise representation of digraph system models. A variety of algorithms may be used to extract information from the models, including the clusters of information flows and their interfaces (top) and the most appropriate order of attempting tasks to minimise the impact of iteration (bottom).

Dependency Structure Matrices (DSMs) are matrix representations of digraph system models. They are square matrices with the rows and columns labelled with the names of the nodes of the digraph, in the same order. Digraph arcs are encoded by placing a mark in the relevant cell of the matrix. The DSM is usually constructed so that the element in the row depends upon that in the column. For any digraph with n arcs there are $n!$ possible orderings of the rows and columns in the DSM.

In the context of design, the DSM is known as the *Design Structure Matrix* (Steward, 1981; Eppinger, 1991). Browning distinguishes between four types of DSM, namely: the *Activity* (or *Schedule DSM*) for modelling dependencies between tasks; the *Component DSM* (or *Architecture DSM*) for modelling relationships between components or sub-systems; the *Parameter DSM* for representing parametric interdependencies; and the *Team* or *Organisation DSM* used to model organisational structures (Browning, 2001).

The DSM can reveal characteristics of the connectivity relationships within a model. For example, if all the marks lie below the leading diagonal in one or more of the possible orderings, the DSM represents an acyclic graph; in the case

of an Activity DSM, this indicates that the design process may be completed by attempting tasks sequentially or in parallel. Conversely, if it is not possible to find such an ordering the graph must contain at least one cycle; some of the design tasks are interdependent and may involve iteration. Several algorithms have been developed to analyse a DSM to exploit such structural characteristics. These include: *partitioning*, attempting to find a lower diagonal reordering, *i.e.*, finding a sequence of tasks to minimise information feedback and therefore reduce the possibility of iteration; *banding*, identifying independent elements in a partitioned DSM, *i.e.*, tasks which may be attempted in parallel (Grose, 1994); and *clustering*, attempting to group elements into strongly-connected sets with low inter-cluster connectivity. Examples of these operations are depicted in Figure 2.21 above.

A number of authors have used the DSM as the basis of process simulation. For one example, Carrascosa *et al.* use a time-stepping simulation based on the Activity DSM to model the effects of coupled tasks which create rework for one another. In their model, each task dependency is parameterised in terms of the probability of the source task creating rework (likelihood) and the amount of rework which is created for the sink task (impact). These are represented as functions of the time spent executing the task (Carrascosa *et al.*, 1998).

Domain-Mapping Matrices (DMMs) are an extension to the DSM which allows modelling of linkages between different types of element. Danilovic and Browning (2007) discuss application of DMMs to explore connectivity between the process domains of *tasks*, *components* and *teams*. By analysing the clustering of each domain independently and in combination it is possible to identify mismatching structures. For example, a team structure which does not reflect the ordering of tasks in the process may contribute to communication overhead or rework (Kreimeyer *et al.* 2007). An example DMM analysis is illustrated in Figure 2.22 overleaf.

One factor which may contribute to the popularity of DSM- and DMM-based modelling approaches is their ease of implementation using spreadsheet software. However, while these approaches provide a compact notation and are effective for exploring the architecture of a process, they are not well-suited to convey detail — for instance, it is easy to misplace marks when constructing large ma-

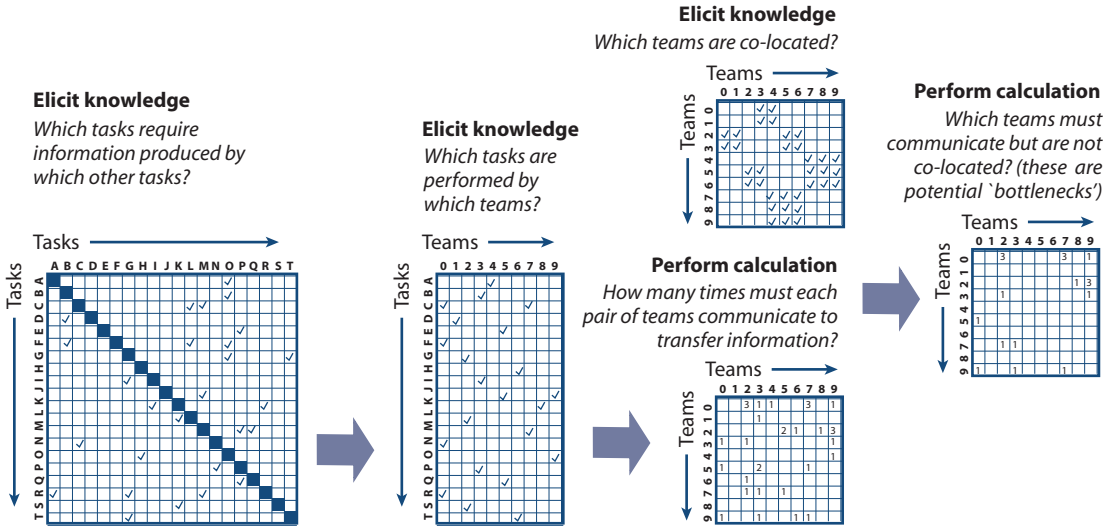


Figure 2.22 An example application of domain-mapping matrices to highlight mismatches between process and organisational domains.

trices, resulting in the introduction of potentially significant error to the analysis results. Additionally, it could be argued that the more sophisticated extensions compromise the pragmatic simplicity of the approach.

ADePT PlanWeaver is a planning support tool for the construction industry which draws upon the IDEF0 and DSM representations. The approach is based on an IDEF0-style representation which captures the information transfer underlying each dependency, together with the strength of the dependency (Austin *et al.*, 1999). The ADePT approach is based on a library of generic construction processes which are used to construct a customised process model for a specific project, expressed in the form of an Activity DSM. Tearing 'strong' dependency loops in the DSM allows the project to be sequenced and a detailed schedule to be produced. In case studies carried out using the ADePT method, the process library accounted for more than 90% of the activities required for a typical construction project model. This reduced the time required to develop models comprising 200 to 260 tasks to between 26 and 40 hours (Austin *et al.*, 1999).

In summary, the task dependency representation forms the basis of many of the analytical approaches in the literature. It also appeals to industry, perhaps because it does not require a detailed understanding of the order of task execution prior to modelling. In the context of IDEF0, Austin *et al.* identify one disadvantage of dependency representations in that untrained readers tend to incorrectly assume a task sequence (Austin *et al.*, 1999).

2.5.1.4 Dynamic task models

Dynamic task models are based upon the assumption that the design process may be modelled as a set of tasks concerned with identification, estimation and iterative refinement of key design and performance parameters. These models consider that the process is organised dynamically around the state of the design. They assume that tasks are selected *in-situ* by considering the levels of quality, maturity or value of design parameters. As a consequence of specifying processes in terms of knowledge about individual tasks, these models typically imply a wide range of possible processes which emerge from the underlying information structure of the model.

Signposting was proposed by Clarkson and Hamilton to support aerospace design by identifying the set of tasks which are appropriate at each step, based on the designer's current *confidence* in their solution (Clarkson and Hamilton, 2000). The meaning of confidence in this model depends upon the parameter being described. According to the original definition, having high confidence in a parameter usually means that its value is detailed, accurate, robust, well understood, and physically realistic (Hamilton, 1998).[8] Tasks are represented as knowledge transformers which are appropriate only when specified levels of confidence are achieved in the input parameters. Upon completion a task causes the confidence in output parameters to change in a specified manner. Quality tends to increase as the design progresses (Hamilton, 1998); a process is considered complete once a sufficient level is achieved in each parameter.

At each step in the process, the Signposting tool presents a list of tasks which are both *available* and *appropriate* to attempt (Clarkson and Hamilton, 2000). A task is considered *available* if the current design confidence is greater than or equal to that required by the input conditions of the task, and *appropriate* if execution of the task would lead to an increase in confidence in one or more parameters. The approach is adaptive in that guidance responds to the emerging state of the design.

[8]This definition of confidence differs from design maturity since it decreases when tests reveal problems in the design. However, design maturity does not necessarily decrease following test failure, since the information generated by a test can be used to propose an improved solution.

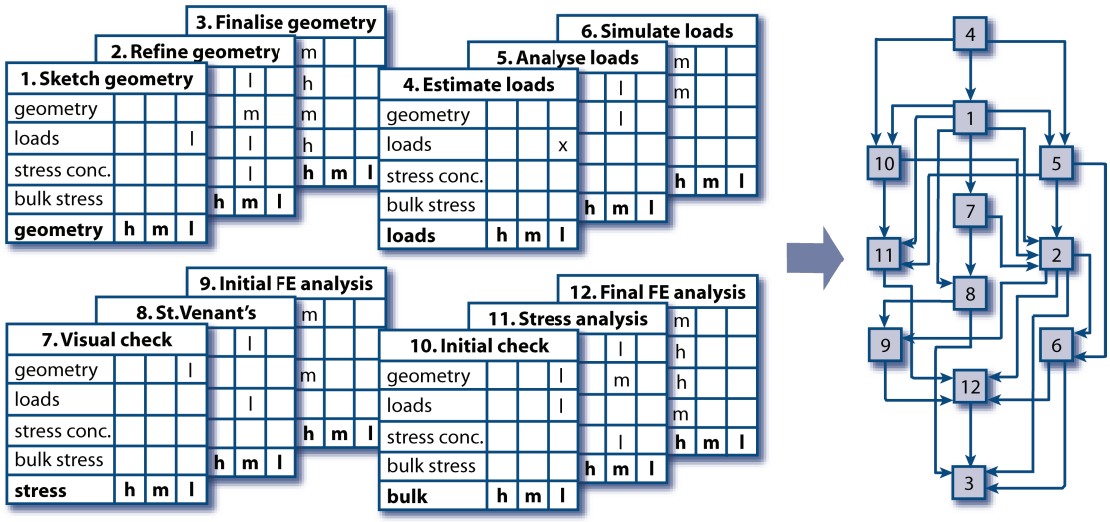


Figure 2.23 Signposting models are based on a simple graphical notation in which each task's context may be elicited in isolation (left). They usually imply a wide range of possible task sequences (right) (Adapted from Clarkson *et al.*, 2000).

The knowledge base for a Signposting model is elicited using the graphical notation of Figure 2.23. Each task is presented as a matrix in which the bottom row indicates the single output parameter modified by the task. The upper rows indicate the minimum levels of confidence required to attempt the task. For instance, Figure 2.23 indicates that the task 'Sketch geometry' requires the 'loads' parameter to at least *low* confidence; completing the task sets the 'geometry' parameter to *low* confidence. Tasks are structured into groups to support knowledge elicitation and navigation within the tool.

MIDAS (Manufacturing Integration and Design Automation System) is a grammar-based approach for managing the execution of dynamic processes (Chung *et al.*, 2002). The design process is represented as a set of *logical tasks* representing high-level goals, and *atomic* tasks which represent and encapsulate individual computer tools. Inputs and outputs are defined in terms of *data specifications*, which form a hierarchy of increasing detail. A task's inputs and outputs are described at a level of detail commensurate with the task description.

The system proceeds by prompting the user to specify a high-level sequence of logical tasks, together with clarifying information such as the required accuracy of the solution. The grammar replaces each logical task with one or more atomic tasks, chosen such that the inputs and outputs are more specific versions of the

original data specifications. Where more than one possibility for replacement exists, the instantiating tasks are selected to minimise an objective function defined by the task itself. For example, such a function might evaluate the computational time required for analysis given the current state of input data. The process may be viewed as a flow network at any point in this instantiation process. Chung *et al.* argue that a key benefit of the approach is its ability to generate alternative configurations when unexpected outcomes are encountered during process execution.

Extended Signposting was developed by Melo (2002), O'Donovan (2004) and Flanagan (2006) to simulate concurrent design processes. The model extends Signposting to include: a triangular probability density function (PDF) defining the duration of each task; multiple outcomes with a probability of each occurring; resources required by each task; the distinction between *necessary* and *useful* input information for each task and the effect of this input quality on output quality.[9] The definition of parameter was also extended to allow representation of any information which may change during the course of a process. This could include project-focused information such as documentation, reports and other deliverables (O'Donovan *et al.* 2004).

O'Donovan (2004) used Monte-Carlo simulation to develop a state-contingent plan representation termed the *Conditional Precedence (CP) matrix* from this model. When multiple tasks are possible the CP matrix indicates which should be attempted first to minimise process duration. It is contingent in that it provides guidance from any situation which may occur (Figure 2.24).

[9]Multiple outcomes were incorporated as alternative confidence transformations. For example, if technical problems were encountered when executing the 'detail design of gripper' task confidence in the 'gripper geometry' parameter would fall. The reduction in confidence would cause certain tasks to be re-executed. Alternatively, the task might be successful and confidence in 'gripper geometry' would increase.

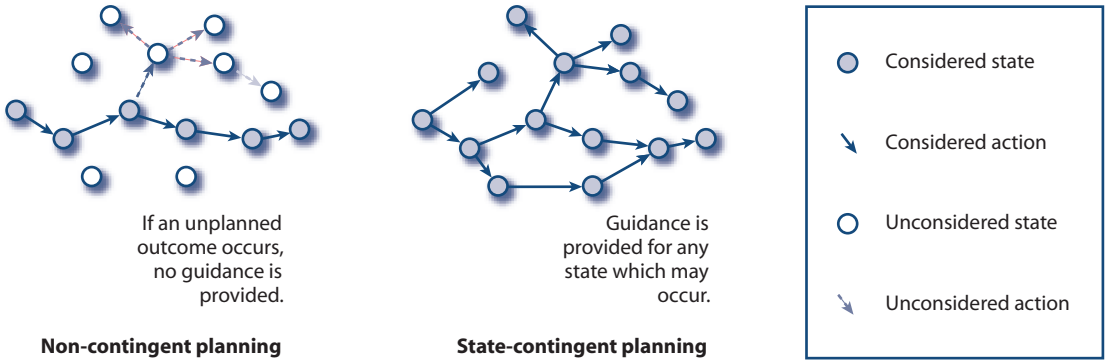


Figure 2.24 Non-contingent and state-contingent planning (Adapted from O'Donovan, 2004).

The *Adaptive Test Process (ATP)* is similar to Signposting in modelling design activities in terms of tasks, parameters and their quality (Lévárdy *et al.*, 2004). ATP tasks have a number of alternative *modes*, each of which represents a different fidelity of the activity. Modes thus allow the modelling of activities whose characteristics are contingent upon the context of execution.

The ATP represents *Technical Performance Measures (TPMs)* and the uncertainty associated with these measures in terms of a triangular probability density function for each performance attribute. In addition to modifying the confidence in design parameters, an ATP testing task has the effect of reducing risk by tightening the spreads of certain performance measures. As a process progresses, therefore, the confidence in design parameters increases and the technical risk decreases. At each step in the process, the ATP simulation attempts to choose the best mode of each activity by minimising the product of cost, duration and the reduction in spread of all TPMs (Lévárdy and Browning, 2005).

In summary, Hamilton proposes that dynamic task models may be constructed from knowledge of individual tasks because their connectivity is not explicitly represented (Hamilton, 1998). This is beneficial as it bypasses the requirement for an overview of process structure — as will be discussed in Chapter 3, developing this overview can be difficult due to the specific nature of most designers' process knowledge. However, representing tasks' context in terms of information maturity levels requires that these levels are consistent within the context of a model. It is therefore important to carefully define the maturity levels when constructing a dynamic task model.

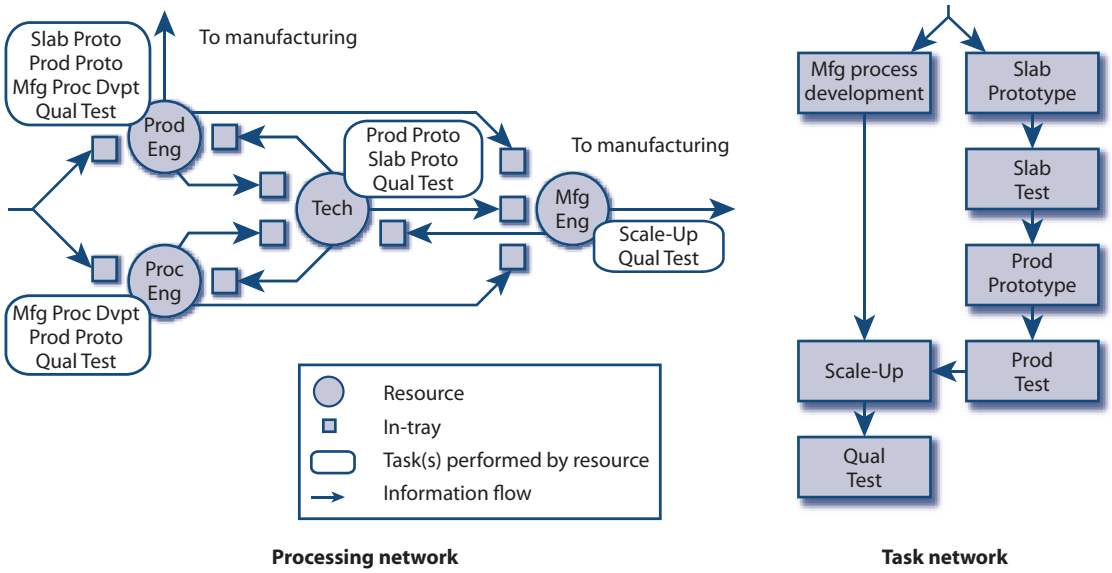


Figure 2.25 A stochastic queueing model of product development (Adler *et al.*, 1995).

2.5.2 Queueing models

The task network simulations discussed above characterise process dynamics as a transient pulse of information which flows through the activities.[10] In contrast, queueing models emphasise the repetitive nature of workflows and the effective utilisation of the limited resources available to perform tasks. They may be used to explore the steady-state process behaviour which emerges from these flows.

Adler et al. (1995) proposed the stochastic queueing model of product development shown in Figure 2.25. This model provides a simplified perspective of task connectivity, together with the processing stations (*e.g.*, design teams) which must perform each task. It is assumed that two forms of process are conducted during product development, namely new product development and change processes. In the full model, new product development is modelled as incorporating an iteration cycle whereas the change process is assumed to follow an ordered sequence of tasks. It is assumed that several development and change processes may be in execution at any given time, according to a stochastic model which governs the triggering of each process. Each processing station uses round-robin time-slicing to schedule jobs in its queue.

[10]Resource-limited task network models may also be viewed as queueing systems. However, most resource-limited models of design processes are structured such that task concurrency is primarily limited by information flows rather than resource constraints.

Other queueing models include Q-GERT, an extension of the GERT approach which supports modelling of queueing systems (Pritsker, 1979), and the model proposed by Narahari *et al.* (1999). A limitation of these approaches is the assumption of similarity of workflows over time. They are less relevant for the complex design processes studied in this thesis. In these cases, the rate of process change can be significant relative to the long duration of individual projects.

2.5.3 Multi-agent models

To recap, multi-agent models view processes as a collaboration of decision-makers who negotiate a solution to the design problem. They incorporate process simulations which may be applied to explore the influence of agent-oriented factors, such as co-ordination strategies, upon project behaviour.

Mihm et al. (2003) propose a multi-agent model which aims to develop general insights into the behaviour of complex engineering processes. They view designers as agents which are responsible for optimising a single component. When an agent changes its component in this model, the design goal of all other components is updated thereby forcing each of the other agents to modify their design. This is coupled with the assumption of limited bandwidth for transferring information between designers, such that each must operate based on outdated information about their design objective. Based on this model, Mihm *et al.* investigate the circumstances under which processes 'oscillate' between alternative design solutions. They conclude that these oscillations become less stable and more likely to occur as the design problem becomes larger and more strongly connected. They propose a number of strategies to minimise this effect, including modularisation of the product (to reduce interdependencies) and releasing preliminary design information (to reduce the delay between a design change being made and other agents being notified).

The *Virtual Design Team (VDT)* proposed by Levitt *et al.* (1999) is a multi-agent discrete-event simulation modelling framework which accounts for a variety of influences upon agent behaviours, including variable strength of interdependency between activities and incongruency between actors' goals. In contrast to the model of Mihm *et al.* (2003), the VDT is intended for application to study

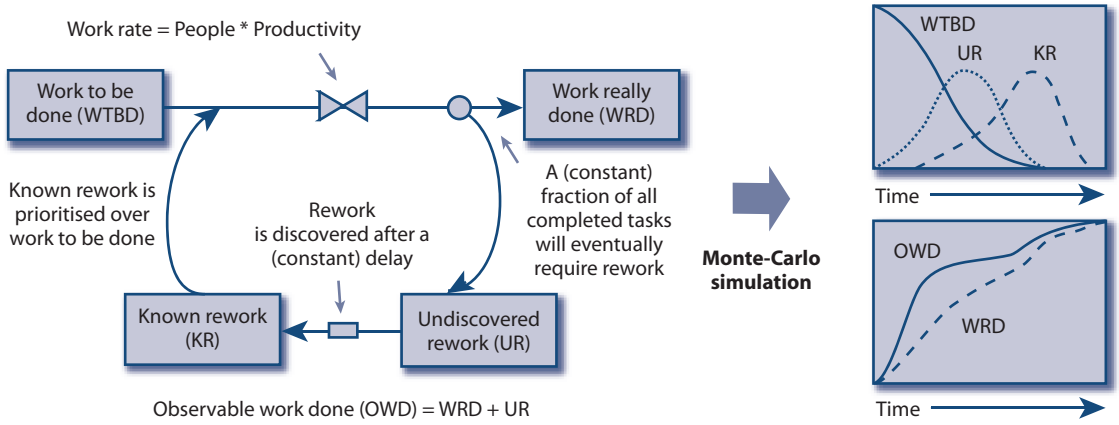


Figure 2.26 A system dynamics model of product development (Cooper, 1993).

specific design processes. For instance, Levitt *et al.* (1999) discuss a case study of satellite launch vehicle design, in which the approach was used to evaluate the effects of changes to the design process' configuration upon its performance. These changes were specified in agent-oriented terms such as 'Increase agent skill levels' and 'Align agent goals'.

2.5.4 System dynamics models

System dynamics models aim to encompass the effect of feedback and feedforward on process behaviour. In the product development literature, such models treat activities as fungible entities which flow between pools that represent their current state of execution. Since these system dynamics models do not represent the detail of individual activities they are appropriate where the focus of interest lies on the behaviour of the project as a whole rather than on individual tasks.[11]

Cooper (1993) developed a system dynamics model to investigate the effect of rework on project schedules. This approach represents tasks as interchangeable units which flow between the four activity pools shown in Figure 2.26. The rate at which tasks leave each pool is determined by project-specific constants, and it is assumed that a certain percentage of tasks leaving the *work to be done* pool enter *undiscovered rework*, with the remainder accumulating in *work really done*. The state of a task at any time is determined by the pool in which it resides. A project begins with all tasks in the *work to be done* pool and is considered complete once *work really done* contains all tasks.

[11] Browning describes system dynamics models as 'holistic'.

Cooper argues that undiscovered rework cannot be distinguished from work really done, and uses Monte-Carlo simulation of the above model to show that this can account for schedule slippage in development projects. His results highlight that the actual work completed lags behind the observed work completed at all times (Figure 2.26). Furthermore, although the observed work completed increases rapidly in the early stages of a project, this slows as rework is discovered after the initial delay. Cooper proposes this model can explain the '90% syndrome', *i.e.*, the tendency for projects to appear almost complete for an extended time.

Ford and Sterman (1998) developed a more sophisticated system dynamics model incorporating multiple *phases*, each of which comprises the four activity pools of *completion* (containing tasks being completed rather than complete tasks), *quality assurance*, *change* and *co-ordination*. The rate at which tasks flow out of a pool is determined by the project-specific constants *resource constraint* and *average duration*, as well as the number of activities in the pool at that time. As in Cooper's model, rework is modelled explicitly as the fraction of tasks leaving *completion* which must flow through *change* prior to re-entering *completion*. Tasks not requiring change enter *quality assurance* and finally accumulate in *co-ordination*. When this final pool reaches a certain size, the tasks are released into the *completion* pool of the subsequent phase. The model thus represents a gated cascade of development through each phase of a project.

The model incorporates task interdependencies within phases using an *internal process concurrence* function. This equation limits the concurrency of tasks by throttling the release rate of the *completion* activity — in effect, new tasks cannot be attempted until a certain percentage of tasks have passed quality assurance.

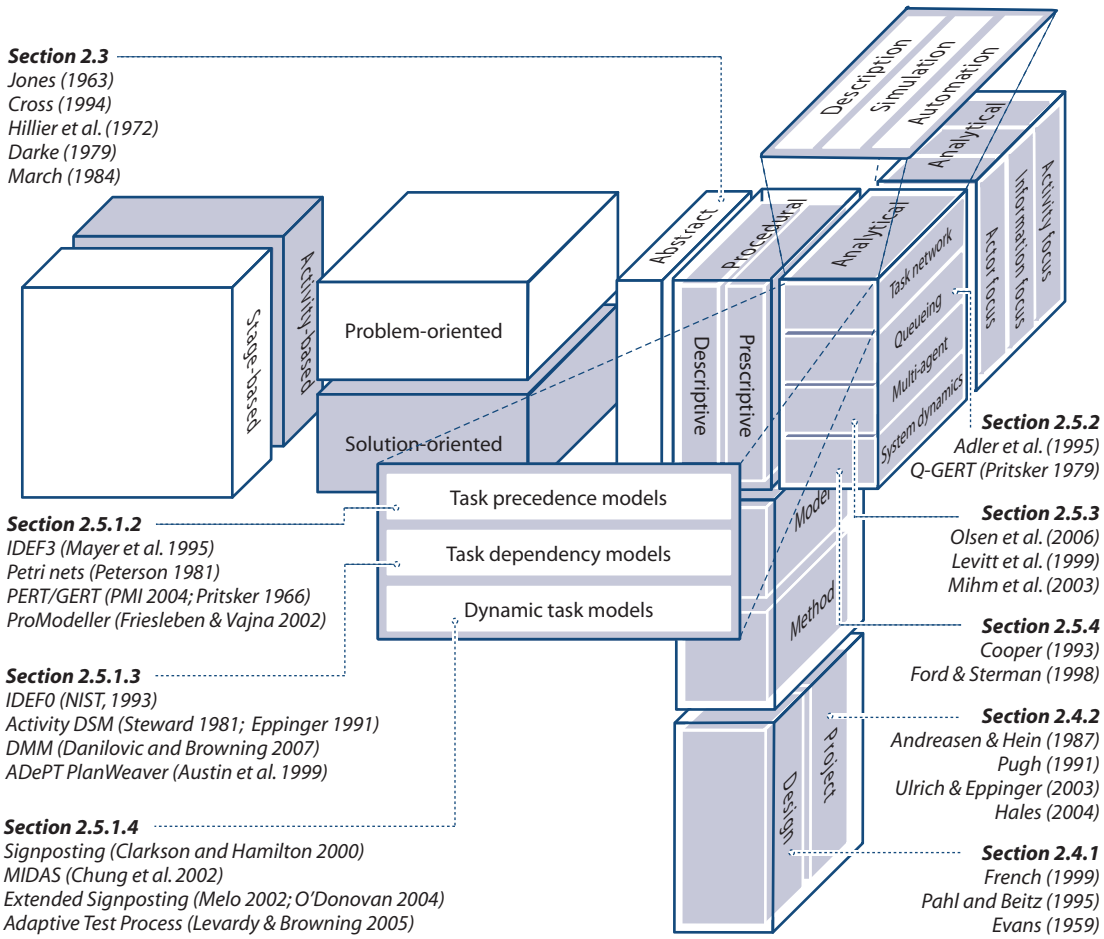


Figure 2.27 The completed framework for organising the models of design and development reviewed in this chapter.

2.6 Summary

This chapter has reviewed models of the design and development process and developed a framework to organise discussions of this area. The review characterises the modelling approaches available but does not provide an exhaustive list of implementations or examine the details of their application; this reflects the focus of the thesis. The completed framework summarises the chapter and is presented in Figure 2.27. In summary:

- The framework highlights that the theories, models and methods in the literature span a diverse range of issues and disciplines. Many of these publications examine the design and development process from a high level of abstraction. Their focus ranges from the individual designer's problem-solving process through to the need for continual business development.

Although they offer useful insights which can help to guide process improvement activities, such models are often too general to provide detailed, implementation-level advice.

- Other approaches aim to support improvements to the design and development process through the modelling and analysis of a specific domain. They provide a lower-level, more descriptive perspective of process behaviour than the abstract and procedural approaches, and are directly pertinent to the research questions stated in Section 1.2.

- Although many analytical approaches have been published, they are based upon the relatively small number of representational bases introduced above. These differ significantly in their modelling assumptions. In common with the abstract and procedural approaches, none is agreed to adequately represent all aspects of the design process.

The diversity of analytical modelling approaches raises further questions which are pertinent to the objectives introduced in Chapter 1: *How can the most appropriate modelling approach for a given modelling domain and objective be identified? Can — and should — the alternative perspectives revealed by the review be reconciled into a more comprehensive model?* These questions are explored in Chapter 3 through an extended case study of aero-engine design.

—— This page is intentionally blank ——

Chapter 3

Representing design practice

This chapter explores the utility of analytical process models to represent design practice. The research draws on an empirical study in which the author spent eight months on-site at Rolls-Royce Bristol developing a process modelling and management support approach. This highlighted that design iteration forms a key influence on process behaviour which is usually too complex to represent in a model. It is argued that each analytical modelling approach reviewed in Chapter 2 is better suited to modelling certain aspects of iteration — and that the most appropriate approach should be selected by considering the characteristics of iteration in the specific process and the purpose for modelling. It is proposed that the task precedence representation provides the most appropriate basis for modelling aero-engine component design to support project management.

Discussion proceeds in five sections. Firstly, the research objective and methods are discussed and the case study is introduced. Secondly, the component design process observed during the study is discussed in detail. Thirdly, observations of planning practice in the company are summarised and design iteration is highlighted as a major driver of planning difficulties. Fourthly, iteration is discussed in more general terms, key challenges in its modelling are identified, and the suitability of different representations for different forms of iteration is highlighted. Finally, this analysis is used to identify an appropriate approach for modelling the component design process observed during the study.

3.1 Overview

This section discusses the research objective and methods prior to describing the blisk design process in depth.

3.1.1 Research objective

The following research objective arose from the literature survey in Chapter 2:

> Explore the utility of analytical process models to represent design practice and thereby identify the most appropriate approach for a given modelling domain and objective.

3.1.2 Research methods

The objective was addressed by conducting a case study to observe industry practice. The study was initiated after the author spent 5 days at Rolls-Royce Derby assisting O'Donovan with data collection (O'Donovan, 2004).[1] Company personnel suggested the author conduct a follow-on study to extend and apply the research. A project was proposed by a programme manager who had identified a business need which could be addressed using analytical process modelling.

3.1.2.1 Company background

Rolls-Royce Group plc. is one of three leading providers of power systems in the civil aerospace, defence aerospace, marine and energy sectors. In 2007 the company employed around 36,000 personnel in 50 countries. It has annual revenues of £6.6bn at the time of writing — 54% of which are derived from engine maintenance and financial service offerings (Rolls-Royce Group, 2007).

3.1.2.2 Business need

The case study was proposed shortly after completion of Stage 2 (*i.e.*, full concept design) of the F136 aero-engine programme. The programme manager sponsoring the research was responsible for delivering F136 fan module milestones under the

[1]The author had undertaken 15 months research in design process modelling and simulation prior to the case study. The early research focused on developing methods to construct Extended Signposting models.

Figure 3.1 The GE Rolls-Royce Fighter Engine Team F136 turbofan engine (JSF, 2007).

project's scheduling and cost constraints. An important component of this role was to monitor the progress of individual work packages and ensure that potential delays were resolved in time to avoid costly consequences.

A 'lessons learned' exercise following Stage 2 completion had highlighted that design-make plans for major engine components provided a detailed representation of manufacturing processes, whereas the iterative design stage was represented at poorer resolution — some design activities were described on a scale of months instead of days. Analytical process modelling was perceived as an opportunity to develop a more accurate picture of these activities and their contexts within the project, thereby supporting planning, monitoring and the mitigation of schedule risk.

3.1.2.3 Business-oriented objective

The business-oriented objective of the case study was developed from this business need and agreed with the sponsor prior to beginning research:

- Develop a model-based approach to support the planning and re-planning of activities in the component design process, thereby improving progress monitoring and programme control.

This objective provided the context for the observations discussed in this chapter. It also forms the main research objective of Chapter 4.

3.1.2.4 Research methods

The case study comprised eight months spent on-site by the author between March and October 2004. Throughout the study the researcher observed meetings and conducted informal interviews with engineering and management personnel:

- **Informal interviews.** Informal interviews and discussions were conducted with many personnel. 21 such discussions were scheduled and documented but more were held on an opportunistic and undocumented basis. Most of the individuals worked in the engineering-focused *Compression Systems Operational Business Unit (OBU)* or the project-focused *F136 Customer Facing Business Unit (CFBU)*. A list of interviewees and a summary of the Rolls-Royce organisational structure is provided in Appendix A.

  Interviews during the first two months were concerned with understanding the company's operations, particularly to elicit the limitations of current programme management tools and practice. Later discussions focused on method development and the elicitation of specific process knowledge.

- **Meeting observations.** A number of regularly scheduled meetings and one-off workshops were observed to supplement the informal interviews. Details of these meetings are tabulated in Appendix A.

Research focused on the mechanical design of the F136 first stage fan 'blisk' due to time limitations and the complexity of aero-engine design. The blisk is a major component of the fan module whose timely delivery was critical to project milestones. The technical challenges of blisk design were only examined to the depth required to address the research objectives.

Due to the opportunistic nature of the study and the primary aim to improve rather than describe industry practice, a rigorous empirical methodology was not followed. The discussion in forthcoming sections therefore reflects the author's interpretation of those issues considered important to the thesis, informed by extensive notes taken during the study and by later discussions with industry personnel. Quotations used in the chapter were selected from notes taken during interviews and are presented to clarify rather than validate the argument.

3.2 Component design practice

This section describes the observed design practice under the following headings: the fan blisk; the blisk design process; limitations in process knowledge; and representations of the design process.

3.2.1 The fan blisk

The fan blisk is a complex component which must satisfy a number of technical requirements.[2] For example, it must exhibit certain aerodynamic properties as a part of the fan module; it must resist the steady stresses caused by rotation and the periodic vibrations caused by aerodynamic interaction with the inlet guide vanes. As the foremost rotative component in the main gas path it must be resilient to impacts from ingested matter. The blisk must also satisfy various materials and manufacturing constraints. Issues such as testability, manufacturability, maintainability and disposal are considered to ensure economic viability over the engine's lifecycle.

A successful blisk design process satisfies business-oriented objectives in addition to the technical requirements outlined above. For example, most engine contracts stipulate that modules are rig-tested on specified dates during development; this requires the release of intermediate blisk designs for prototype manufacture. This must be achieved with limited human, computational and manufacturing resources.

Aero-engines are developed in large part by modification from existing designs. In this environment, radical designs carry a high risk and most designers participate in few truly novel projects during their careers. Just as the form of the blisk is constrained by the need to satisfy many design objectives, so its design process is determined by the available design technology and experience.

[2] A 'blisk', or *bladed disk*, is a single row of aerofoil blades welded to a rotating disk on an engine shaft. This design has lower weight than configurations in which individual blades or blade sets are attached to the disk using conventional fixtures (Rolls-Royce, 2005).

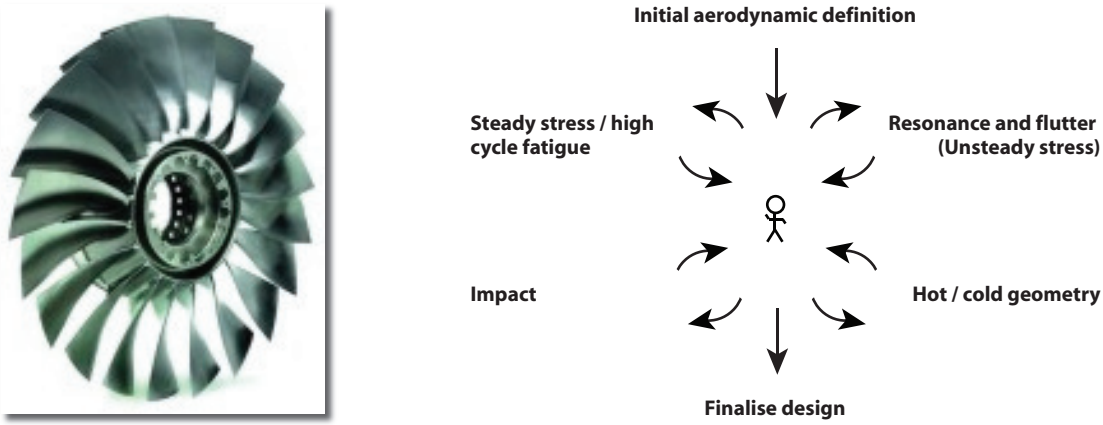


Figure 3.2 An example fan blisk and an overview of its design process, focusing on the mechanical design activities.

3.2.2 The blisk design process

From the mechanical design perspective, design proceeds through iterative refinement of the blisk parameters via four main activities (Figure 3.2), which may be decomposed into increasing levels of detail. Each activity may be attempted at different resolutions according to the quality of the input data and perceived importance of the corresponding objective at that time. For example, a low-fidelity *steady stress* analysis may be applied to preliminary aerofoil data to ensure viability of the design concept. Later, application of the same analysis to a model of the assembled blisk allows verification of the completed design. In general, the effort devoted to each activity increases over time to reflect design maturity.

Activities may be performed concurrently to support early estimates or compress schedules. For example, *impact analysis* is a lengthy activity which must be undertaken in parallel with others. Because it is computationally expensive and sensitive to the geometry definition, impact analysis would ideally be attempted only when major design alterations are no longer expected.

Blisk design may be described in terms of the iterative application of design and analysis tools used to perform specific activities. The tools and their interactions are well-defined, although the order of application is difficult to describe as it is determined by *in-situ* decisions. According to the process documentation:

> *"There is no correct sequence for the mechanical design. The optimum sequence will depend upon the starting point and the relative sensitivities of the design objectives to the proposed design style."* — Technical process documentation.

Changes in design constraints may be required at any time to accomodate emerging issues in other engine sub-systems or customer requirement changes. As a high-performance component the fan blisk usually cannot absorb such change:

> *"Any externally-driven change to the design definition could require all mechanical design processes to be revisited."* — Technical process documentation.

This redesign may occur several times during an engine development project.

3.2.3 Limitations in process knowledge

Understanding the relationship between the form of the blisk and any design objective requires specialised domain knowledge. Consequently most aerospace engineers are trained in specialist disciplines and are expert in the application of particular tools to certain design or analysis activities. Many possess limited knowledge outside their area of specialisation. For instance, it may not be expected — or necessary — for an aerodynamicist to have a detailed understanding of a fan blade's material properties or the processes by which these properties are assessed.

Whereas design engineering requires a detailed understanding of specialist methods and tools, design management requires overview of many issues which span the domains of product, process and personnel. Although managers therefore possess a broader picture, their technical understanding is often strongest in the discipline in which they trained. During the case study, no individual could describe the design process in its entirety.

This distribution of process knowledge among participants is characteristic of all complex design processes (Eckert and Clarkson, 2003). It was widely recognised in Rolls-Royce that, while limited overview is often unavoidable, it can lead to difficulties in identifying and prioritising design activities.

3.2.4 Representations of the design process

Several descriptions of the blisk design process were examined during the study (see Figure 3.3 for four examples). These process maps focus on technical aspects of the design process, are almost exclusively confined to a single discipline and were constructed using standard office productivity tools. They describe the process in terms of key software packages which interact through the transfer of data types and other design descriptions. Most are structured as informal flowcharts which indicate information flows and represent potential rework as cycles on the diagram (Diagrams 1, 3 and 4 in Figure 3.3). It was suggested by one engineer that they emphasise ambiguous or risky elements of the process. In contrast, Diagram 2 represents the design process as a set of hierarchically-structured objectives. The stress engineer who constructed this diagram indicated that he had chosen this representation to highlight the disordered nature of iteration, which could not be easily represented in a sanitised flowchart.

These documents were produced by members of the design teams to represent technical process information relevant to a small number of personnel. As can be seen in Figure 3.3, there is little agreement on notation or format between the process maps. The diagrams are informal and proved difficult for the author to interpret without prior domain knowledge. Most were considered outdated to some degree; one designer suggested that the effort required to maintain and update documentation was prohibitive due to continuous improvements in design technology. The limited audience and specialist function of these descriptions, combined with constant time pressures, appeared to provide little incentive for co-ordinating documentation activities at this level. A more integrated picture was provided by the company's intranet site, which provided a higher-level, more prescriptive view of the development process.

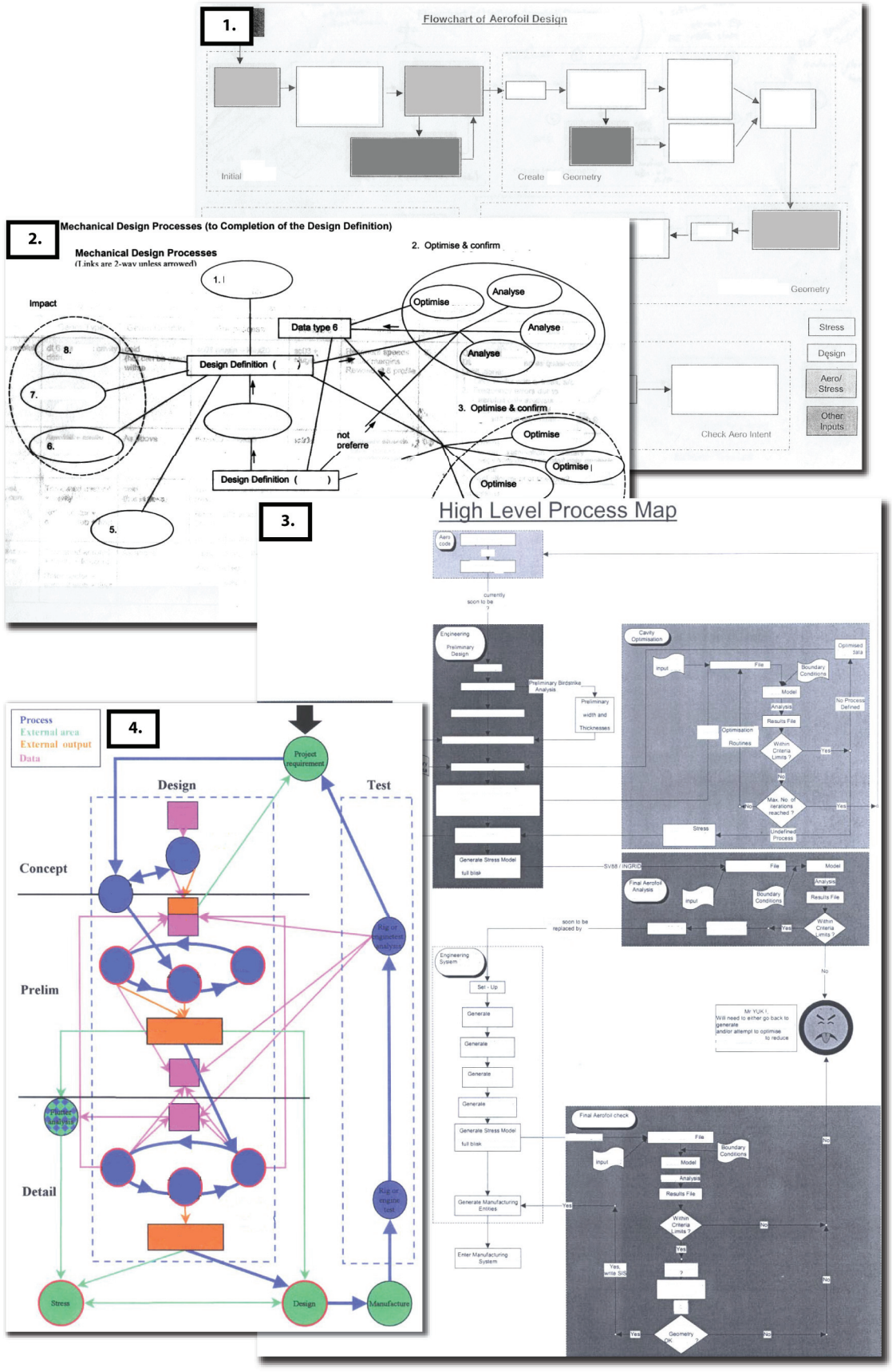


Figure 3.3 Some examples of process documentation examined during the case study (some text obscured to protect commercial sensitivity).

Two models of the whole engine design process were examined in addition to these component-level process descriptions. They were based on Excel spreadsheets tabulating tasks, deliverables and deadlines. They did not represent design iterations or the low-level detail of component design. A company specialist involved in the development of one such model indicated that overcoming differences in perspective and terminology was a key challenge during modelling:

> *"We tried to subdivide the [process modelling] problem... but when the pieces came back we couldn't fit them together."* — Process improvement specialist.

In general, process models and documentation appeared to provide limited guidance for the planning, execution and management of daily design activities. Coordination at this level was based upon informal communication and the assumption of shared understanding.

3.3 Planning and monitoring practice

This section summarises the design planning and monitoring practice observed during the study and highlights the shortcomings of existing support technology.

3.3.1 Programme management in Rolls-Royce

Programme management in Rolls-Royce is carried out on several levels. Each manager is responsible for delivery of components or sub-systems according to specified time and budget constraints, and constructs their own plans to support this. Plans at lower levels, such as the blisk design plan, are usually more closely tied to technical activities than those at a higher level. There is no single document which describes the entire engine design process in full detail. Due to the frequent need to re-plan and the difficulty of composing a coherent picture, a fully-detailed master schedule would not be possible using current technology:

> *"We did top-down planning which didn't work... High-level plans are very unstable."* — Programme management specialist.

In common with most other companies, Rolls-Royce make extensive use of Gantt charts to support planning and monitoring. Gantt charts assume that the tasks in a process and their precedence relationships may be described in advance and that each task is only attempted once. Although there is no explicit representation of uncertainty in many Gantt-based scheduling packages, it is usually straightforward to account for changes in task duration or resourcing levels. Such changes are automatically propagated through the schedule by analysing the connectivity between tasks.

3.3.2 Limitations of Gantt charts

With respect to the blisk design process, a key limitation of the Gantt chart is its inability to represent iteration. The design process cannot easily be described as a detailed sequence of tasks suitable to construct a Gantt chart:

> *"I can't make [the Gantt chart] represent iterations. So all I've done is put loads and loads of rows in parallel."* — Stress engineer.

Gantt charts are useful despite their limited representation of process behaviour, since they may be updated when the actual process diverges from the document. However, since any Gantt chart is still an external representation of a mental model its utility is limited by the planner's perspective and overview of the process. To minimise omissions arising from limited overview, Gantt charts from previous projects are commonly used as the basis for constructing new schedules.

Gantt charts require experts' process knowledge and substantial time to produce. However, they do not capture the reasoning and assumptions behind their construction and thus communicate only the tasks which need to be completed. In practice, detailed schedules often become little more than checklists due to the expense of re-planning when they are invalidated by unexpected events. However, having an accurate plan against which progress could be compared was widely recognised as important to effective monitoring and control. This observation supports the business-oriented objective of the case study by indicating that more detailed activity planning would be beneficial to the company.

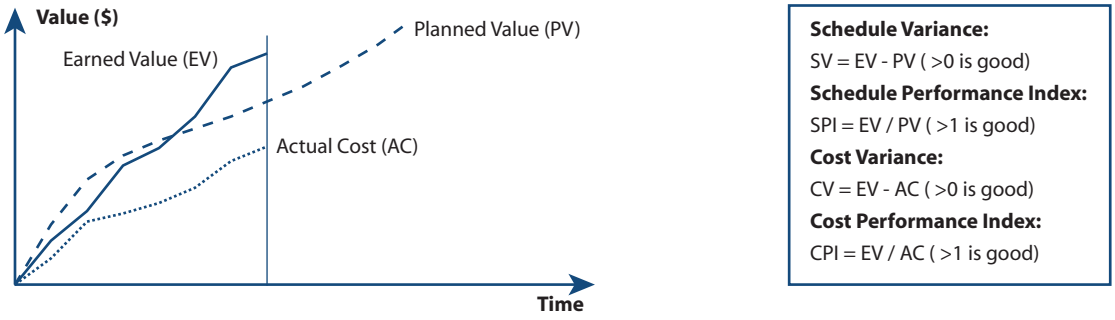


Figure 3.4 Earned value management is a technique to monitor project progress by comparing the total value of completed deliverables against the planned value and the total cost accumulated.

3.3.3 Progress monitoring and control

The F136 project used an Earned Value Management System (EVMS) to support project monitoring. This is a widespread approach whose implementation is a contractual obligation for many U.S. Department of Defense projects. The system is documented in the ANSI standard ANSI/EIA-748-A-1998. In overview, it requires a project to be decomposed into a work breakdown structure of independent deliverables, each of which is assigned a monetary value. When each deliverable is finalised its value is added to the earned value of the project. At discrete monitoring periods, the earned value to date is compared against the planned value and the cost accumulated. Simple metrics then indicate project performance regarding schedule and cost (Figure 3.4).

The EVMS can indicate when an increased rate of progress is required to meet programme milestones. However, it does not indicate *how* this could be achieved. Another limitation arises from the assumption that a project can be decomposed into discrete deliverables which cannot be invalidated by later work, and which are distributed throughout the programme such that value is earned in small increments. This assumption is inaccurate in the case of blisk design, which is a lengthy process of iterative refinement that produces few finalised deliverables prior to completion. Although useful for project-level monitoring, the EVMS therefore provides insufficient detail for managing the iterative component design process. The F136 project team addressed this limitation by conducting weekly meetings to monitor the progress of individual components. Technical risk, completed tasks, and earned value were tracked in these meetings.

3.3.4 Trade-offs between quality- and schedule-focused objectives

The need to identify trade-offs between objectives is characteristic of all design problems. In the context of programme management both quality- and schedule-focused objectives must be considered. To illustrate, consider that a minimum level of design maturity is required to satisfy any product's requirements. Once this objective is met, further design work is concerned with mitigating technical risk and improving secondary objectives by increasing efficiency, reducing weight and manufacturing cost *etc.* In practice, balancing the potential benefits of refinement against the risk of late delivery requires careful co-ordination between the engineering personnel responsible for ensuring technical quality and the management personnel responsible for timely delivery. For instance, since technical issues discovered late in the process tend to require costly iteration to address, an effective strategy to minimise technical risk is to dedicate effort to ensuring the design is as mature as possible as early as possible. However, from the scheduling perspective this early refinement work expends schedule slack, a resource that should not be expended on unneccesary improvements but which should be reserved to buffer problems which might arise later. This trade-off between improving quality and expediting delivery is considered by many personnel throughout a project. In the context of planning and scheduling, it may be viewed as the allocation of limited time and resources between tasks that contribute to each objective.

3.4 Modelling iteration in the design process

The case study revealed that iteration is a major feature of the blisk design process. Furthermore, it was found that representing iteration was a key challenge in scheduling and programme control. Identifying an approach suitable for modelling iterative behaviour in blisk design was therefore considered key to meeting the business-oriented objective introduced in Section 3.1. This section analyses iteration and its modelling in more general terms to achieve this.

Firstly, six perspectives of iteration are introduced to frame the discussion. Secondly it is argued that, although this framework is generally applicable, different perspectives tend to dominate for certain process participants, phases of the design process and complexities of the product being designed. Thirdly, key

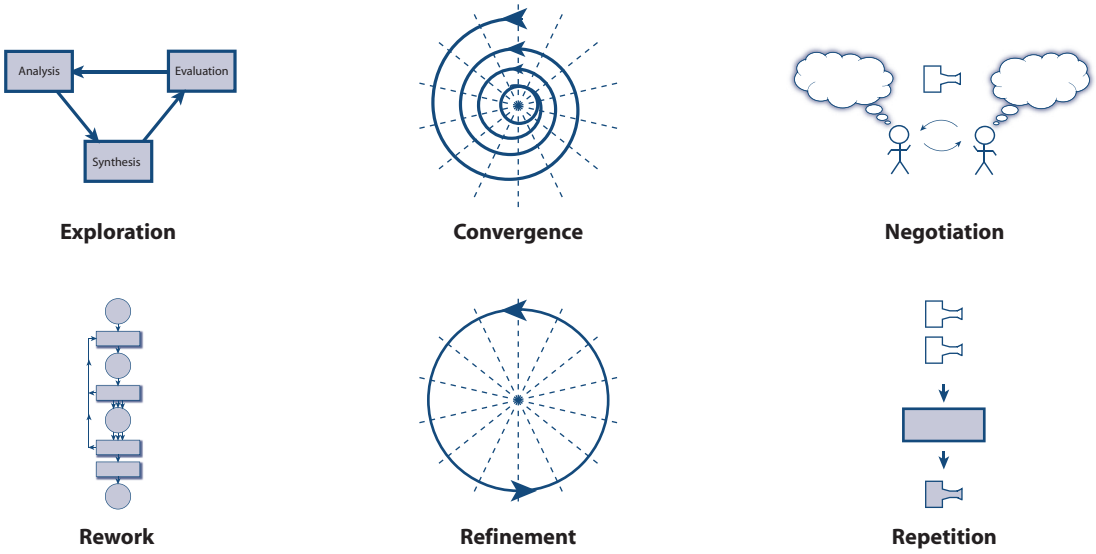


Figure 3.5 Six perspectives of design iteration.

challenges in modelling iteration are highlighted. Fourthly, it is argued that no process model can capture the full complexity of iteration, and that a key factor for selecting an appropriate modelling approach is therefore the characteristics of iteration in the domain to be modelled. This analysis allows the identification of an approach suitable for modelling the blisk design process in Section 3.5.

3.4.1 Perspectives of design iteration

Iteration forms a recurring theme across much design literature. However, it is usually discussed in the context of a particular design process model or analysis (*e.g.*, Smith and Eppinger 1997; Mihm *et al.* 2003; Yassine *et al.* 2003). Relatively few publications discuss iteration in empirical or model-independent terms (for rare examples see Browning 1998; Safoutin 2003; Costa and Sobek 2003; Costa 2004). Furthermore, although a number of classification schemes have been proposed for analysing iteration, it remains difficult to characterise due in part to the subjectivity of most definitions. For example, a distinction may be drawn between 'productive' and 'unproductive' iteration to motivate process reconfiguration to reduce the latter — but a manager's definition of productive work is likely to differ from a designer's. Despite this difficulty of classification, a framework is necessary to organise the analysis in forthcoming sections. Six non-orthogonal perspectives are therefore proposed (Figure 3.5):

- **Exploration.** In modern thinking about design, it is an almost universal view that the concurrent, iterative exploration of problem and solution spaces is fundamental to the creative process. This solution-oriented perspective of design problem solving is the subject of many publications (Section 2.3; p. 20). Exploration involves both divergence (during synthesis) and convergence (during evaluation) of the solution space.

- **Convergence.** Many technical design problems may be viewed as the selection of product parameters to meet well-defined performance objectives. Where the relationships between parameters and objectives are complex and a satisfactory solution cannot be identified by inspection or direct analysis, an iterative process is used to converge upon a solution (Evans, 1959). This convergence strategy is used in designer-driven processes as well as automated design and optimisation systems. The sequence of tasks is typically disordered, since designers react to emerging issues when determining the next focus for their attention.

- **Refinement.** Once a design meets its primary requirements, designers often continue to improve secondary characteristics such as the elegance of their solutions. Excessive refinement can occur if it is not obvious when to stop working on a problem, for example if there are few milestones in a development schedule or if evaluation criteria are subjective — as is often the case where products have fashionable or aesthetic appeal (Eckert, 2006).

- **Negotiation.** Many design problems require integration of the contributions from personnel from disparate disciplines, who often have a limited understanding of one anothers' fields. In such cases, iteration allows trade-offs between competing goals to be negotiated (Bucciarelli, 1996).

- **Rework.** Emerging problems with the product or other unexpected events may drive rework of activities which were previously considered complete. Unnecessary rework may be caused if the process is too complex to identify the most efficient configuration, *i.e.*, order of work execution (Eppinger *et al.* 1994). Rework requires activities to be re-attempted because the information on which they were based was updated. This is undesirable because effort is expended with no increase in design performance or knowledge.

- **Repetition.** Similar activities are often performed at different points in the design cycle to apply similar operations to different information. Repetition differs from *exploration*, *convergence*, *refinement*, *negotiation* and *rework* in that it involves re-visiting similar activities to achieve a different goal, rather than re-visiting a goal using potentially different methods.

3.4.2 Influences on iterative behaviour

Although the six perspectives of iteration introduced above may be identified in most design processes, the emphasis of each differs according to domain-specific factors. This section argues that certain perspectives predominate for different participants, stages of the design process and complexities of the product and process. The argument is presented to illustrate the complexity of iterative behaviour and does not form an exhaustive framework of influences.

3.4.2.1 Participant's viewpoint

Participants in a process often have different perspectives of iteration, according to their backgrounds, responsibilities and goals. For example:

- **Component engineers.** As discussed above, component design may often be viewed as the iterative application of tools for *convergence/refinement*. Due to the complexity of many components and the need to compress project schedules, they are designed by teams of experts working concurrently. *Negotiation* is therefore a key aspect of component design iteration. At any time, change could occur and require *rework* of all activities.
- **Project managers.** At a higher level, project managers usually describe processes in terms of deliverables and lead times rather than individual tasks and information flows. As a result projects are perceived as interconnected concurrent workflows which continuously exchange information, often organised around components or sub-systems. From this perspective, *rework* is the primary form of iteration and is perceived as an undesirable characteristic which increases schedule risk and lengthens cycle time. This contrasts with the component engineer's view of iteration as a fundamental and necessary aspect of designing.

- **Specialists.** Just as many components must be designed in parallel during a project, several projects are conducted concurrently by most organisations. In the aerospace industry, design often involves a small number of specialised tasks which are the responsibility of a small group of experts who address several projects concurrently. During the case study, this *repetition* was often not perceived as iteration.

3.4.2.2 Stage of the design process

Certain perspectives of iteration are characteristic of each stage of the design process. For example, *exploration* plays a key role during concept design as various alternatives are proposed and evaluated. In embodiment design, work is divided into concurrent streams and *convergence/refinement* dominates. *Negotiation* between experts plays a key role throughout this stage.[3] During detail design, similar analyses are often performed upon different components. For example, the detail design of most load-bearing components require stress analysis and, depending on the organisational structure and division of responsibilities, *repetition* may therefore occur. Finally, towards the end of a project changes are often necessitated by the discovery of issues during systems integration and testing. Additional *negotiation* and *rework* may thus be required in these stages.

3.4.2.3 Complexity of the product and process

Wynn *et al.* (2007) draw on additional case studies in a diesel engine manufacturer to argue that the importance of each perspective of iteration is affected by the complexity of the product and design process. Highly complex products such as the aero-engine are designed by large and geographically dispersed teams. Besides a dedicated conceptual design team, these organisations often have few generalists who possess an overview of the whole product. Aerospace design is therefore conducted by component and sub-system teams who must co-ordinate their work. This requires *negotiation*-driven design which necessitates *convergence* and drives *rework* when problems occur. The individual components, such as the fan blisk described in previous sections, are often sufficiently complex that

[3] As key design parameters and the interfaces between components and modules are frozen following concept design, negotiation is mostly confined within product boundaries.

they are designed by large teams whose members work concurrently and therefore engage in an inner *negotiation* strategy.

Iteration during the development of less complex products such as the diesel engine can also arise from the need to co-ordinate multiple projects within the organisation. In such companies a relatively small number of experts typically contribute to several projects, and the resulting resource limitations necessitate that many tasks begin based on early assumptions, potentially leading to *rework*. This risk is difficult to manage if it occurs at many points throughout a process. For example, relatively simple routine change processes are used to generate customised versions of diesel engines. In these cases a dedicated team deals with most change requests, but experts must be re-assigned from core development to address any specific problems which arise (Jarratt, 2004). Such bottlenecks require that personnel must either begin work on incomplete information and accept the risk of *rework* or downstream tasks must be delayed. This example illustrates one way in which complex iterative behaviour can arise from the interactions between processes that would be more straightforward in isolation.

3.4.3 Challenges in modelling iteration

This section identifies some key challenges in representing iteration within an analytical process model. The discussion thereby introduces the argument that it is impractical to comprehensively reflect iterative behaviour in such a model.

3.4.3.1 Identifying an appropriate representational basis

The discussion thus far has highlighted that alternative ways of thinking about iteration are available. Additionally, the modelling frameworks reviewed in Chapter 2 are based on different representations of iterative behaviour. However, unlike a CAD rendering of a component in which a particular feature is obscured, it can be difficult to determine whether a particular way of thinking about and modelling iteration provides an adequate representation of the process. In common with the CAD model, different perspectives are often appropriate to support reasoning about different problems. Although the perspectives may be characterised, they remain difficult to reconcile into a coherent picture. Selecting an appropriate representational basis is thus a key challenge in modelling iteration.

3.4.3.2 Identifying an appropriate level of detail

Prior to constructing a formal representation of a process a mental model must be developed. If iteration is defined as the repetition or rework of activities, its representation depends on the chosen level of description, which is a simplification of the mental model and in turn the observations of reality on which it is based. Representing task-based iteration requires both perceiving a process to be composed from discrete activities and classifying two such activities undertaken at different times as similar, although they are performed to meet specific objectives in each context and therefore must be distinct at some level. The perception of similarity between tasks was found to be highly subjective during the case study, depending upon individuals' roles and their understanding of the process. Furthermore, because most tasks are ill-defined and may be adequately described at varying levels of detail, the terminology or representation used to initiate a discussion was found to strongly influence the information elicited.

To illustrate the subjectivity of iteration, consider the following three viewpoints of the concurrent design and manufacture of a component. From the programme manager's perspective, a convergent dialogue occurs between the design and manufacturing groups. However, the team developing the component perceives a sequence of many different tasks. A researcher conducting a protocol study might look closer still. From this perspective an iterative process would emerge again, composed of many repetitions of a generic problem-solving process.

Due to this subjectivity, decisions made while constructing a process model can influence its representativeness. In terms of iteration, the fidelity of a model is influenced by the level of detail and mode of description, as well as the characteristics of the chosen modelling framework. To illustrate, *convergence/refinement* iteration that involves revisiting a number of tasks could be modelled in detail or as a single, higher-level task. However, the possibility of unplanned rework occurring within the cycle could be easily overlooked if tasks are aggregated.

3.4.3.3 Modelling the occurrence of rework

Many activities during the design process have potential to reveal rework. A dynamic model of process behaviour should include such possibilities. However, it is difficult to fully enumerate the potential failure points and modes of failure which drive rework. Furthermore, even a limited list is too large and the failure modes too complex to incorporate in a task-based model. In practice, the potential failures which are considered must be selected subjectively based upon judgement of risk or potential impact based on recent experience. Characterising the circumstances under which rework is discovered must include experts' judgements of uncertainty; this is biased by many factors (see, *e.g.*, Ayton and Pascoe 1995 for further discussion).

3.4.3.4 Modelling the consequences of rework

Rework of one task may invalidate assumptions and ultimately require many downstream tasks to be re-attempted. A number of factors contribute to challenges in modelling the consequences of rework, including:

- **Determining which tasks must be re-attempted.** The subset of tasks which are re-attempted following rework discovery is influenced by several factors, including: the decisions of personnel working on the project; the perceived criticality of the rework; and the availability of design margin to absorb the rework. If insufficient margin is available, a change to one aspect of the design is likely to propagate, forcing knock-on changes to other parameters, features, components, or sub-systems to accommodate it. Such propagations commonly follow three patterns: *ripples* of change which die away quickly; *blossoms* which are eventually brought under control; and *avalanches* which may ultimately require rework of all activities (Figure 3.6; Eckert *et al.* 2004). As a change to any feature may require many other tasks to be re-attempted, a key influence upon a project's response to rework is the state of the design when the rework is discovered. This is not incorporated in any of the process simulation models reviewed in Chapter 2, and, due to the limited information available to modellers, arguably would not be feasible to represent in most cases. In addition to this product-

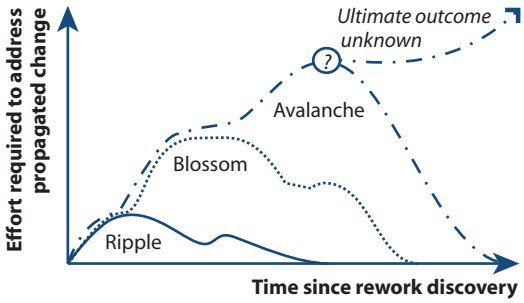


Figure 3.6 Many tasks may be re-attempted when rework is discovered. This is difficult to predict as it may propagate explicitly via task sequences and implicitly, through connected components or parameters (Adapted from Eckert *et al.*, 2004).

related complexity, performing unexpected rework typically results in delays to downstream tasks. The structure of information flows in a process can cause such delays to be absorbed, propagated or amplified in a similar fashion to design changes. This behaviour is sufficiently complex that it is difficult to understand even when a detailed model of information flows is available (Chalupnik *et al.*, 2007).

- **Determining how much effort is expended on each re-attempted task.** Each time a task is re-attempted it is likely that a different amount of effort will be expended. For example, Evans (1959) describes that during refinement the effort and resources which can be applied increase with each iteration as the solution converges. In contrast, Cho and Eppinger (2001) observed that task durations tend to reduce on subsequent iterations. In general, the time expended on a task may be influenced by a number of factors such as the fidelity of input information, the knowledge to be gained by attempting the task and the perceived likelihood of later changes necessitating rework.

- **Determining how the rework will be managed.** In practice, the delays associated with rework are often absorbed by fire-fighting and re-organising the schedule.[4] In general, the management of rework is difficult to model but forms a major influence upon project behaviour. This is elaborated in Section 3.4.3.5 below.

[4]The recent Airbus A380 project is a relatively rare example in which specific design rework was cited as the cause of major project delay.

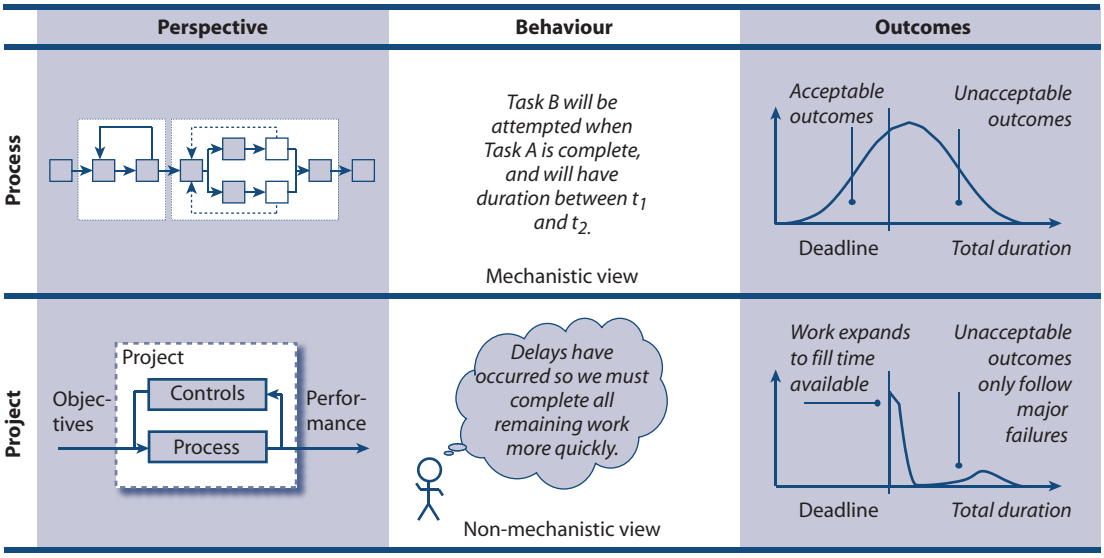


Figure 3.7 Iteration is not a mechanistic property of the design process, but is controlled by project participants (Adapted from Wynn *et al.*, 2007).

3.4.3.5 Modelling project controls

Many analytical modelling approaches are based on the mechanistic view that uncertainty about process outcomes arises from the interactions between discrete activities.[5] One limitation of these models is that they do not incorporate the active control of projects by actors who reason *in-situ* about the current state, short- and long-term goals and their predictions about future events (Figure 3.7). For example, if a test revealed that significant design changes were necessary, management decisions would usually be taken to determine whether major intervention was required. Such intervention might involve obtaining more resources or negotiating relaxed milestones; in any case, it is likely that the short-term goals of many personnel would change as they adapt their working plans to the altered project context. In major rework scenarios the process may also be re-planned such that delays do not accumulate but are reset to a new baseline. As with the product-related factors which influence iterative behaviour, the roles of actors and plans in guiding the outcome of a project are difficult to incorporate in a process simulation model.

[5]In this context, a mechanistic model represents a system as a collection of constituent elements whose interactions are governed by well-defined rules. Such models assume that uncertainty about the system's behaviour arises either from the difficulty of summation — if the elements are individually uncertain and/or were poorly understood prior to modelling — or from the complexity which emerges from interactions between elements and rules. Such a model need not be deterministic or simplistic.

3.4.4 Iteration as a key factor guiding framework selection

To recap, previous sections have identified that iteration is a defining characteristic of the design process, argued that different perspectives of iteration dominate according to domain-specific factors, and highlighted some challenges of representing iteration in analytical process models. This section argues that each of the analytical process modelling approaches reviewed in Chapter 2 is best suited to representing certain perspectives of iteration. Each approach therefore predisposes the modeller towards a certain way of thinking about the process. Although this may seem obvious, it can be easily overlooked if the modeller is inexperienced or unfamiliar with the range of modelling approaches available. Additionally, due to the lack of objective reference points, it can be difficult to determine whether an approach is appropriate if the modeller is unfamiliar with the process.

Of the task network approaches reviewed in Section 2.5, dynamic task models such as Signposting are most appropriate for modelling processes characterised by *rework* or *exploration*, since they account for the current state of the project when determining which tasks are attempted at each point (O'Donovan, 2004). However, they are less suited to modelling well-defined *convergence/refinement* processes since the repetition of an ordered sequence of tasks cannot be easily represented. In contrast, task precedence models such as IDEF3 (Mayer *et al.*, 1995) are well-suited to ordered *convergence/refinement* but cannot easily represent *rework* since the complexity of possible failure modes cannot be incorporated. The treatment of iteration in task dependency models such as the DSM (Steward, 1981) depends on the specific analysis algorithm and assumptions. In general, however, they are suited to model highly concurrent processes characterised by *convergence/refinement* and *rework*. Unlike dynamic task models they require the process network to be modelled in advance. They also provide a less intuitive notation than the task precedence models.

The other analytical modelling approaches reviewed in Chapter 2 are not primarily activity-focused and do not provide the task-level detail required for applications such as planning support. Queueing models focus on the high-level behaviour of an organisation's processes and view iteration only in aggregate terms. Multi-agent models are well-suited to represent processes involving *nego-*

tiation and *rework*, since they incorporate aspects of distributed decision-making and self-organisation. However, they are less intuitive than the task network models and could thus be difficult to verify by discussion with process participants. Finally, system dynamics models could potentially encompass the influence of project controls on iterative behaviour. However, the models of this type in the design process modelling literature do not explicitly decompose a process into individual tasks. While they can provide useful insights into project behaviour, they are thus poorly suited to provide direct support.

The predisposition of each modelling basis towards certain perspectives of process behaviour is summarised in Figure 3.8. This analysis highlights that none of the process modelling approaches reviewed in Chapter 2 is suitable to capture every aspect of iteration nor to reflect the full complexity of iterative dynamics. The discussion in this chapter could provide a starting point to develop a more comprehensive modelling approach — for instance, by extending a task network simulation to incorporate the role of project controls. However, such a model would need to incorporate additional assumptions about project behaviour and/or require additional input data. The case study revealed that the data available in practice is of insufficient fidelity to justify this approach in the context of blisk design.

Due to the complexity of iterative behaviour, it is therefore impractical to model the blisk design process with sufficient fidelity that simulation can predict the profile of project outcomes in task-level detail. Furthermore, the ubiquity of iteration indicates that this finding is applicable to a broader range of engineering design processes. Any support method based upon design process simulation should therefore account for the limited predictive utility of the underlying model. The modelling framework underlying such an approach should be carefully selected by considering the characteristics of iteration in the domain to be modelled and the suitability of each framework to represent this behaviour, in addition to any specific requirements of the modelling application.

Framework basis	Examples	Strengths and weaknesses for modelling iteration
Task network	**Task precedence**,	Good for convergence/refinement
Section 2.5.1	*e.g.*,	- If order of attempting tasks is well-defined
	IDEF3 (Mayer *et al.*,1995)	- Intuitive graphical notations ease interpretation
	PERT/GERT (PMI, 2004)	*Poor for negotiation*
	Petri net (Peterson, 1981)	*- Cumbersome to model concurrent activities exchanging information*
	ProModeller (Vajna, 2005)	*Poor for rework*
		- Many failure points/modes complicate network
		- Cannot easily represent revisiting different tasks on each iteration
		Poor for exploration
		- Cannot represent in-situ task selection
		Can be difficult to manipulate/restructure models
	Task dependency,	As above, except:
	e.g.,	Good for negotiation
	IDEF0 (NIST, 1993)	- Explicit model of workflow not required
	Activity DSM (Eppinger, 1991)	- Heavily concurrent processes easily represented
	DMM (Danilovic and Browning, 2007)	*Can be difficult for non-experts to interpret models*
	Dynamic task,	Good for exploration
	e.g.,	- Implicitly incorporates in-situ task selection
	Signposting (Clarkson and Hamilton 2000)	Good for negotiation
		- Explicit model of workflow not required
	MIDAS (Chung *et al.* 2002)	- Heavily concurrent processes easily represented
	ATP (Levardy *et al.* 2004)	Good for rework
		- Implicitly incorporates revisiting different tasks
		Poor for convergence/refinement
		- Difficult to represent well-defined repetition and sequencing
		Difficult to visualise and hence to interpret and validate models
		Difficult to validate assumptions underlying task selection rules
Queueing	*e.g.*,	Focus on high-level process behaviour
Section 2.5.2	Adler *et al.* (1995)	Iteration is modelled in the same way across projects
	Q-GERT (Pritsker 1979)	
	Narahari *et al.* (1999)	
Multi-agent	*e.g.*,	Good for negotiation and rework
Section 2.5.3	Olsen *et al.* (2006)	- Represents distributed decision-making and self-organisation
	Mihm *et al.* (2003)	*Modelling is complex and may require programming expertise*
	Levitt *et al.* (1999)	*Difficult to understand assumptions and their consequences*
System dynamics	*e.g.*,	Represent feedback which governs process behaviour
Section 2.5.4	Cooper (1993)	*Abstract approach may be unintuitive to non-experts*
	Ford and Sterman (1998)	*Simplifying assumptions do not represent details of individual tasks*
		Cannot offer advice regarding task-level improvements

Figure 3.8 Each of the process modelling approaches reviewed in Chapter 2 is better suited to representing certain perspectives of design iteration.

3.5 A basis for process modelling to support project management in blisk design

To recap, the business-oriented objective introduced in Section 3.1.2.3 (p. 65) required the development of a model-based approach to support planning and re-planning of activities in the component design process, with a particular focus on the mechanical design of fan blisks. Observations during the case study revealed three framework requirements for a process modelling approach on which this support could be based:

- **FR1** Decompose the design process into individual tasks to support more detailed scheduling. Existing documentation indicated this could be achieved by describing the process in terms of the computer tools used to design the blisk (Section 3.2.4; p. 70).
- **FR2** Represent the iterations which are undertaken to meet the mechanical design objectives. These are characterised by the *repetition* of certain task sequences governed by the well-defined interactions between computer tools, as well as by disordered *convergence/refinement* which is governed by *in-situ* design decisions (Section 3.2.2; p. 68).
- **FR3** Provide an intuitive notation to allow process participants who have limited overview to engage with the approach (Section 3.2.3; p. 69).

The analysis of design iteration in Section 3.4 allowed each analytical modelling approach to be reviewed in terms of these framework requirements. Firstly, only task network models can meet Requirement FR1, since other approaches do not provide the task-level detail required for scheduling. Secondly, Requirement FR2 does not further distinguish between the types of task network model available, since each is best suited to modelling either ordered- or disordered task sequences. Finally, consideration of Requirement FR3 indicates that a graphical task precedence representation would provide the most appropriate basis for modelling blisk design, since the graphical notation of such approaches is similar to the existing process documentation. The development of such a model and its application to meet the business-oriented objective is discussed in Chapter 4.

3.6 Summary

The objective of this chapter was to explore the utility of analytical process models to support industry practice. This was achieved by conducting an empirical study of aero-engine component design, complemented by a more general analysis of modelling iteration in the design process. To summarise:

- The design processes for components such as the fan blisk are similar across product generations and can be modelled. However, although all personnel were expert in their area, no individual could provide a description of the entire design process in sufficient detail for modelling. This arose in large part from the difficulty of describing design iterations.
- Iteration is ubiquitous in design and critical in determining the dynamic behaviour of design processes. Although this is widely recognised, relatively few publications discuss iteration in empirical or model-independent terms. This chapter contributes to the discussion by proposing six perspectives of iteration and arguing that no single modelling framework can capture the full complexity of process behaviour. It was therefore proposed that iteration should be considered a key factor in identifying the most appropriate analytical framework for modelling a given process for a given objective.
- The analysis of iteration was used to identify the task precedence modelling approach as the most appropriate basis for modelling the design of fan blisks to support project management. This forms the basis of the modelling and simulation framework which is developed in forthcoming chapters.

In addition to these specific contributions, the extended period of industry interaction formed a significant influence on the direction of the research project. In particular, the case study led to insights regarding the design process and its behaviour; the stakeholders in process modelling approaches and their requirements; and the practical challenges faced in introducing new approaches to an industry environment. These insights focused the subsequent research upon issues pertinent to practical application.

—— This page is intentionally blank ——

Chapter 4

Support for project management

This chapter discusses the context in which the research questions were addressed. A model-based approach to support project management in aerospace component design was developed during the case study described in Chapter 3. This research allowed a prototype modelling approach and software tool to be proposed and evaluated. These prototypes formed the basis for the more sophisticated approaches discussed in Chapters 5 and 6.

Discussion proceeds in six sections. Firstly, the research objectives and methods are introduced. Secondly, other approaches which address similar requirements are briefly reviewed. Thirdly, the support method is discussed in detail. Fourthly, the process modelling framework and software tool developed to implement the approach are discussed, together with the construction of a detailed model of the blisk design process to illustrate its feasibility. Fifthly, the research objectives are revisited and opportunities for further research to enhance the support method are highlighted. Finally, the contributions of the chapter to the thesis are summarised.

4.1 Overview

This section introduces the chapter by discussing the research objectives and methods used to develop the management support approach.

4.1.1 Research objectives

This chapter begins to address the main research questions outlined in Chapter 1. The context for exploring these questions was provided by the business-oriented research objective stated in Section 3.1.2.3 (p. 65):

> Develop a model-based approach to support the planning and re-planning of activities in the component design process, thereby improving progress monitoring and programme control.

This objective is addressed in the forthcoming sections. Four support requirements for the method arose from the research reported in Chapter 3:

- **SR1** Synthesise a coherent perspective of activities and their interactions from the process participants' localised understanding (Sections 3.2.3 and 3.2.4, pp. 69–72).
- **SR2** Identify a schedule which provides an appropriate trade-off between objectives (Section 3.3.4, p. 75).
- **SR3** Support monitoring and re-planning using metrics which are not expressed in terms of independent deliverables (Section 3.3.3, p. 74).
- **SR4** Function effectively given uncertainty *about* the process. In other words, application of the methodology should require neither an exhaustive understanding nor a fully comprehensive model of the process. These are unlikely to be available in practice (Section 3.4.3, p. 80).

These requirements were further qualified by two pragmatic considerations. Firstly, the design process was viewed not only as the subject of the support method but also as the context into which it would ultimately be delivered. It was thus considered important that direct benefit was provided to the technical design teams who would support the modelling activity. This led to the following requirement:

- **SR1.1** Engage with technical design teams by basing the method on a graphical modelling formalism allowing improvement and integration of the existing process documentation.

Secondly, to obtain buy-in from company personnel it was necessary that the importance of the problem and the proposed solution can be quickly outlined. It should thus be possible to engage with the approach at different levels of resolution; a limited application to provide modest benefit could then be expanded to provide a more sophisticated solution if necessary. This led to the following qualification to Requirement SR4:

- **SR4.1** Complement existing tools and practice by reducing the cost and frequency of re-planning rather than eliminating it entirely.

4.1.2 Research methods

A method to address these requirements was developed during an eight-month on-site case study at Rolls-Royce Bristol (Chapter 3). Throughout this time, the author observed meetings, conducted interviews with management and engineering personnel and developed a detailed process model. The unusual length and ethnographic elements of the case study allowed domain experts to strongly influence the direction of research; the issues addressed were considered relevant and important by both engineers and managers. Weekly meetings were held between the author and the study sponsor to ensure the practical feasibility of the emerging approach. Following conclusion of the study, the method was further refined through laboratory research and discussions with other industry personnel.

From the outset it was acknowledged that the long lead-time of aero-engine development projects would not present an opportunity to conduct an industry evaluation within the timescale of this project. Although the method addresses requirements identified through an extended case study, it is thus presented as a primarily theoretical contribution.

4.2 Other approaches

A number of model-based approaches to support design project management may be found in the literature. Most focus on developing improved process configurations or optimised plans using structural or simulation analysis of a task network model. For example, Cho and Eppinger (2001) propose a framework for project planning based on DSM simulation. Browning and Eppinger (2002b) discuss application of a similar simulation to evaluate alternative process architectures. Melo (2002) and O'Donovan (2004) develop plans which offer guidance from any reachable process state. These approaches are theoretical in nature. They require high model fidelity prior to planning and thus do not satisfy Requirement SR4. Most do not provide monitoring support as stipulated by Requirement SR3.

Few publications address the management of design plans throughout their lifecycles, *i.e.*, by supporting the three stages of planning, monitoring and re-planning. For one example, Lévárdy and Browning propose that activity plans may be modified in response to the emerging state of the project to reach cost, schedule and technical performance targets (Lévárdy and Browning, 2005). Most such publications focus on the underlying process model and not the pragmatic aspects of its application. A small number of commercial tools have arisen from other academic research in this area, including ADePT Management Ltd's *PlanWeaver* (Austin *et al.* 1999), Acsian Ltd's *Plexus Modeller* (Acsian, 2007) and ProNavigate GmBH's *ProNavigator* (Friesleben and Vajna, 2002). These approaches focus on planning from the high-level product *development* perspective and do not address the difficulties of modelling, planning, monitoring and controlling design iterations at the component level. Furthermore, they aim to support specific management activities and either do not provide direct benefit to the design teams who must support modelling, or are not flexible enough to meet Requirement SR1.1 above. Nevertheless, the commercialisation of these approaches highlights the importance of this research area and the potential benefits to industry.

To summarise, although a number of approaches to support design project management exist in the literature, none of these publications addresses the specific requirements outlined in Section 4.1.1 above. This motivated the development of the support method described in forthcoming sections.

4.3 Model-based support for project management

This section introduces the support method under seven headings which reflect its main steps: *Model the design process*; *Simulate the process*; *Select acceptable outcomes*; *Resolve any conflicts*; *Identify a schedule of work*; *Monitor progress*; and *Re-plan*. These steps are illustrated in Figure 4.1 overleaf.

4.3.1 Model the design process

The engineers who carry out the design work construct a process model to represent tasks, iterations and key deliverables. Modelling is based on a formal task network model which allows stochastic simulation. The modelling framework is not critical to the approach; however, as outlined above it should provide an intuitive and accessible notation for ease of application. Development of such a framework is discussed in Section 4.4.

The model should decompose the process to a level of detail suitable for progress monitoring. In the case of blisk design, tasks should represent several hours' to several days' work. The model is parameterised by specifying the expected duration of each task, the numbers of *convergence/refinement* iterations which are planned and any resource limitations which might cause bottlenecks. Primary sources of uncertainty are identified and their characteristics estimated. For example, tasks' durations may be specified using probability density functions. Convergence/refinement cycles may be parameterised with a probability distribution characterising the number of iterations required, and design evaluation tasks with the estimated likelihood of each failure mode occurring.

The model also includes the metrics used to determine process performance and those used to monitor project progress. All such metrics must be expressed in terms of contributions from individual tasks in the process model. Browning *et al.* argue that many useful measures may be linked to activities in this way (Browning *et al.*, 2002a). For example, commonly-used metrics include the total cost expended and value of schedule milestones delivered, both of which must meet or exceed specified profiles of accumulation. These metrics may be respectively modelled as increasing when cost-accumulating tasks are completed or design re-

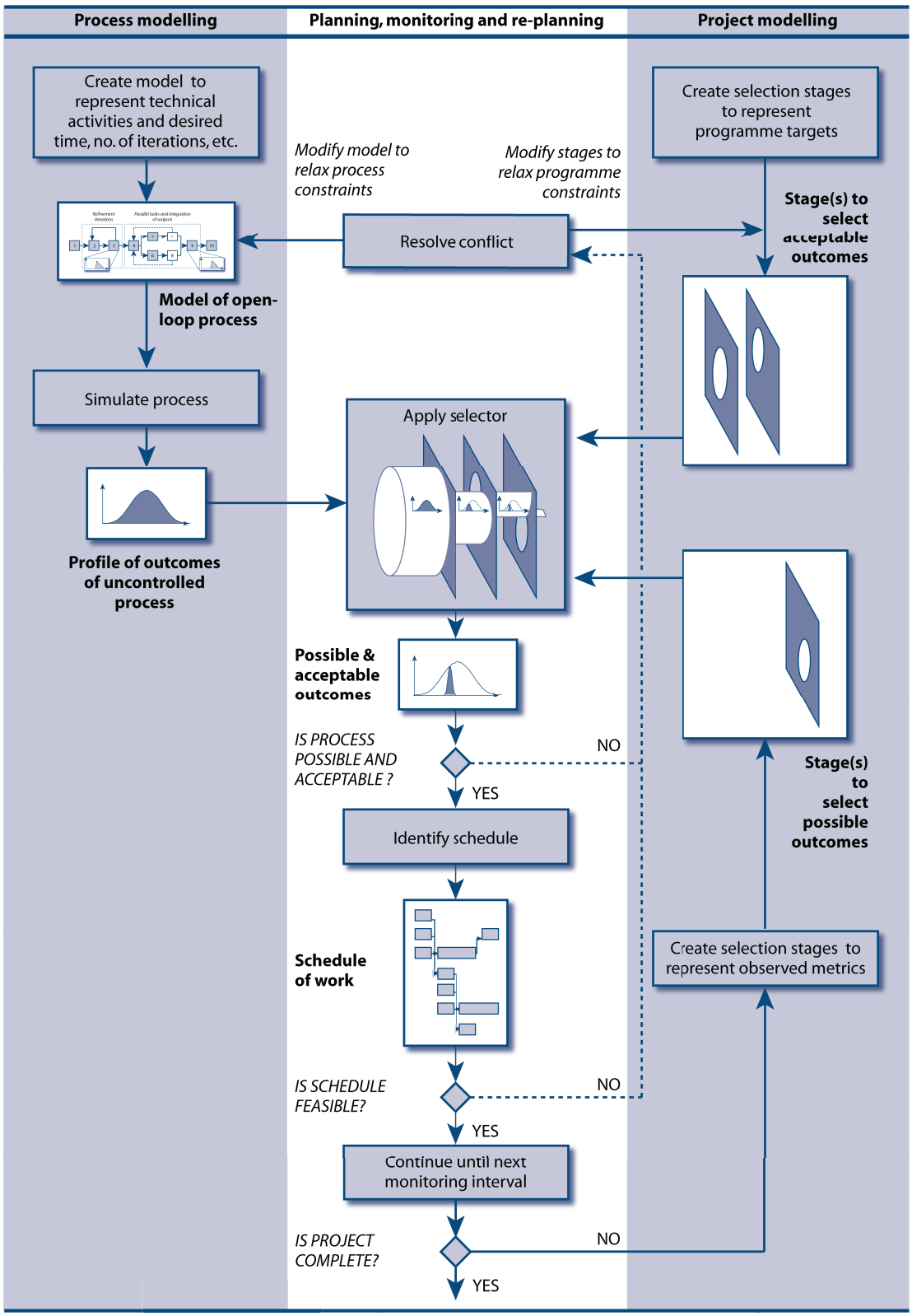


Figure 4.1 Steps in the model-based approach to support project management.

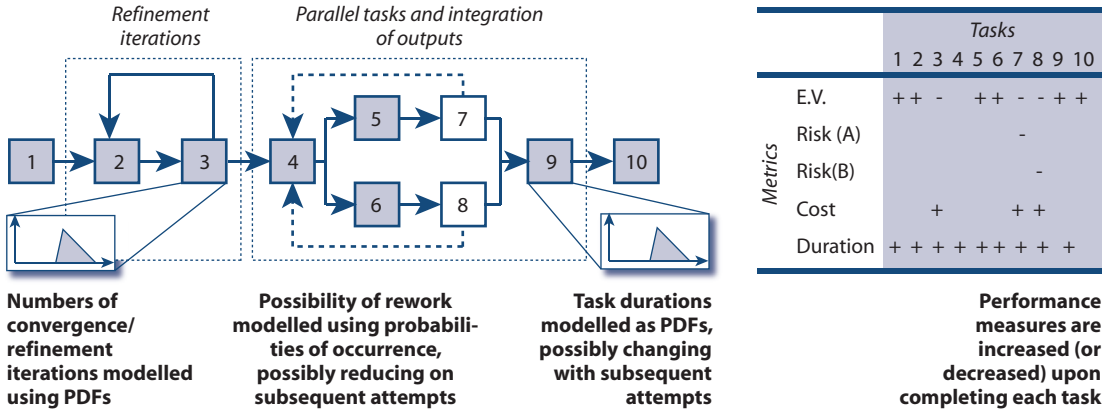


Figure 4.2 The approach is based on a simulation model which captures design tasks, uncertainties and the metrics used to measure progress and determine success.

leases are made.[1] In the latter case, the earned value should be removed if those activities are later invalidated by the discovery of rework. Another metric observed in Chapter 3 is technical risk, which Lévárdy *et al.* argue can be modelled as reducing when testing activities are performed (Lévárdy *et al.*, 2004). Figure 4.2 illustrates this concept by tabulating metrics against the tasks in a simulation model from which they are derived.

Models constructed in this way include many uncertain events distributed throughout the schedule. They may incorporate non-linearities due to: queueing behaviour resulting from resource bottlenecks and concurrent tasks; test failures, which may cause rework to propagate through many tasks in the schedule in a manner dependent upon the tasks attempted prior to the failure; and any non-linearities which are explicitly incorporated, such as learning effects. Inspection of models constructed during the research project indicated that although they may be difficult to understand due to the very large numbers of tasks and sources of uncertainty, they are in practice structured such that combinations of the uncertain events do not cause significant unexpected consequences. In other words, the variability in simulation outcome is derived from the accumulation of many small uncertainties rather than complex trajectories caused by non-linear dynamics.

[1]Modelling cost accumulation can require many contributing factors to be incorporated. For example, projects must often purchase resource which must be paid for at a fixed rate, regardless of the efficiency of its use. However, additional overtime may be purchased if required. Cost also accumulates as hardware is manufactured, testing is performed and invoices are paid. For simplicity it may be assumed that cost accumulates at a constant rate representing human resource costs, and is further increased when 'additional cost' tasks such as rig tests are completed.

4.3.2 Simulate the process

Applying discrete-event Monte-Carlo simulation to the stochastic process model reveals a profile of possible outcomes. Each outcome represents the effect of a particular sequence of task-level events upon the project-level metrics. The profile of outcomes represents the open-loop response of the project, since the uncertainties associated with individual tasks are modelled. In other words, it is assumed that participants do not respond to counteract accumulating delays or other problems which emerge during the process. As it does not capture this active guidance towards acceptable outcomes, the model may exhibit greater variability than the project it is used to support.

As discussed in Chapter 3, any process simulation can only represent a subset of the possible outcomes due to the large number of events affecting the process and the requirement for parsimony. For instance, the model may only incorporate the possibility of failure modes which are considered most likely to occur or which cannot be easily absorbed by the project. In the blisk design process, it is not possible to incorporate the decisions determining which sub-process is attempted in which order and at what level of resolution, as these decisions are made *in-situ* by considering the current state of the design and process (Chapter 3). Even if the reasoning underlying these decisions could be determined with sufficient accuracy for modelling, the information on which such a calculation might be based cannot be cost-effectively incorporated in the task-based model. This type of iteration must therefore be unrolled into a *planned* process involving distinct, increasingly detailed (and possibly concurrent) applications of the revisited sub-processes.

As any model-based approach can only consider possibilities which are represented in the model, in the event that others occur any guidance will be invalidated. For example, the correctness of a model incorporating unrolled iterations is contingent upon the planned sequence occurring in practice. The simulation model is therefore a form of plan. In common with other plans it should be optimistic in outlook, and it is expected that re-planning will be necessary in response to major adverse events which occur.

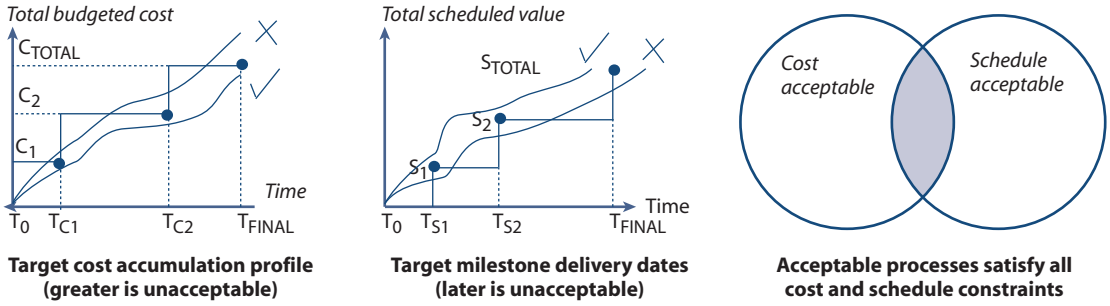


Figure 4.3 Project constraints are codified as a set of selection stages. The selector identifies the set of acceptable processes, *i.e.*, the intersection between the sets which satisfy individual constraints.

4.3.3 Select acceptable outcomes

To create a plan of work the set of acceptable processes must be identified from the profile revealed through simulation. The definition of acceptability typically encompasses multiple criteria. For example, as discussed above it is usually necessary to meet a cost accumulation profile, deliver intermediate project milestones and complete all work by a certain date. These success criteria must be codified as constraints on the metrics derived through simulation. In this case, meeting the cost accumulation criteria would be modelled using multiple constraints, each requiring the *total cost* metric to be below a certain threshold on a certain date. Milestone constraints would be incorporated by requiring certain tasks to be completed in advance of the milestone dates.

A *process selector* is used to identify the set of processes which meet or exceed all acceptability criteria.[2] A selector consists of a number of *stages*, each of which represents a filter that passes only those processes which satisfy a single constraint. The subset of processes which meet all criteria may be identified by applying each selection stage in turn; this identifies the intersection between the sets of outcomes which satisfy individual constraints (Figure 4.3). Configuring and applying the selector does not require an explicit understanding of the model or the circumstances under which criteria are met. This is a critical point; the *simulation-and-selection* methodology introduced here provides an intuitive approach to identify acceptable means of reaching the desired goals, even where models contain too many influencing factors or are too complex for inspection.

[2]The process selector is based on the engineering materials selector discussed by Ashby *et al.* (2004).

4.3.4 Resolve conflicts

It is often the case that no acceptable processes can be immediately identified using the selector. This occurs when one or more constraint-satisfying sets are empty or do not intersect, and indicates a conflict between individual project targets or between one or more targets and the plan of work. For example, the sum of desired durations for each task might exceed the permissible total, given the resource constraints and possibilities for concurrency which arise from the project structure.[3]

Prior to generating a schedule any conflicts must be resolved by relaxing the constraints of the simulation model and/or the selection stages which represent programme targets. In the above example, this could be achieved by some combination of: reducing the planned duration of individual tasks; reducing the number of planned *convergence/refinement* cycles; increasing resourcing levels; or removing non-critical tasks altogether.[4] Since models are not complex it is assumed that an appropriate resolution may be identified by inspection.

4.3.5 Identify a schedule of work

The selector may reveal a number of possible processes, representing a sample from the set which satisfies the stage constraints (Figure 4.3). A single process is chosen from this set, rendered as a Gantt chart and used as the plan of work. This provides a decomposition of the process into individual task durations which, if achieved, would satisfy the project targets. The approach therefore converts high-level targets into a detailed, incremental checklist of sub-goals described in terms of the engineering activities. The schedule may then be used to monitor progress and support design teams in guiding the project towards a satisfactory outcome.

[3]Conflicts may be incorrectly identified if acceptable outcomes exist but were not identified by the Monte-Carlo sampling approach. It is therefore important to perform enough simulation runs to adequately sample the model.

[4]Modifying a plan to reduce project duration is known as *crashing* in the project management literature. Identifying suitable tasks for crashing may be difficult in practice. For example, no personnel encountered during the case study believed they performed completely unnecessary activities. Risk mitigation activities are the primary candidates for removal, but this might be expected to increase the likelihood of problems emerging later. If this did occur the delayed discovery of rework would require more tasks to be re-attempted. In general, crashing may therefore be viewed as a trade-off between reducing the expected duration and increasing the range of outcomes, *i.e.*, the risk and potential magnitude of over-run.

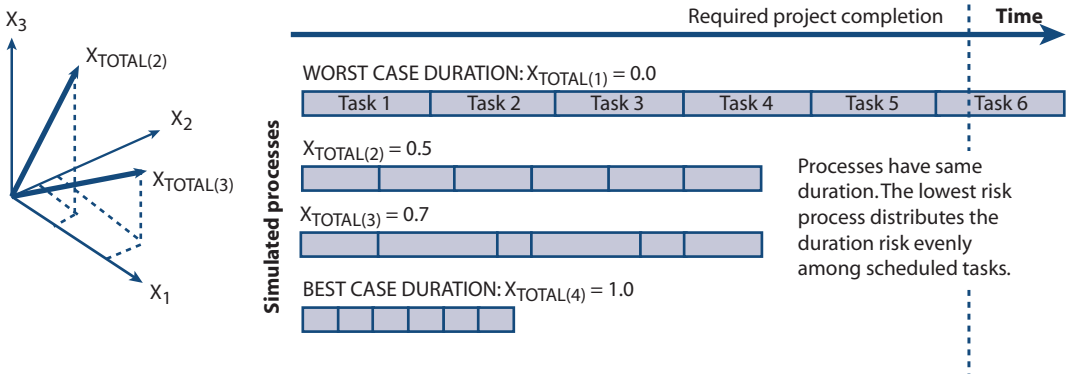


Figure 4.4 A single schedule is selected to minimise the sum of square duration risks associated with scheduled tasks, thereby evenly distributing the aggregate duration risk and minimising schedule volatility.

A schedule is selected from the set of acceptable outcomes by distributing the total scheduled duration risk amongst its constituent tasks. This heuristic aims to minimise the volatility of the schedule (*i.e.*, the likelihood that re-planning will be required) by identifying the single simulation outcome for which the sum of squared task duration risks χ^2_{TOTAL} is minimised (Figure 4.4):

$$\chi^2_{TOTAL} = \sum_{n=0}^{N} \chi^2_n \tag{4.1}$$

In this context, the n^{th} task's duration risk χ_n is defined as the likelihood that the task's actual duration will exceed the scheduled duration T, which may be calculated from the task's duration probability density function $f(t)$:

$$\chi_n = 1 - \int_{t=0}^{T} f(t).dt \tag{4.2}$$

Although more sophisticated approaches could be used to identify the optimal schedule, such as weighting the duration risk of a task by the perceived likelihood of reduced duration leading to technical problems, the approach outlined here is sufficient to illustrate the support method.

4.3.6 Monitor progress

For simplicity it is assumed that schedules may provide continuous guidance but that progress is monitored at discrete intervals. This reflects the practice of regular review meetings observed in Chapter 3. At the outset of the project,

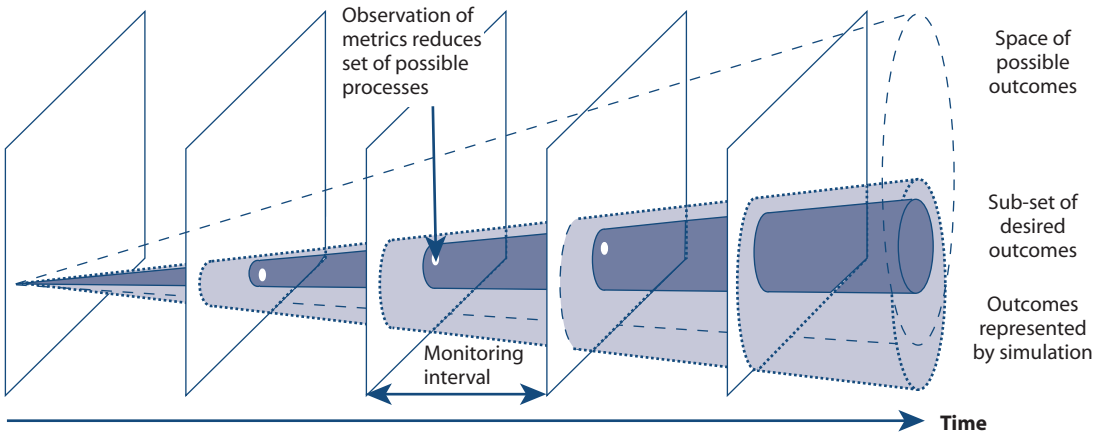


Figure 4.5 Creating selection stages to represent the observed performance measures at each monitoring period allows the uncertain state to be projected to indicate the range of outcomes.

the selected schedule provides a detailed Gantt chart indicating which activities should be complete by the end of the first monitoring period. However, it is both an imperfect and non-contingent representation. It can therefore be expected that reality will quickly deviate from the detail of this plan, and that when the first monitoring interval is complete the schedule for future intervals will no longer be valid. When this occurs, a revised schedule may be generated by constructing additional selection stages which represent the estimated state of the project at the current time. These stages must select those processes which are equally- or less complete than the estimated state at the current time. After re-applying the selector, the remaining subset contains processes which are both acceptable and still attainable following any delays which have occurred. The size of this set reflects the risk of not meeting all acceptability constraints, given the planned activities and the current state of the project.

Progress monitoring should be based on indicators which may be reliably and regularly estimated and which have a well-defined leading relationship to delivery, *i.e.*, can be used to identify issues in time to take remedial action. In practice, however, estimates are based on a basket of observations whose exact values and relationship to project completion are difficult to quantify. Monitoring is therefore difficult because managers cannot accurately determine the current state of completion.

In the following two sections, progress monitoring in the simulation-and-selection approach is discussed in terms of task completion and in terms of other metrics.

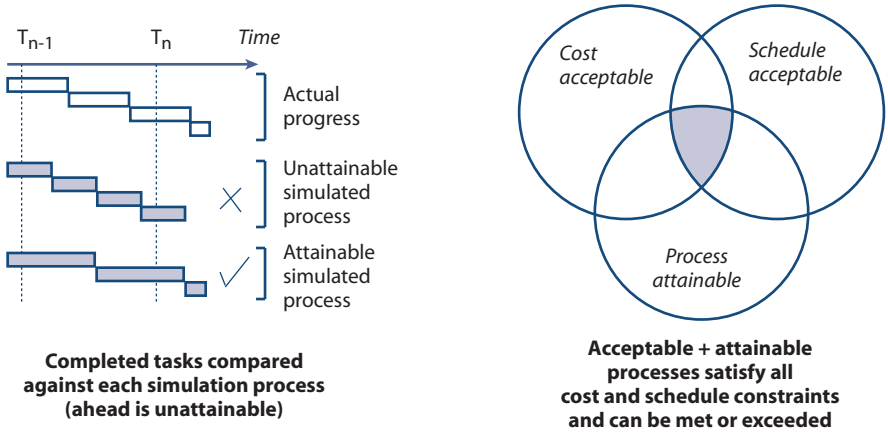


Figure 4.6 Attainable processes are still possible to meet or exceed, given the observed status of the project.

4.3.6.1 Identifying the current state by monitoring tasks

In general, tasks which have been worked upon may be classified into four categories: tasks which have been completed; tasks which are in progress, but once completed will not be re-attempted; tasks which have been completed but which may be re-attempted in future; and tasks which are in progress and which may be re-attempted following completion. Accurately categorising tasks in this way is usually not possible due to the uncertainty surrounding design iterations. This can lead to concealed rework and optimistic progress estimates.

Such categorisation is not necessary in the methodology proposed here, since the schedule may be updated by incorporating a selection stage which selects only those processes where *other tasks have not been completed* at the current date (Figure 4.6).

4.3.6.2 Identifying the current state by monitoring other metrics

Task-based monitoring requires the simulation model to accurately reflect the order of attempting tasks such that they may be 'ticked off' as they are performed. This level of fidelity is often not available in practice; for example, if emerging risks are mitigated by performing additional analyses. In this situation, it may still be useful to support the revision of a template activity plan while acknowledging that this template is not fully representative of the design process. Progress monitoring in such cases must be based on metrics which are not expressed in terms of tasks completed.

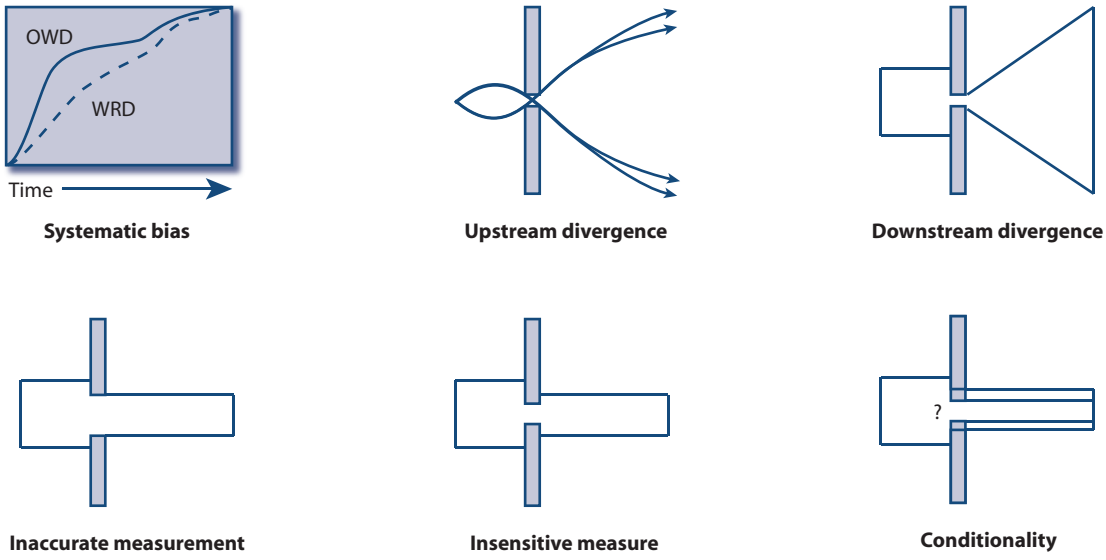


Figure 4.7 Six limitations of metrics used to track project progress.

Examples of such metrics include aggregate performance measures such as earned value and work completed to date, as well as more subjective measures such as experts' confidence in design completeness, and the inversely-related measure of technical risk; however, these can be difficult to define and quantify in practice (Hamilton, 1998; Flanagan, 2006). Such measures typically offer lower fidelity than task-based monitoring and therefore provide a less certain picture of progress. This arises from a number of inter-related factors (Figure 4.7):

- **Systematic bias.** Metrics may have systematic bias with respect to the process state. For example, Cooper (1993) used a system dynamics model to illustrate that the apparent completeness of a development project tends to lag behind the actual completeness due to undiscovered rework (Section 2.5.4; p. 58). Bias is misleading but can be taken into account once recognised. In this example, project managers quickly learn that delivery estimates are usually optimistic.

- **Upstream divergence.** A basket of variables may be insufficient to fully determine the process trajectory. For example, consider a process in which an early design decision determines the result of a subsequent test whose failure would require rework of all intermediate activities. Although success or failure is predetermined, if no metric is correlated to the unobserved event it cannot be determined until the test is performed. The current

state of a process may therefore be uncertain even if all metrics are known with absolute accuracy. A corollary is that uncertainty in outcome may be amplified in cases where trajectories which diverge due to upstream events cannot be distinguished at the point of measurement.

- **Downstream divergence.** As time progresses the outcome of a stochastic process becomes more predictable as uncertain events are encountered and their outcomes become known. In contrast, the accuracy of a predictive model tends to decrease as the horizon is extended, because uncertainties in individual events accumulate.[5] Any metric is therefore of limited utility to predict the outcome of a stochastic process due to the divergence of outcomes caused by uncertainty in future events. Where many events are subject to uncorrelated uncertainties, it is usually not possible to make early observations which significantly reduce the *perceived* variability in outcome.

- **Inaccurate measurement.** The utility of an indicator depends on the accuracy to which it may be measured. For example, the F136 project discussed in Chapter 3 used a three-point scale to track the perceived risk of delayed delivery for each engine component — even if risk was directly correlated to the ultimate project performance, measurement error would limit the utility of this metric. Causes of measurement error may include subjectivity (*e.g.*, in risk estimates), time delays prior to observations becoming available (*e.g.*, personnel typically enter their hours into the ERP system at uncertain intervals, thereby reducing the accuracy of "actual work completed" metrics) and actors "gaming the metrics" for local advantage.

- **Insensitive measure.** An increase in project completeness is not always reflected by an increase in the observed value of a metric. For example, many testing tasks have potential to reveal additional work; reduce confidence in the design; and require budget increases. Work completed, confidence and budget expended are therefore non-monotonic metrics for which a single observed value implies one of many project states.

[5] Although uncorrelated uncertainties tend to accumulate in a favourably non-linear manner. For example, the total duration PDF for a sequence of tasks with uncorrelated durations is defined by the convolution integral of the individual tasks' duration PDFs. Although the total range in this example equals the sum of the individual ranges, the outlying values are very unlikely to occur.

- **Conditionality.** The utility of a metric may vary throughout the process and may be conditional upon uncertain events. For example, measuring aerodynamics confidence may be a useful indicator early in the blisk design process. However, because many aerodynamic properties are developed during early concept design it may be less useful in later stages.

In general, a metric may suffer from several of these limitations. Nevertheless, any metric incorporated in the simulation model as discussed in Section 4.3.1 may be used to monitor progress. This is achieved by creating additional selection stages to identify those processes which the observations do not exclude. For example, to estimate the current state given the observation that 'technical risk' is between 40% and 60%, a stage would be created to select those processes for which the metric is within the specified range at the current date.

4.3.7 Re-plan

In the context of this approach, re-planning refers to the modification of the simulation model or selection stages representing acceptability constraints. Re-planning is necessary when the project diverges from the plan sufficiently that no acceptable and attainable processes can be identified by the selector. This may follow an accumulation of delays, as illustrated in Figure 4.8, or may be caused by adverse events whose possibility was not incorporated in the simulation model. Re-planning may also be necessary if activities were overlooked during the planning stage. Irrespective of their origin, conflicts must be resolved as outlined in Section 4.3.4 above.

4.4 Developing a process modelling approach

To evaluate the feasibility of the task-based simulation required by the support method, a modelling approach was developed and used to model the blisk design process described in Section 3.2.2 (p. 68). The task precedence representation was identified as suitable for modelling this process (Section 3.5; p. 88) and therefore forms the basis of the approach. Forthcoming sections describe the approach development during the case study. A detailed technical description is provided in subsequent chapters.

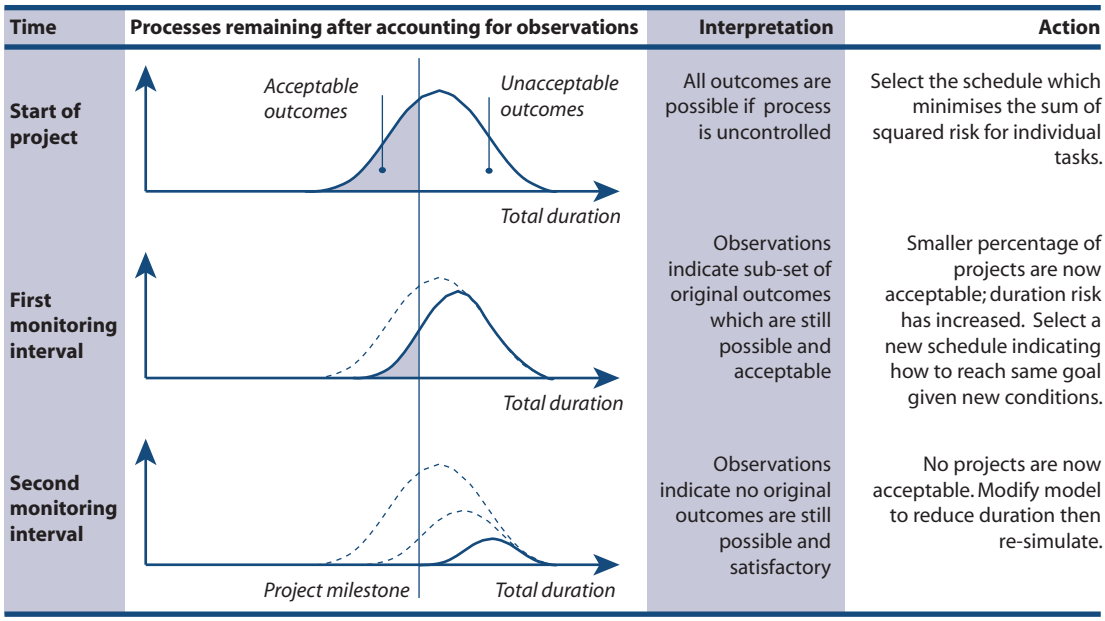


Figure 4.8 Observations at discrete monitoring intervals are used to identify current progress. This one-dimensional example assumes total duration is the only success criterion. In practice, several performance measures are typically required, such as milestones and cost. In such cases the simulation result is characterised by a joint PDF.

The modelling framework was developed in parallel with the support method and with the blisk design process model. This allowed refinement of the approach as its application was explored. Experts' process knowledge was elicited through many informal discussions with technical and management personnel, study of existing documentation and a small number of group workshops. Modelling was conducted using a prototype software tool developed by the author.

4.4.1 Prototype software

Excel 2000 VBA was selected as the implementation language for the prototype software, since this would allow the application to be used on the company network. Although Excel macros are usually associated with spreadsheet calculations, *Excel 2000 VBA* is an object-oriented language which allowed relatively robust and user-friendly software to be developed. Screenshots from the modelling application are shown in Figure 4.9 overleaf. Further detail is provided in Appendix B.

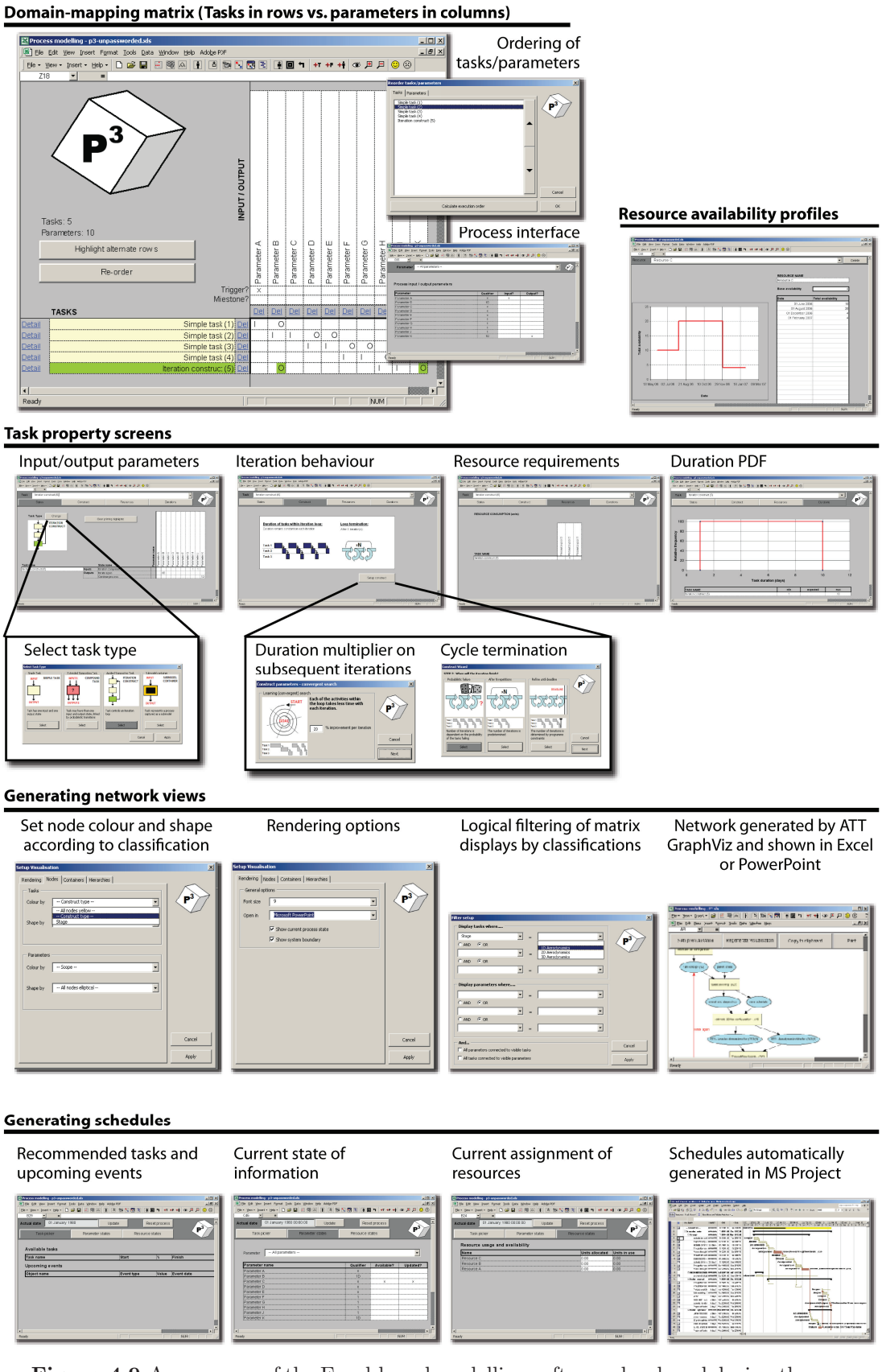


Figure 4.9 A summary of the Excel-based modelling software developed during the case study. Further detail is provided in Appendix B.

4.4.2 Task-parameter mapping matrix

The prototype software was based on a domain-mapping matrix which mapped design tasks to the parameters which they required and produced. Each task was represented as a set of rows in the matrix, with the first row representing the task inputs and subsequent rows representing each of the task's alternative output scenarios. For example, a design evaluation task would have two output scenarios to represent success or failure while a data conversion task would have only one. Parameters were represented as matrix column headings. An entry in a matrix cell indicated that the task's scenario, in the row, included the requirement for or modification of the parameter in the column. This matrix was initially populated by examining process documentation collected during the study (see Figure 3.3; p. 71 for examples). The model was then refined through discussion with the engineers who conducted the process.

Many of these refinement exercises involved sketching process fragments in a flowchart format. Since this indicated that the flowchart was a more intuitive representation for knowledge elicitation, the software was subsequently extended to generate node-link diagrams based upon information captured in the matrix. This revealed that the visualisation used during knowledge elicitation and modelling can strongly influence the form of the resulting model. The initial approach of eliciting knowledge and constructing a model using the task/parameter matrix resulted in a very strongly connected model. However, once presented with a flowchart visualisation, engineers described their process knowledge in a sparsely-connected form which resulted in a visually appealing diagram.

4.4.3 Automatic network layout

The *dot* algorithm distributed in the Bell Labs *AT&T GraphViz* package was used to automatically generate flowchart-style visualisations from the mapping matrix. These views were based on a color- and shape-coded notation, in which blue ellipses were used to denote the parameters required or modified by tasks. Tasks with only one outcome were represented as yellow rectangles and those with several possible outcomes as green diamonds.

4.4.4 Process building blocks

As a more detailed model was developed, increasing difficulties were encountered in orienting the domain experts to allow effective critique of the representation. This indicated that conceptualising tasks in terms of a textual description and their input and output parameters was insufficient; to distinguish between activities in a large model it was also important to recognise the context of a task with respect to the overall process.

The matrix used for data entry also became difficult to manipulate as the model grew in size, and the benefit of its compact representation was lost as viewport scrolling was required. In particular, the area of the matrix and hence the amount of scrolling required to view it increases with the product of the number of tasks and parameters in the model. However, the number of populated matrix cells was found to increase in approximately linear proportion to the number of tasks and parameters.[6]

The model was thus organised into hierarchical 'building blocks' defined in terms of the interfaces which specified their input and output information. *Convergence/refinement* iteration was specified explicitly using cyclic dependencies between tasks within building blocks. Iteration was also specified implicitly by re-using building blocks in different contexts within the process.

4.4.5 Process simulation

A Petri net-based algorithm was used to model the dynamic behaviour of the process, based on treating tasks as transitions and interactions as places. The approach is similar to that of McMahon and Xianyi (1996; Section 2.5.1.2; p. 44) but extended to account for resource limitations and the propagation of rework. This algorithm formed the basis of a discrete event Monte-Carlo simulation. A full description of the simulation approach is provided in Chapter 5.

[6]This reflects the organisation of the blisk design process into clusters of tasks with well-defined interfaces; a necessary architecture to reduce its complexity such that technical personnel do not need a detailed understanding of all activities and may therefore develop the necessary expertise in individual specialisms.

4.4.6 Modelling the blisk design process

Modelling proceeded by an iterative process of critique and refinement. Print-outs from the software were used to provide the focus for discussions with company personnel. The print-outs were annotated during each interview, then used by the author to refine the model prior to the next interview. Most personnel were interviewed multiple times, allowing them to critique the emerging model. This was found helpful to communicate the level of detail and nomenclature required.

Negotiating a common perspective of the process was a key challenge during modelling. In particular, many early discussions revealed substantial shortcomings in this perspective and necessitated time-consuming reconfiguration of the model's structure. This was ultimately resolved as the author's understanding of the domain increased and conventions for its representation emerged. Combined with an understanding of the limitations and boundaries of the model, these conventions allowed incremental improvement to the representation and significantly faster progress was subsequently made. They comprised three aspects:

- **An appropriate level of detail.** A consistent level of detail allowed a library of 'building block' processes to be developed, several of which were used in multiple contexts to represent activity repetition. Each building block comprised one to seven sub-tasks.

- **A nomenclature convention.** Conventions for task and parameter names supported interpretation of the model and ensured consistency. The names of tasks conducted using software tools consisted of the purpose of the task followed by the name of the tool. The names of parameters representing data files were composed from the data type (*i.e.*, file format) appended by the content (*i.e.*, the aspect of the design represented in the file). Qualifiers for these parameters were sub-divided to more precisely indicate the file content in each context. The sub-divisions were unique to each parameter. For example, the finite-element mesh of a fan blade used for stress analysis represents that blade in different forms according to the fidelity of the analysis. In this case, each qualifier used the pattern 'X Y', where X referred to the detail of the blade core and Y to the detail of the blade exterior. X and Y could each take one of a set of discrete values.

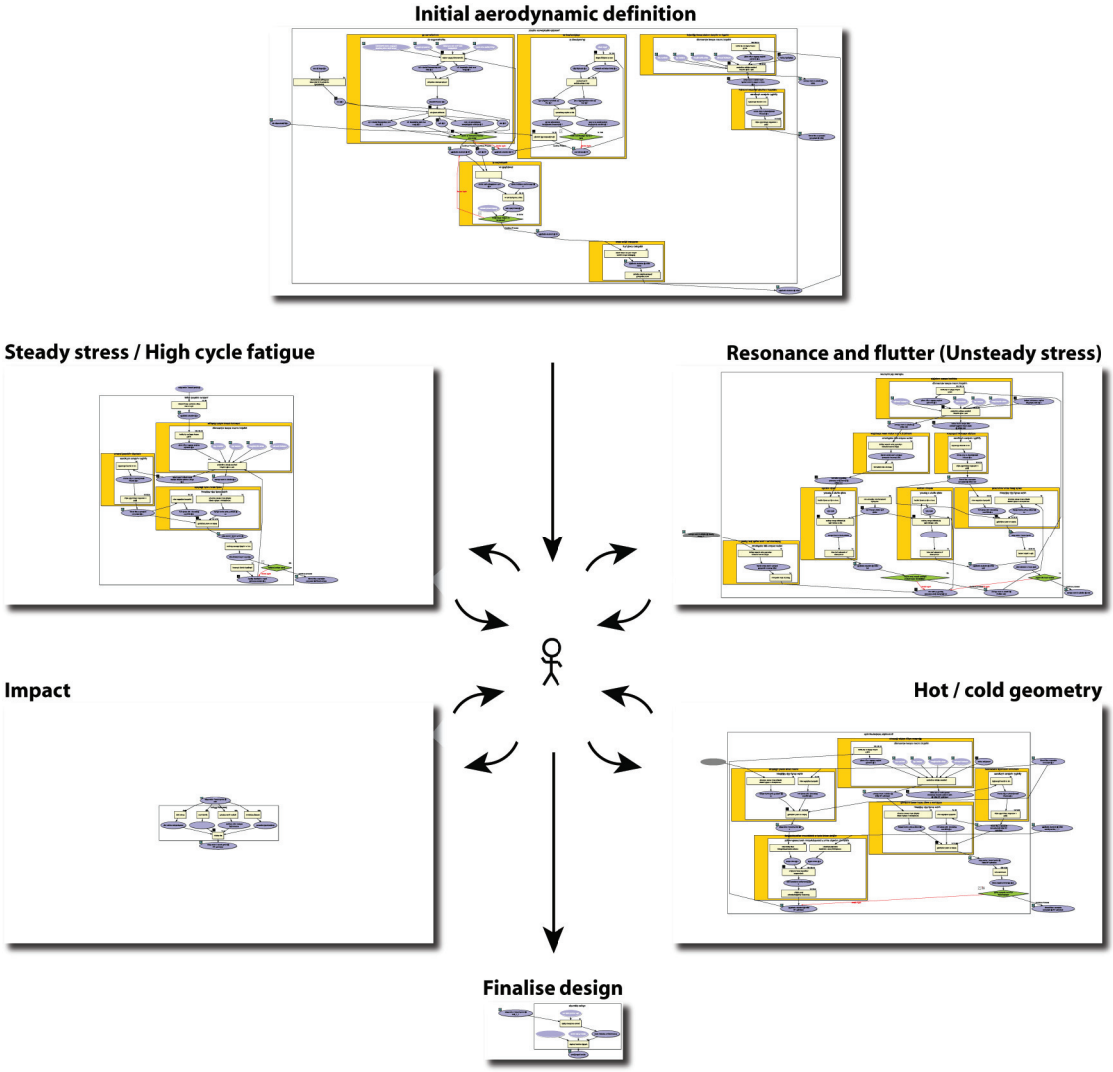


Figure 4.10 Six sub-processes comprising the hollow fan blisk design process model developed during the study. The unusual layout results from the Bell Labs AT&T GraphViz 'dot' algorithm used to generate the flowchart diagram.

- **An abstract perspective.** Blisk design can be viewed as a repeating cycle of *Define geometry - Convert data format - Analyse model - Evaluate design.* The model was thus structured as an instantiation of this sequence, such that each task or sub-process corresponded to one activity or an aggregation of consecutive activities. This ensured a consistent level of description and helped identify tasks which were previously overlooked.

This conceptual framework allowed the development of high-level building blocks to represent each of the six design activities introduced in Section 3.2 (p. 68). These building blocks are shown in Figure 4.10 above.

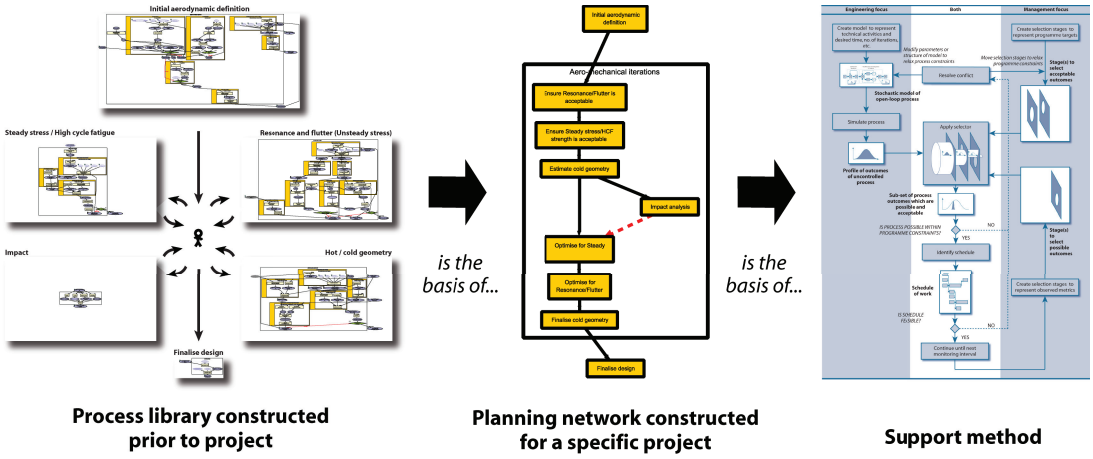


Figure 4.11 Generic process building blocks may be developed prior to a project and used to develop planning networks for a specific project.

4.4.7 Developing planning networks for specific projects

Taken together, the six high-level building blocks form a model of the generic blisk design process. This model incorporates uncertainty in the duration of each task and in the numbers of ordered *convergence/refinement* iterations which occur within each sub-process. These were modelled using triangular or uniform probability density functions. The possibility of key evaluation tasks revealing rework was also modelled using the probability of occurrence. Additionally, uncertainty about the order of attempting the four main sub-processes according to *in-situ* decisions was known to exist, but was not represented since the decision process could not be modelled with sufficient fidelity for simulation.

This model could be used to develop planning networks for simulation of specific blisk design projects by instantiating the six sub-processes and sequencing them to unroll the disordered iterations. Each sub-process instance could then be parameterised to reflect the time planned for that iteration and the perceived likelihoods of rework (Figure 4.11). Although planning networks constructed in this way may appear straightforward, this can be deceptive since interactions may occur between tasks within the sub-processes. In this case, if insufficient resources are available to fully parallelise the impact analysis and high-fidelity analysis tasks, the process duration will increase as work queues form and tasks within the processes are interleaved. These potentially complex interactions emerge from the construction of sub-processes and their arrangement in the plan. They are accounted for by the simulation and do not require explicit consideration.

4.4.8 Discussion

In summary, the case study showed that it is possible to construct a task network model of the blisk design process with sufficient detail for use in the simulation-and-selection methodology. Initial development of the model was found to be a time-consuming process, in part due to the distribution of process knowledge among specialised engineers. However, the study indicated that generic models could be constructed *a-priori* and used as the basis of specific project plans which would require relatively little effort to configure. Although the modelling focused on blisk design, Rolls-Royce personnel who had worked on other aspects of the aero-engine and in other companies believed that these findings would be applicable to other aerospace design processes.

4.5 Review of objectives and contributions

Prior to summarising this chapter in the context of the thesis, the following three sections review the research objectives, highlight the primary contributions of the method and discuss opportunities for further work.

4.5.1 Review of support requirements

The simulation-and-selection approach proposed in this chapter meets each of the support requirements identified in Section 4.1.1. To summarise:

- **SR1** Synthesise a coherent perspective of activities and their interactions from the process participants' localised understanding.

Section 4.4 (p. 106) described the development of a prototype modelling framework and its application to develop a model of the blisk design process. It was found possible to develop simulation models with sufficient detail to support project management, using an iterative process of modelling, feedback and refinement.

- **SR2** Identify a schedule which provides an appropriate trade-off between objectives.

Section 4.3.5 (p. 100) argued that this trade-off may be viewed as the identification of the most appropriate schedule from a set of acceptable alternatives revealed through simulation. A heuristic was proposed to identify such a plan by distributing the risk of over-run amongst all scheduled tasks.

- **SR3** Support monitoring and re-planning using metrics which are not expressed in terms of independent deliverables.

This objective was addressed in Section 4.3.6 (p. 101), which discussed the limitations of progress measures and illustrated how any metric expressed in terms of contributions from individual tasks may be used to inform monitoring and re-planning.

- **SR4** Function effectively given uncertainty *about* the process. In other words, application of the methodology should require neither an exhaustive understanding nor a fully comprehensive model of the process.

The applicability of the simulation-and-selection methodology despite limited knowledge about the process is a key feature of the approach. Firstly, Section 4.3.3 (p. 99) described how the process selector can be used to identify acceptable plans without requiring an overview of the simulation model or of how success criteria are met. Secondly, Section 4.3.6 (p. 101) discussed how schedules developed through the approach may be updated to account for unplanned delays.[7] Thirdly, Section 4.4.7 (p. 113) illustrated how a simulation model which describes the engineering process may be used to support detailed planning without requiring consideration of the model's full detail.

[7]The methodology may thus be viewed as providing guidance which adapts to new information in a similar sense to the Signposting approach proposed by Clarkson and Hamilton (2000) (Section 2.5.1.4; p. 52). The new modelling approach was accordingly termed the *Applied Signposting Model*. However, whereas Signposting uses a computable model to identify the next task from a designer's estimate of the design confidence, the approach presented here uses a simulation model to schedule the sequence of tasks required to complete the design based on a project manager's estimate of the process state.

4.5.2 Research contributions

The simulation-and-selection methodology makes two primary contributions to research in design process planning and management:

- The method provides a pragmatic approach to decompose complex process models into the sequences of events necessary to meet multiple success criteria. This does not require an explicit understanding of the model or how the criteria are met. It may therefore be applied where the model contains too many influencing factors or is too complex for inspection.
- The method shows one way in which the open-loop process captured in a simulation model may be used to support the management of a project which is controlled by its participants. It thereby illustrates that a mechanistic process simulation model may be used to support design practice, despite its limited fidelity.

4.5.3 Opportunities for further research

Four opportunities for further work arise from the research discussed in this chapter. These are discussed here as they are specific to the support approach and do not influence the primary argument of this dissertation:

- **Application to a project in industry.** Although the simulation-and-selection methodology is presented as a theoretical contribution, many personnel interviewed during the case study believed the approach could provide benefit to Rolls-Royce. For example:

  > *"I think it's good for motivation to be able to see how your work fits into the wider process."* — Blisk design manager.

  > *"It could be potentially quite a useful tool for getting everyone pushing in the right direction."* — Stress engineer.

  An opportunity for further work is thus to evaluate the practical utility of the method by application in industry. Due to the mission-critical nature of project management this would require a number of barriers to be

surmounted: the software tool would need to achieve near-commercial reliability; training materials would be required; and a project team identified who could commit time for the study. Verifying the approach would thus require extensive preparation and was considered beyond the scope of this thesis.

- **Evaluating progress metrics.** An interesting opportunity exists to investigate the use of simulation models to evaluate the metrics used to track project progress. Recognising limitations in the predictive utility of a basket of observations could help avoid over-optimism and fixation on inappropriate measures.[8]

- **Guidance for conflict resolution.** In the component design processes for which the method was developed, conflicts identified while generating a schedule may be resolved by direct inspection to identify, for example, tasks on the critical path. However, additional guidance would be useful when planning a more complex process incorporating parallel tasks, multiple failure modes and/or multiple success criteria. Research is necessary to investigate the application of statistical methods to explore the behaviour of large, non-linear and multi-variate simulation models which is revealed through simulation. Such methods could predict the effect of changes to the process without requiring computationally expensive simulation-based exploration of alternative configurations.

- **Identifying robust schedules.** There is a need to investigate how robust schedules can be selected from the set of acceptable outcomes identified through simulation. As the component design processes discussed in this chapter have relatively simple structure, the subset of acceptable outcomes is usually clustered such that a small deviation from one satisfactory plan should lead to another — or not to a significantly less acceptable outcome. However, if the set includes significantly different trajectories (*i.e.*, alternative tasks or orderings) it may be important to identify a plan which is both acceptable and *robust* to the uncertainties captured in the model.

[8] Although metrics play both motivational and predictive roles and therefore should not be evaluated according to predictive capacity alone.

4.6 Summary

This chapter has introduced a simulation-and-selection methodology which uses a task network model to support design project management by increasing the fidelity of plans and reducing the cost of re-planning. The approach allows identification of a schedule which meets both process and project constraints even where the simulation model is too large or complex for direct inspection. It also supports re-planning using imprecise metrics, thereby allowing application when tasks and their connectivity cannot be fully modelled.

Although the method development resulted in the specific contributions summarised above, it was primarily undertaken to provide an industry context for exploring the research questions stated in Chapter 1. The following contributions were made to the thesis:

- A prototype modelling approach and software tool were developed over a period of six months through daily interaction with industry personnel while constructing a detailed process model. This confirmed the hypothesis that a task precedence representation is appropriate to model the adaptive component design process and provided the foundations for answering the research questions outlined in Chapter 1.

- Synthesising a coherent perspective of the design process proved unexpectedly difficult. The common language provided by the formal modelling approach was useful to support this. In particular, an intuitive flowchart-style notation allowed the domain experts to interpret and critique the emerging model.

- An iterative modelling approach proved useful to both develop the model and gain the support of process stakeholders. It was identified that appropriate software could facilitate this by reducing the burden of model manipulation.

Chapter 5

Process modelling framework

This chapter addresses the first research question by detailed discussion of the *Applied Signposting Model (ASM)*. This process modelling and simulation framework was developed by enhancing the prototype approach discussed in Chapter 4.[1]

Discussion of the framework is divided into four sections. Firstly, the research objectives and methods are discussed. Secondly, the key decisions regarding the framework design are reviewed. Thirdly, the approach is described using a simple example. Fourthly, design process simulation using the approach is discussed.

[1]The Applied Signposting Model (ASM) was developed within this research project. Since it is a task precedence model and not a dynamic task model it is based on different principles to other Signposting research conducted within the Cambridge EDC. This distinction was discussed in Chapter 2.

5.1 Overview

This section introduces the research objective and methods prior to discussing the modelling framework.

5.1.1 Research objective

This chapter addresses the first research question outlined in Chapter 1. To recap:

- What attributes are necessary in a design process modelling framework used for knowledge capture, management support and process analysis?

Previous chapters have laid the foundations for identifying these attributes. Chapter 2 identified that, although many modelling frameworks have been proposed in the literature, none is agreed to adequately represent all aspects of the design and development process. Chapter 3 argued that a framework should be chosen to represent the iteration which influences process behaviour and proposed that a task precedence approach is appropriate to model component design processes. Chapter 4 evaluated this hypothesis by developing and evaluating a prototype modelling framework in the context of a model-based approach to support planning practice. The prototype framework was based on an extended Petri net approach, in which tasks were described in terms of their input and output parameters.

This led to the following research objective:

> Extend the modelling framework introduced in Chapter 4 for application to knowledge capture and process analysis, thereby addressing the first research question.

Following sections describe the extended framework in full.

5.1.2 Research methods

The *Applied Signposting Model* (ASM) was initially developed during an eight-month case study as part of a pragmatic method to support project management (Chapter 4). During the three months immediately following this study, the

prototype modelling software was re-implemented by Seena Nair, a programmer in the author's research group.

During the subsequent 21 months the ASM was refined and extended to support a broader range of process improvement activities. Throughout this period the software was developed concurrently with the modelling framework, and applied to support process representation and simulation analysis by a small number of users in academia and industry. The rapid development approach undertaken during the case study was thus maintained through regular feedback from these individuals. Each feature was considered both in terms of the aspects of the design process it was intended to represent and its implementation in the modelling interface, thereby ensuring the usability of the emerging approach.

5.2 Main framework characteristics

The ASM is a task precedence modelling framework which represents processes in terms of *tasks* and their *interactions*. The approach assumes that design processes may be characterised as the estimation and refinement of *parameters*. An ASM parameter may represent any aspect of the product or process which may change during design. This may be a direct parameterisation of the design. For example, a parameter may represent the cross-sectional area of a turbine blade internal cooling passage. It might also refer to a data file which defined the blade mesh, a report produced to satisfy a design review, or a course of action the design team intended to take.

The ASM assumes that design processes may be described in terms of changes in the availability and *state* of these parameters. Tasks attempted during the process cause parameters to become available, and/or the state of parameters to change. Likewise, a task cannot be undertaken unless required parameters are available, perhaps at a specified state.

Prior to discussing the framework in depth its key characteristics are reviewed under three headings: complex definition of the modelling framework; indirect interactions between tasks; and a primarily graphical approach.

5.2.1 Complex modelling framework

One indication of the complexity of a modelling framework is the number of classes provided and the constraints placed on relations between elements. In this sense some approaches are significantly more complex than others. The Activity DSM, for example, provides only one class of element and one class of relation which links any two elements (Steward, 1981). The IDEF3 process modelling language, by contrast, allows many types of elements and relations (Mayer *et al.*, 1995).

Applied Signposting was initially conceived as a simple framework to maximise ease of application, as this was an important requirement for the management support approach discussed in Chapter 4. However, experience gained during modelling led to development of a significantly more sophisticated framework. This provides three benefits:

- **Model precision.** Well-qualified, *i.e.*, precise models may be developed using complex frameworks. A larger vocabulary of modelling elements, together with conventions for their application, allows construction of models which are less ambiguous. Furthermore, where a number of classes and conventions are provided an experienced user can often identify a poorly constructed model (and misunderstandings on the part of the modeller) from the misuse of elements.

- **Usability of framework and modelling tools.** Learning to apply the approach is facilitated because individual framework elements may be described as solutions to particular modelling requirements.

- **Suitability for analysis.** Since many analysis techniques depend upon a computer algorithm interpreting the structure of a model, the well-qualified and precise models constructed in a complex framework are amenable to sophisticated analysis. For instance, the ASM framework distinguishes between types of interaction between tasks and parameters (Section 5.3). Together with the classification schemes described in Section 5.3.7 this allows the software implementation to conceal certain interactions and thereby focus the view on different aspects of the model.

The increased sophistication of the modelling framework is reflected in the complexity of the modelling software and, potentially, in difficulty of application for non-expert users. These issues have been addressed by ensuring the software implementation does not require a full appreciation of the framework features for application to its simplest purpose — describing processes as flowcharts.

5.2.2 Indirect interactions between tasks

The task network models reviewed in Chapter 2 allow task elements to be connected either *directly*, such as the Activity DSM (Steward, 1981), or *indirectly*, such as Signposting which defines tasks in terms of the parameters they require and produce (Clarkson and Hamilton, 2000). A framework could also use a hybrid scheme, allowing modellers to define interactions as appears most appropriate in each context. The ASM is based on an indirect interaction approach for two reasons:

- The case study described in Chapter 4 revealed that task selection in component design is driven primarily by the availability and state of design parameters. The chapter also showed that these processes may be usefully modelled in terms of tasks whose interactions are driven by the information they require and produce.

- Modelling processes in terms of indirect task interactions requires modellers to consider the factors underlying task selection in greater depth than is the case for a direct or hybrid interaction scheme. This was illustrated by the modelling of previously documented processes during the case study. It was found that the need to develop a rigorous task-parameter model revealed shortcomings in both existing documentation and the modellers' understanding. These shortcomings needed to be resolved before a useful model could be constructed. The formality of the indirect interaction approach was considered to provide benefit by highlighting assumptions and mismatches in understanding, ultimately leading to a more rigorous model considered to better reflect reality.

Modelling all tasks in terms of inputs and outputs implies that the rationale underlying task ordering is always determined by information transfers. For example, one computer code is applied after another because the first program generates data files which are required by the second. However, in practice process structures can also be influenced by factors which are not known or which are inappropriate to describe in terms of information flows. For example, a component may be designed in a certain way because it has always been done that way in the past, and because the benefits to be gained from a process reorganisation may be outweighed by the perceived risk. In these situations, task ordering must be represented without capturing the underlying rationale in terms of input and output parameters. In the ASM this leads to use of parameters to represent signals which may have no real descriptive purpose. Despite this conceptual difficulty, extensive discussions with users of the framework have led to rejection of a hybrid approach. The benefits of the rigorous indirect interaction framework were agreed to outweigh the occasional need to capture direct interactions.

5.2.3 Primarily graphical approach

Modelling frameworks may be linked to a specific graphical notation, such as IDEF3 (Mayer *et al.*, 1995), or may aim to separate the concerns of representation and user interaction, such as the Adaptive Test Process (Lévárdy *et al.*, 2004).

The method development reported in Chapter 4 revealed that a modelling approach can be strongly influenced by the interfaces provided for its use. In the case of models which must bring together previously undocumented knowledge about the design process, an appropriate user interface is critical to support negotiation of shared understanding during the modelling process. An intuitive visual format allows domain experts who are unfamiliar with the nuances of the approach to become involved in modelling. The ASM was therefore designed as a primarily graphical approach.[2]

[2]Alternative visualisations are provided by the software tool described in Chapter 6. However, the primary network view described in forthcoming sections provides the main modelling interface.

5.3 Process representation

Prior to discussing process simulation using the Applied Signposting Model, this section reviews the approach from the viewpoint of knowledge representation. The discussion is illustrated with an example model shown in Figure 5.1. Although simple, this model incorporates each aspect of the ASM approach. More sophisticated models which require up to 1300 diagram nodes to display in full are discussed in Chapter 7.

5.3.1 Precedence and dependency relationships

To recap, the task network models discussed in Section 2.5 represent the interactions between tasks in terms of either *precedences* or *dependencies*. Although the ASM is a precedence-based modelling framework, it also supports capture of dependencies within the same model. This is advantageous because although attempting a design task may involve consideration of a great deal of information, only a small number of parameters are usually considered to drive the ordering and selection of tasks. It is useful to distinguish between these roles to capture realistic process dynamics while describing tasks' information requirements in full. Without this distinction, the complex network of interactions typically indicates a large number of possible process routes, many of which may be infeasible due to constraints not captured in the model.

5.3.1.1 Modelling precedences and dependencies using interactions

The ASM framework captures dependencies and precedences between tasks in terms of their *interactions* with *parameters*. Two classes of interaction are available and are used to distinguish the role which the interaction is considered to play in driving process behaviour:

- **Data interactions** may be used to represent the requirement or production of information by a design task in the weaker dependency form which does not directly drive task selection. For example, in Figure 5.1 the parameter 'Manufacturing experience' forms part of a general backdrop of information; it plays an important role during the 'Conceptual design' task. This

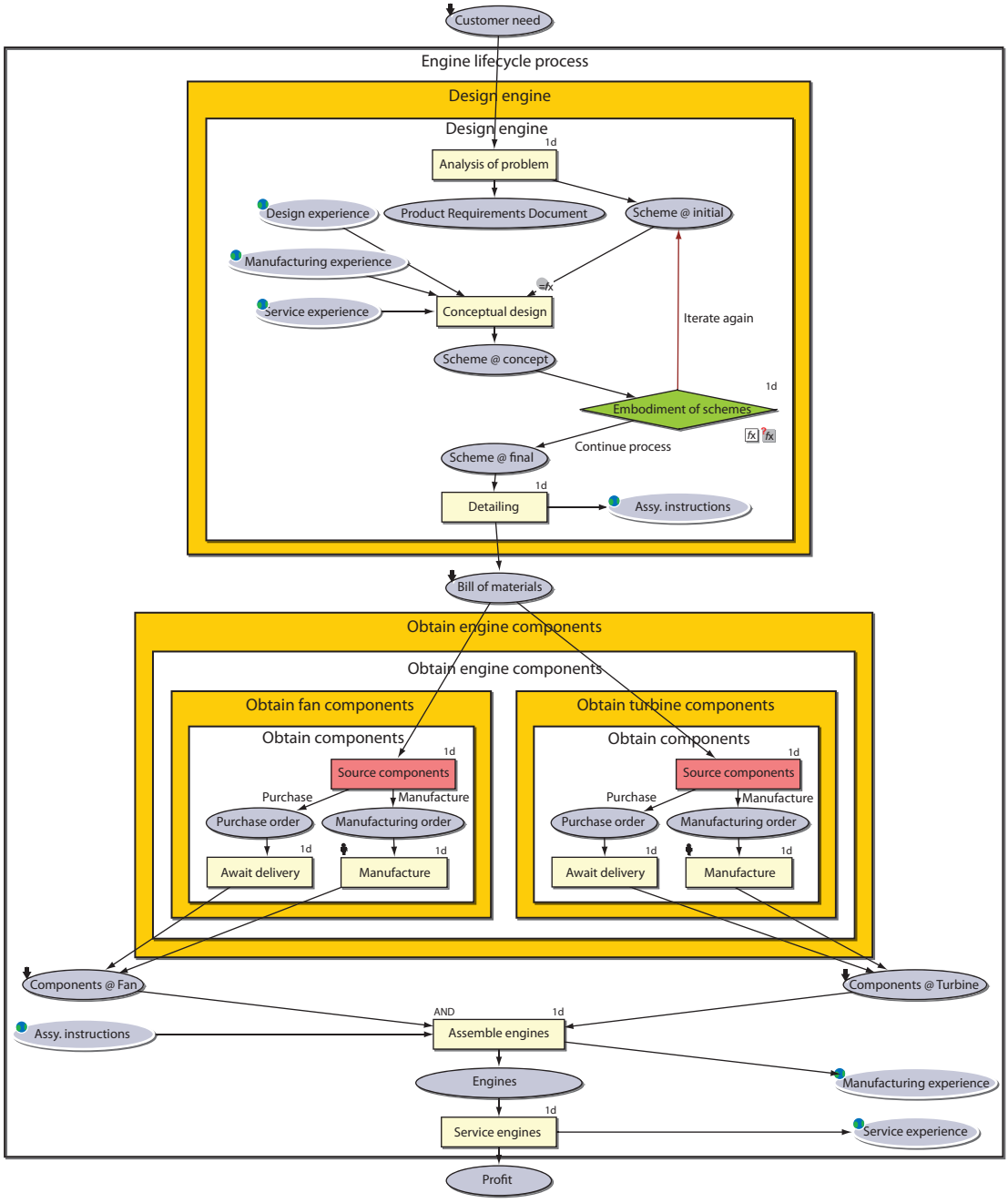


Figure 5.1 An example process model which illustrates the ASM notation.

experience is also refined during the course of the process. In the example, tasks which include consideration or modification of the manufacturing experience are modelled to include data interactions with that parameter. Although these data interactions imply feedback from 'Assemble engines' to 'Conceptual design', the dependency does not play a part in determining process routes.

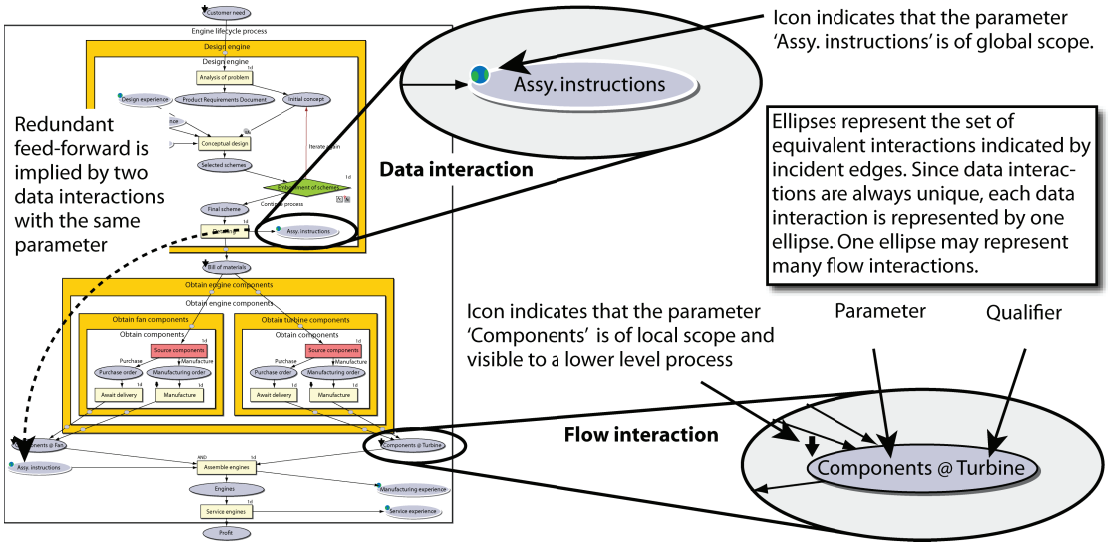


Figure 5.2 Classes of interaction.

- **Flow interactions** may be used to represent the stronger precedence requirement for a parameter to be updated prior to attempting a task. For example, it was observed during the case study discussed in Chapter 3 that most design activities were carried out within computer programs such as finite element packages which transfer data using specific file formats. In such situations, task precedences may be effectively modelled in terms of flow interactions with parameters that represent these files.

5.3.1.2 Redundant precedences

Process models may often include redundant relationships between tasks which can obscure the flow of information. In the ASM, redundant precedences may be modelled using data interactions to reduce the number of arcs in the network view and thereby effectively visualise models which incorporate many feed-forward dependencies. To illustrate, consider the two data interactions with the parameter 'Assy. instructions' highlighted in Figure 5.2. In this case, a feed-forward dependency between the tasks 'Detailing' and 'Assemble engines' may be identified as redundant. Modelling this dependency using data interactions instead of flow interactions simplifies the visualisation without compromising the fidelity of representation.

5.3.1.3 Modelling parameter state with interaction qualifiers

Interactions may optionally be *qualified* using a textual label. Interaction qualifiers are used to indicate the *state* of a parameter in that context.

In the ASM, state refers to an abstract description of the current value of a parameter. Changes in state during a process indicate that a parameter may play a different role in task selection and execution as the design progresses. The precise definition of state is context-dependent. For example, in Figure 5.1 the parameter 'Components' participates in two interactions with distinct qualifiers. In this case, the parameter is used to represent a set of indeterminate components; the qualifier indicates precisely which parts are represented in the context of a particular set of interactions. The parameter 'Scheme', by contrast, is used to represent specific design information. In this case, the qualifier labels are used to indicate the three levels of confidence or maturity of the design scheme.

Although the model assumes that process state changes in discrete steps, the set of parameters typically represents asynchronous information. In other words, the *values* of parameters may not be in accord until the process is completed.

5.3.1.4 Modelling outcomes using scenarios

Interactions are related to tasks via *scenarios*. An ASM scenario refers to a set of interactions which must occur simultaneously. For example, an evaluation task may have one input scenario comprising the interactions required to begin the task and two output scenarios, representing a successful evaluation and the unanticipated discovery of rework respectively. As discussed in Section 5.4, the model assumes that output scenarios are selected and their interactions updated as work on the task is completed.

Scenarios are graphically represented by labelling the arcs linking tasks and parameters. For example, in Figure 5.1 the interaction 'Scheme @ final' is included in the output scenario 'Continue process' of the task 'Embodiment of schemes'. For clarity only those cases where more than one scenario is available are labelled; therefore, input scenarios are never labelled, and the single output scenario of simple tasks is never labelled.

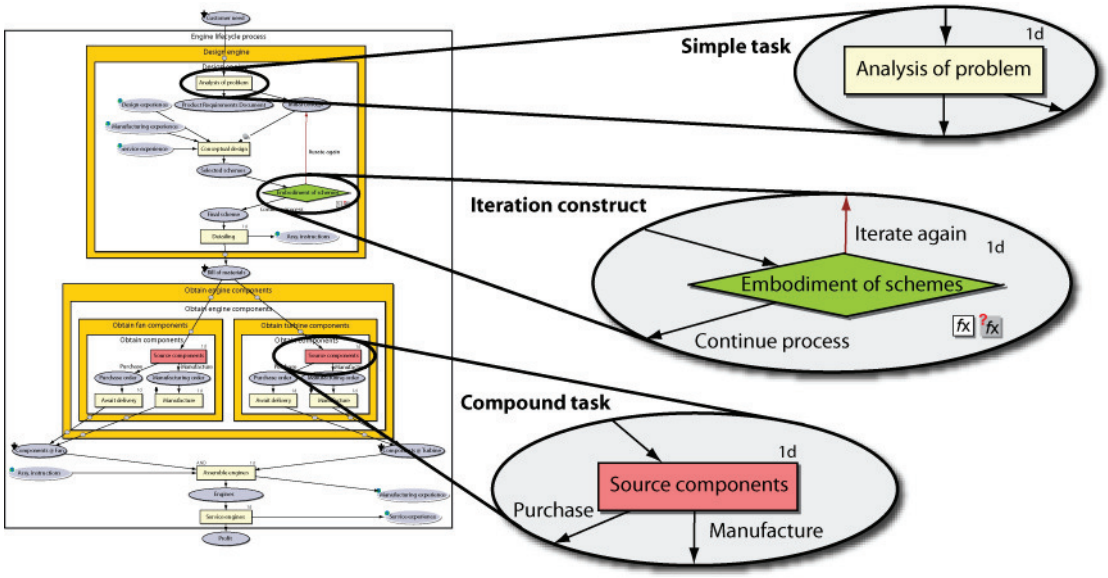


Figure 5.3 Classes of task.

5.3.1.5 Classes of task

The ASM provides three classes of task: *Simple tasks*, which have one input and one output scenario; the *iteration construct*, which has one input and two output scenarios, and *compound tasks*, which have one input and any number of output scenarios. The simple task is used to model tasks whose execution is not considered to *immediately* affect process routes, such as data file conversion tasks. The compound task is a general-purpose element used to represent any activity which may include a decision or outcome affecting the choice of next task. The iteration construct is a specialisation of the compound task, intended to represent activities which control the exit from an iterative convergence/refinement cycle. The distinction between these classes is provided to support interpretation of model behaviour from the graphical notation.

5.3.2 Hierarchical structuring

The ASM supports hierarchical structuring of tasks and parameters. The hierarchical elements have been developed in response to requirements arising during application of the approach, particularly to support manipulation and presentation of large models. They form the major driver of complexity in the modelling framework and its user interfaces. As with other aspects of the framework, com-

plex hierarchical structures may be used as required and are not necessary to construct simple models. Multiple hierarchies are available and can inter-mesh to allow development of more sophisticated structures than the other modelling approaches reviewed in Chapter 2.[3]

Hierarchies may be incorporated in an ASM model to meet four objectives:

- **Improve accuracy of knowledge elicitation and representation** by providing explicit contextual information. To illustrate, consider two similar activities carried out at different times to achieve different goals. The task elements representing them are typically described using similar terminology although the underlying activities and information requirements may be very different. In this situation it is often unclear from textual descriptions alone what each task is intended to represent. This leads to difficulty in eliciting knowledge such as information requirements or expected duration, particularly where the information is subject to interpretation or is distributed amongst several process participants. In practice, hierarchies have been found to carry more precise contextual information than purely textual descriptions, providing an intuitive representation which allows similar elements to be easily distinguished in large models.

- **Support navigation** by allowing elements to be expanded and collapsed to focus on certain aspects of the model. In practice, this is essential to enable visualisation and manipulation without overwhelming the user.

- **Support model re-configuration** by encapsulating process structures within a defined interface boundary. The case study reported in Chapter 4 revealed that it is often necessary to re-configure models in response to insights which emerge during modelling.

- **Support model re-use** by providing a mechanism to develop generic sub-processes which may be used to represent similar activities conducted in different contexts.

[3]In this context, hierarchy is defined in the broad sense which encompasses all multi-level structures such as trees and lattices. A general definition of a hierarchy, as opposed to a more general connectivity structure is challenging, since the meaning of a structure depends upon its interpretation. In general, however, a hierarchy implies *part of* and *encapsulated by* relationships between an element and its 'parent'.

The first two of these purposes are implicit whereas the latter two require explicit incorporation into a model.

5.3.3 Primary task hierarchies

The primary ASM task hierarchy has been developed to allow specification of process elements for re-use in different contexts throughout a model. For example, a process entitled 'liaise with suppliers' might represent a detailed, prescriptive procedure involving the logistics department. Since many design teams may liaise with suppliers and must follow similar steps, it would be useful to define a single process element which could be used to represent each liaison in its local context.

To achieve this, ASM process elements are not decompositions of a higher-level task as in most other modelling frameworks. Instead, each process is composed of other processes. As a model is constructed its processes accumulate in a process library from where they may be referenced multiple times if required. An implication of re-using process elements is that task hierarchies are not constrained to a tree structure. The task hierarchy in an ASM model is typically lattice-structured, as shown in Figure 5.4.

An ASM process element is defined as an encapsulation of a set of tasks and parameters, together with input and output interactions exposed from within the process. In Figure 5.1, 'Customer need' is exposed as an input interaction and 'Profits' as an output interaction from the process 'Engine lifecycle process'. Process elements differ from tasks in that, where permitted by the structure of information flows within, work on the process may begin before all input interactions occur and output interactions may occur before all work is completed. Activities which exchange information during execution may thus be represented as aggregates of subordinate tasks.

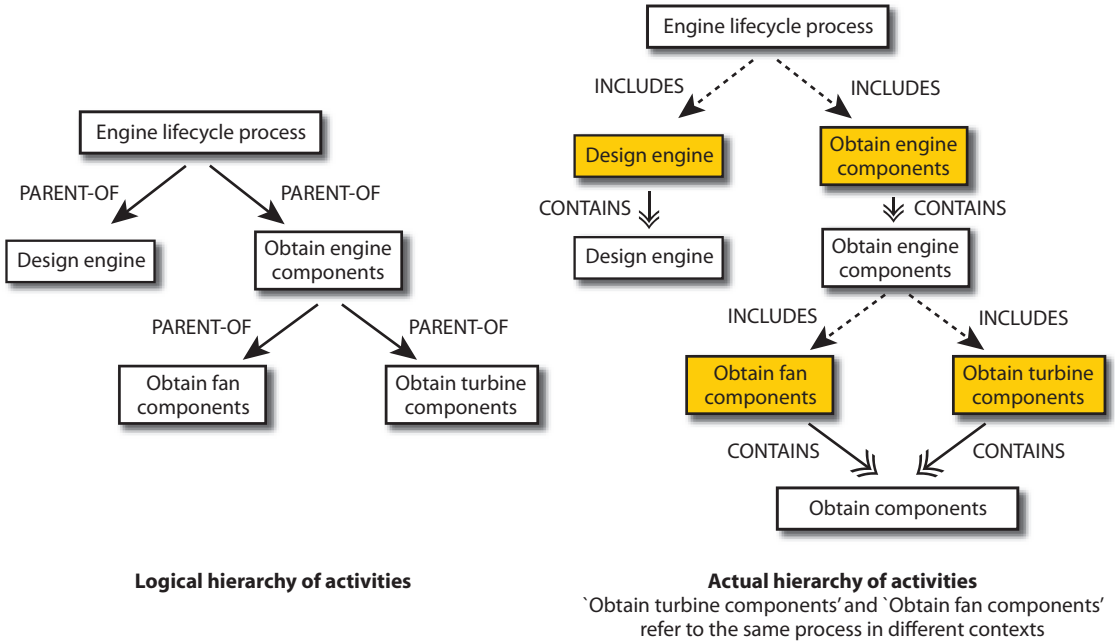


Figure 5.4 The example model illustrates how processes are composed from containers, which refer to other processes. Multiple containers can refer to the same process, thereby allowing it to represent similar activities in multiple contexts.

5.3.4 Indirect model composition

Each task in a process defines its own context in terms of input and output interactions which refer to parameters within the process. Tasks in a process interact directly because their contexts overlap by reference to the same parameters. Unlike a task, a process in the ASM has no 'parent'. Its context cannot therefore be defined in terms of parameters with which other tasks may directly interact.

Each time a process from the library is inserted into a parent process appropriate context is provided by wrapping it in a *container*. This is an element in the parent process which *maps* the exposed interactions of the wrapped process to interactions defined in the parent process. A process thus provides detail for one or more container tasks and each container task provides one context for its wrapped process.

Figure 5.5 illustrates how a container provides context for a process, showing the mapping detail which is hidden in the primary view. In this example, the 'Components' ellipse depicts an interaction exposed from the process 'Obtain components'. The container 'Obtain turbine components' maps this interaction

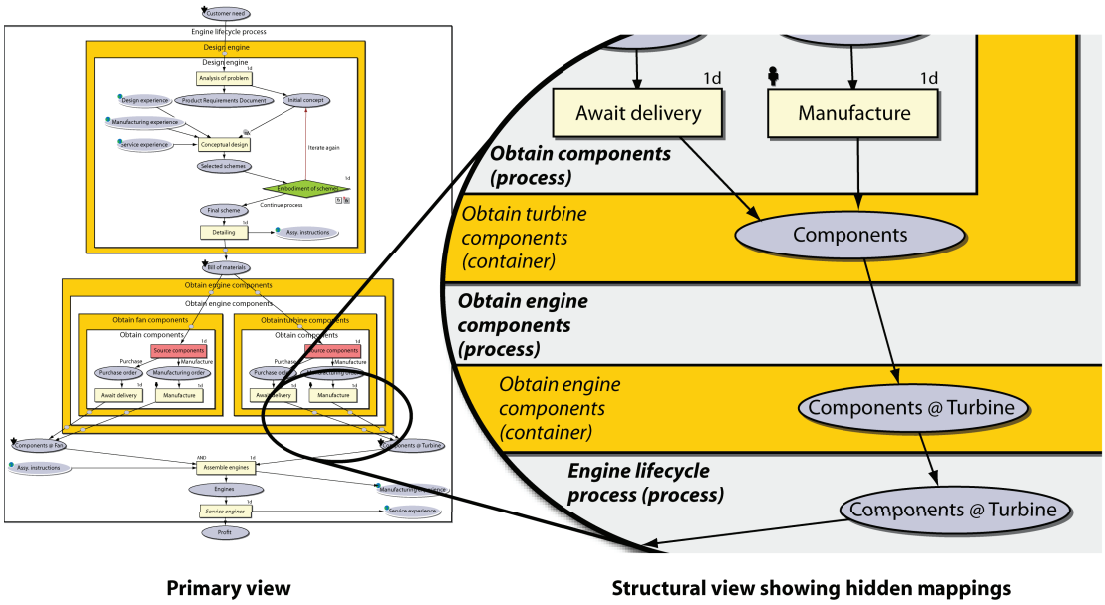


Figure 5.5 The mapping detail which is hidden in the primary view of Figure 5.1.

to 'Components @ Turbine', which is defined in the process 'Obtain engine components'. This interaction is in turn exposed as an output from the container 'Obtain engine components' and mapped to 'Components @ Turbine' defined within 'Engine lifecycle process'. An exposed interaction may only be mapped to an interaction with the same parameter, although the *qualifiers* may be different to allow a process to be used in different contexts throughout the model. This is illustrated in Figure 5.1, where the process 'Obtain components' is used to represent the separate acquisition of fan and turbine components. To achieve this, the two containers map the exposed output interactions to different interactions within the super-process.

Although this scheme appears complex on paper, the details of container tasks and mappings are managed transparently by the software implementation. The benefit of the approach is that library processes remain independent of the contexts in which they are used. Once the interfaces are frozen, processes may thus be modified directly from the library and all instances will be updated accordingly. This facilitates incremental development and maintenance of a knowledge base from which many models may draw.

5.3.5 Parameter namespaces

In many processes the hierarchy of activities can be seen to parallel organisational structures. For example, stress analysis is carried out by mechanical engineers and its detail may not be well understood by the aerodynamics group. This structure is reflected to some extent by the visibility of design information; intermediate data files generated during stress analysis may have little relevance beyond this activity and the team who carry it out. However, other information such as component definition files or requirements documents is shared among several teams to co-ordinate their work. A representation of design information should therefore be sufficiently flexible to reflect the hierarchy of activities or permit deviation from this structure.

The ASM therefore allows parameters to be specified within different *namespaces*:

- **Model namespace** allows any task in any process to interact with the parameter. Parameters in model namespace are referred to as *global parameters*, and should be used to represent information which is developed over the course of a project. Such parameters may be structured into trees as discussed below.

- **Process namespace** defines the parameter as an element within a given process. Parameters in process namespace are referred to as *local parameters* and should be used to represent information which is of transient importance and is encapsulated by a process boundary. Local parameters are thus secondary to the task hierarchy of previous sections, such that each process encapsulates one or more tasks and a number of parameters.

  A common requirement is to specify parameters which are visible to a group of sub-processes while remaining encapsulated from the rest of the model. To facilitate this, a local parameter is considered *exposed* from its parent process if any interactions with that parameter are exposed to the process interface. The parameter is then visible to all processes which contain its parent process. This maintains logical consistency of the model, since any parameter which represents information transfer across an interface must by definition be visible to both processes.

Parameters with intersecting namespaces cannot have the same name. The namespace of a global parameter intersects that of all other parameters in the model; its name must thus be unique and should be fully-qualified. In contrast, a local parameter may have a partially-qualified name because its context is determined by its location in the activity hierarchy. For example, distinct local parameters representing distinct bladefiles in the 'compressor design' and 'turbine design' processes could simply be entitled 'Bladefiles'. However, if either or both of these parameters were located in model namespace they should be entitled 'Compressor bladefiles' and 'Turbine bladefiles' respectively.

Namespaces are indicated on the primary network view by icons placed above each interaction ellipse (Figure 5.2). A globe above an interaction ellipse indicates that the associated parameter is of global scope, *i.e.*, every interaction with that parameter refers to the same information. An up arrow indicates that the parameter is of local scope and that it is exposed to all super-processes. A down arrow indicates that the parameter is exposed to one or more sub-processes.

5.3.6 Primary parameter hierarchies

The ASM framework allows global parameters to be structured into trees to allow description of parent-child relationships (*e.g.*, 'engine requirements' incorporates 'fan requirements') and thus to support modelling of large numbers of parameters at different levels of detail. The precise interpretation of the parent-child relation is dependent upon the parameter definitions. To illustrate, consider two primary usages of ASM parameters: to represent *aspects* of the design; and to represent *descriptions* of the design. In the former case, a sub-parameter *is part of* its parent; in the latter, the sub-parameter *incorporates aspects of* the parent. In the latter, several design descriptions may reflect the same conceptualisation of the design, but are formatted appropriately for specific tasks. For example, at one point in the blisk process introduced in Chapter 3 aerofoil definitions exist as both splines and solid meshes. These definitions are used for aerodynamics and FEA analyses respectively.

In the example above, 'fan requirements' might be modelled as a sub-parameter of 'engine requirements'. In this case, a task which interacts with 'engine requirements' is considered to also interact with 'fan requirements'. Conversely, a modification to 'fan requirements' is assumed not to drive changes in the super-parameter 'engine requirements'.

5.3.7 Secondary hierarchies

The method development discussed in Chapter 4 revealed that in practice models quickly become large enough to present difficulties in navigation and manipulation. In these cases it is necessary to work with subsets of a model to reduce the computational requirements of manipulation and prevent 'information overload' for the user. Although the hierarchies described above support this to some degree, it is often beneficial to define perspectives which cut across the primary structure of a model. For example, to examine the role which data files play in the process it is useful to conceal parameters representing other types of information. Similarly, when developing a model through feedback to several process participants it can be advantageous to focus on individuals' tasks and the contexts in which they take place.

The primary hierarchies described in previous sections are fundamental to the composition of a model. Since it may be necessary to develop many perspectives of a model, perspective information should be superimposed on the primary structure so viewpoints can be easily defined and modified without requiring careful consideration of all model elements.

To serve this need, the ASM allows definition of secondary hierarchies for both tasks and parameters in terms of multiple user-defined classification schemes and classification categories. Any task or parameter may be placed in a number of schemes and categories, enabling mark-up of these elements with additional information. The software implementation described in Chapter 6 can process these structures to develop perspectives of the process model. Network views, for example, can be filtered to provide simplified visualisations whose layout is coupled to the primary view.

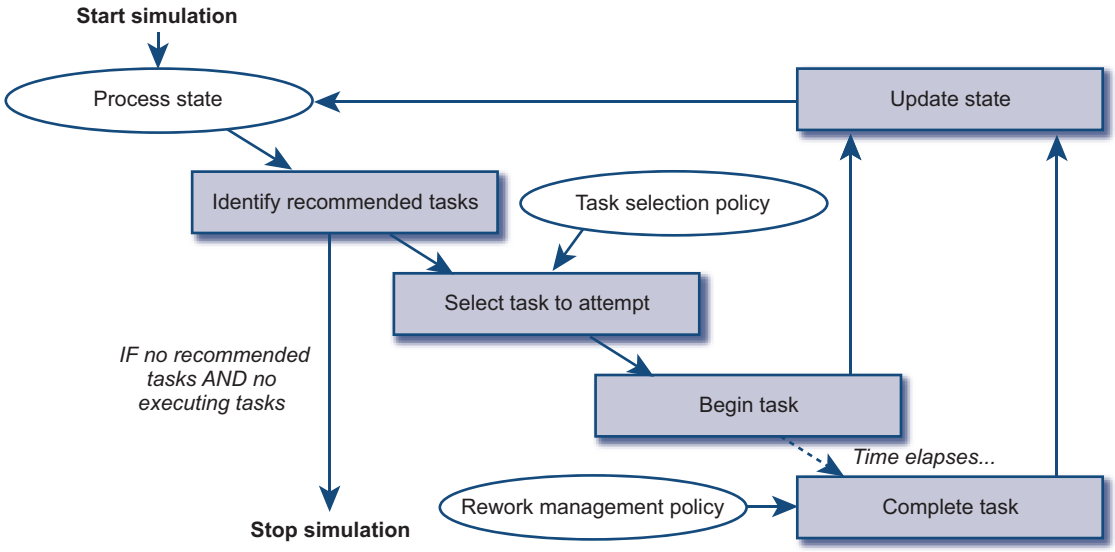


Figure 5.6 An overview of the process execution algorithm.

5.4 Process simulation

A key motivation for adopting a process modelling framework is that such approaches offer benefits beyond informal graphical models. In particular, a model may be analysed to determine ways in which the underlying process might be improved.

The ASM supports step-wise *execution* of process models. Simulation techniques based on Monte-Carlo integration may be used to explore the behaviour of executable models. For example, it may be valuable to calculate the criticality of individual tasks or to examine the trade-offs between metrics such as process duration, resource utilisation and schedule risk.

The execution algorithm used by the ASM is summarised in Figure 5.6. This is a discrete-event algorithm governed by a representation of process state, which is sequentially modified as events are processed. It is discussed in detail in the following sections.

5.4.1 State representation

Process state is represented as the three aspects of *information availability*, *resource availability* and *process variables*. These aspects are discussed in turn.

5.4.1.1 Information availability

The execution algorithm is required to operate effectively upon an ASM process described using the intuitive graphical notation introduced in previous sections. The model of information flow in an ASM process is thus based on the Petri net approach. This allows execution of models structured intuitively as flowcharts; tasks are attempted in the order implied by flow interactions in the model (Figure 5.7).

Each task in an ASM process corresponds to a Petri net transition and each input interaction to each task is associated with a place. In contrast to standard Petri net places, ASM interactions take one of three states: *unavailable*, indicating that the information represented by the interaction's parameter is not available at the state represented by the interaction's qualifier; *available*, indicating that the information is available; and *updated*, denoting that the information has been modified since the last attempt of the task and therefore that rework may be required. In general, it is assumed that some subset of the input interactions must be *available* prior to attempting a task and some subset must be *updated*. This will be elaborated in Section 5.4.5.1.

Upon initialisation of the algorithm, all interactions are *unavailable* except those which are either specified as a process input or are not specified as an output from another task. The states of these interactions are initialised to *updated*, thereby determining the first task(s) which may be attempted.

5.4.1.2 Resource availability

The ASM framework allows specification of *resource requirements* for each task. Each resource requirement indicates a number of units from a given *resource pool* which the task requires to execute. Each resource pool represents a set of fungible entities and consists of a *calendar* which indicates the hours and days for which the resources are available and an *availability profile* which describes how many units are available between given dates.

Any required resources are removed from their pools upon starting a task, and returned when the task is complete. Where several tasks compete for the same

Figure 5.7 An example illustrating the steps in process execution.

resource it is assumed that one task is selected and executed at full capacity. Other competing tasks cannot then be attempted until the first is complete. Subject to other aspects of model configuration, resource constraints may thus affect simulation behaviour.

5.4.1.3 Values of process variables

Process variables may be defined to explicitly represent numeric values which may change during a process. This is useful for three purposes:

- **Instrumentation.** Many metrics used to evaluate a process may be linked to events in the process. For example, the rates of accumulating cost, increasing confidence and decreasing risk in a process may all be decomposed into contributions from individual tasks. Encoding such contributions into the simulation allows application of a process selector to identify processes with satisfactory performance (Chapter 4) as well as perturbation analysis to predict the effects of changes to the model upon the performance. The ability to include performance measures in the simulation could also support future work to identify improvement opportunities by direct analysis.

- **Modelling conditional probabilities.** A task's outcome and duration are often conditional upon prior events. For example, the time required to reassemble an engine module after servicing is dependent upon the degree to which it was disassembled; and the likelihood of discovering rework following testing may reduce with the number of iterations prior to the test. A means to incorporate such conditionalities is necessary as they are extremely common and can strongly influence simulation behaviour.

- **Modelling process decisions.** It may be useful to explicitly model decisions which are made within the process. For example, as a design is refined more time may be devoted to detailed analysis with each iteration; the iterative refinement of a component may continue until some parallel activity is completed; and the next design or analysis activity selected at a point in the process may depend upon the current confidence in aspects of the design.

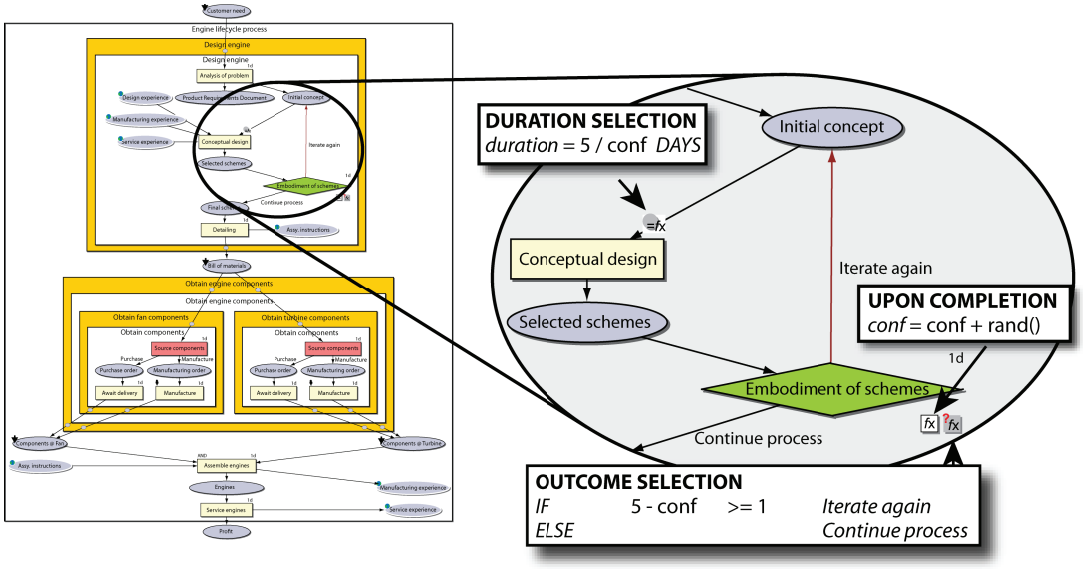


Figure 5.8 A simple example which illustrates how process variables and functions may be used to control the simulation behaviour.

The ASM allows variable values to be modified by *functions* which are defined using an intuitive *C / Java* -style syntax and attached to the output scenarios of tasks.[4] When a task is completed during execution, any specified functions are evaluated to determine the new value of each process variable in the model. Expressing task durations and scenario selection decisions as functions of process variables then allows simulation models to be developed which exhibit dynamic behaviour more realistic than Markov models.

A simple example illustrating the incorporation of 'learning' into a convergence/refinement cycle is shown in Figure 5.8. In this case, the duration of the 'Conceptual design' task is assumed to vary in inverse proportion with the confidence in the design when that task is attempted. Confidence is assumed to increase by a uniformly-distributed random value between 0 and 1 following completion of the subsequent 'Embodiment of schemes' task. At that point, the concept must be re-visited unless confidence has reached a threshold value of 4. Although this example makes use of a single variable, it is possible to construct multi-variate functions which represent the combination of many factors in determining the behaviour of each task.

[4]The *Singular Systems* library *Java Expression Parser (JEP)* is used to evaluate functions and determines the precise syntax.

In summary, process variables and functions provide a flexible means to incorporate many behaviours. The functionality allows users with limited programming knowledge to develop graphical models exhibiting system-dynamic behaviour.

5.4.2 Task selection and execution

At each step, simulation proceeds by identifying recommended tasks, selecting a task for execution, and beginning that task. When all recommended tasks have been started, the simulation moves forward to the next date at which a task finishes or at which a change in resource availability occurs.

5.4.2.1 Identify recommended tasks

A traffic light scheme similar to the original Signposting approach is used to represent the suitability of tasks for execution. At any point in time, the state of each task in a model is either: *not possible* (red), *possible* (orange), or *recommended* to begin (green).

The state of each task is determined by examining the information state of the model and the required input information for that task. Due to the hierarchical definition, only the state of information in the task's parent process can affect its state. Processes are thus described in terms of: 1) *knowledge* about individual tasks and their input/output characteristics; and 2) *assumptions* regarding the limited scope of a task's effect upon other tasks in the model.

Once the state of each task has been determined, resource requirements are examined and any *possible* or *recommended* tasks for which insufficient resources are available are removed from consideration.

5.4.2.2 Select task

If more than one task is *possible* or *recommended* to begin, a *task selection policy* is used to identify the highest priority task for immediate execution. The choice of policy affects simulation behaviour if information constraints allow concurrency but resource constraints limit the number of tasks which may be attempted in parallel. In other cases the policy plays no role. Task selection policies are discussed in more detail below.

5.4.2.3 Begin task

When the selected task is attempted, the process state is updated by removing the required number of resource units from their pools and setting all input places to *available*. Depending on the task's configuration, its duration is determined by selecting a value from a probability distribution or by evaluating a function of the process variables. The duration is extended to account for non-working time in the calendars of any resources used by the task.

5.4.2.4 Complete task

Upon completion of any task a single output scenario must be selected. The states of all output places in this scenario are set to *updated* and any resource units used by the task are returned to their respective pools. Scenario selection can be modelled as a stochastic process by specifying the probability of each scenario occurring. Alternatively it may be specified in greater detail using process variables and functions, as discussed above.

5.4.3 Resource conflicts

In practice, prioritisation of tasks which compete for limited resources is difficult to model as it is influenced by the local decisions of personnel who execute the tasks. A variety of factors might impact upon these decisions, including: the perceived importance of the information or criticality of the task; the designer's enjoyment of each task; and the impatience or seniority of any personnel who may be waiting for output information.

The ASM simulation code allows task selection policies to be provided as plug-in modules. Policy modules in the default implementation are: *random recommended task*, in which a task is selected at random from the possible options; *shortest duration first*; and *longest duration first*. The most 'effective' policy depends upon the structure of the model, where effectiveness may be defined according to multiple criteria. For example, a policy might be chosen to minimise expected process duration or maximise robustness of expected duration to variability in task outcomes. Whichever criteria are chosen, effectiveness is dependent upon the model structure as well as the policy.

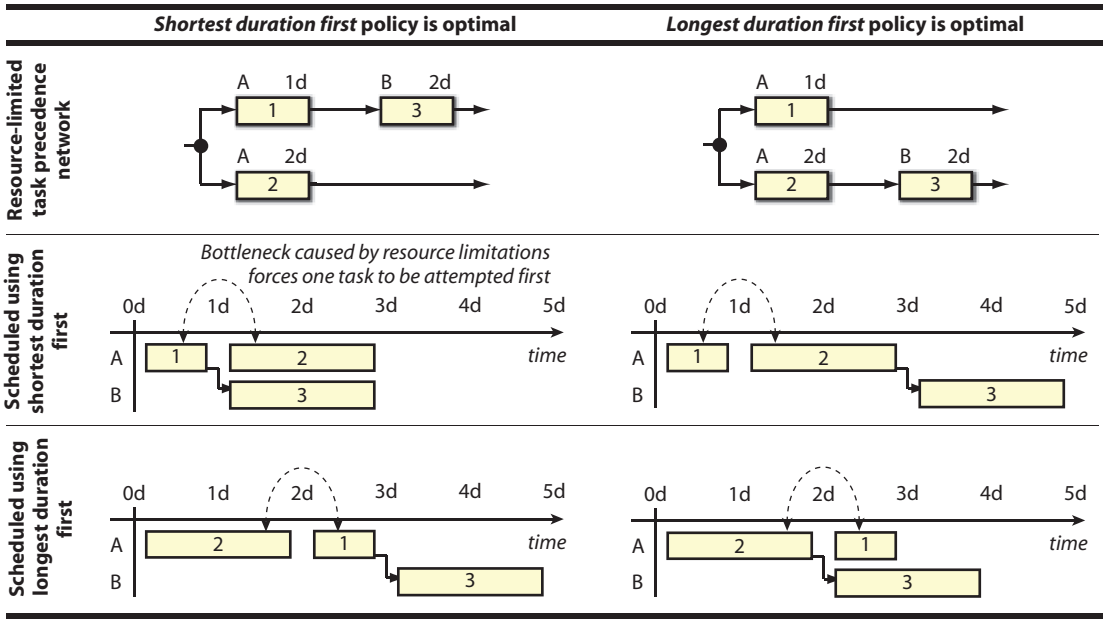
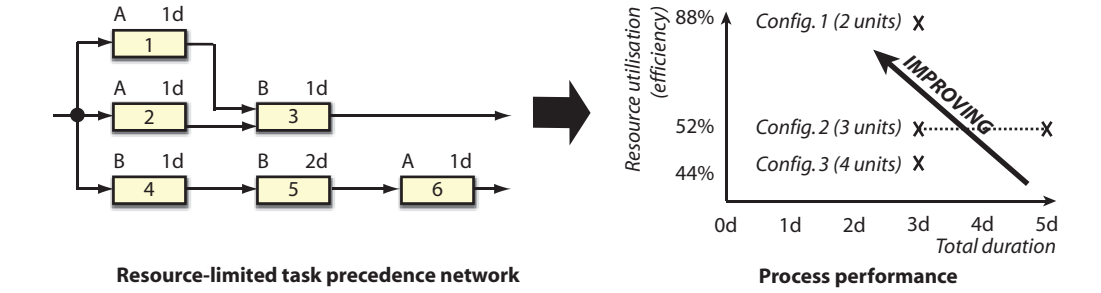


Figure 5.9 The most effective task selection policy depends on the model structure. In this example, two resource-limited configurations have minimum duration under different policies if 1 unit of resource A and 1 unit of resource B are available.

Figure 5.9 illustrates this using an example process in which *shortest task first* results in the lowest possible duration, and another in which *longest task first* results in faster completion. In general, the *random recommended task* should be used as this results in a different sequence of decisions on each simulation run and therefore introduces a source of variability which represents uncertainty in task prioritisation. This is appropriate where the actual policy cannot be accurately determined and incorporated in the model.

5.4.4 Complexity of resource bottlenecks

Although ASM models may appear simple in the absence of process variable-controlled dynamics, this can be deceptive since limitations in resource availability can cause non-linear and counter-intuitive behaviour. To illustrate, consider the resource-limited model shown in Figure 5.10. In this case, the minimum configuration of one unit in each resource pool 'A' and 'B' causes a bottleneck between tasks '1' and '2' (Configuration 1). This may be resolved by adding another unit to resource pool 'A', thereby allowing the conflicting tasks to be performed concurrently. It might be expected that this would lead to a reduction in total process duration.



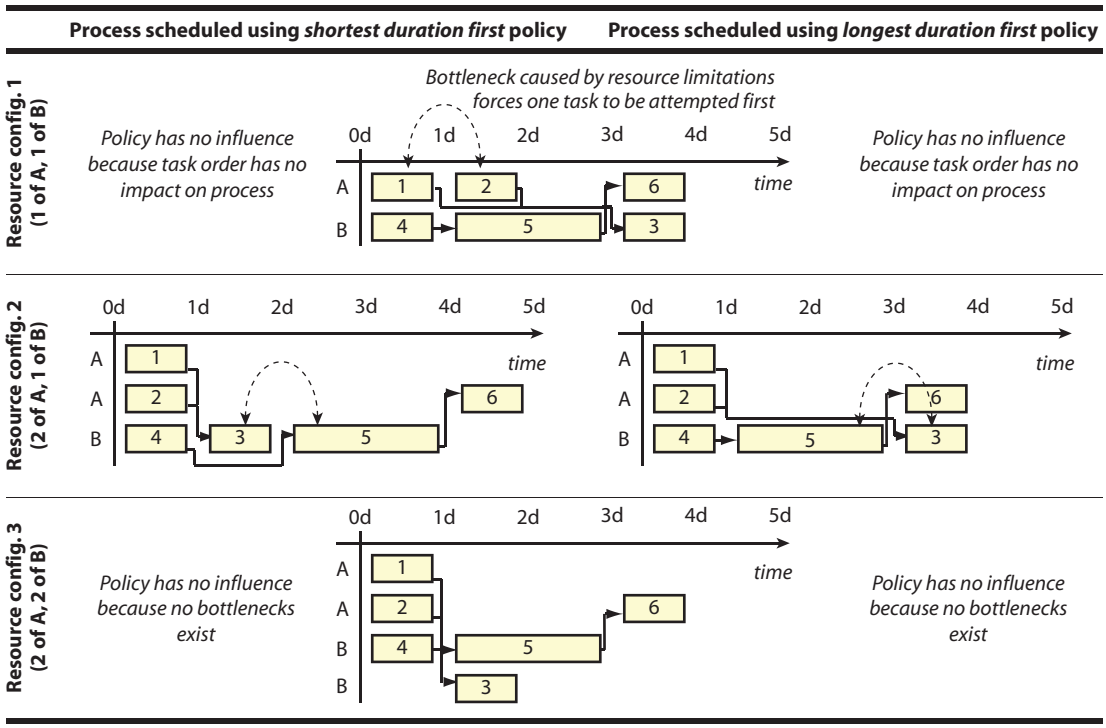


Figure 5.10 Resource bottlenecks can cause counter-intuitive behaviour. In this example, additional resources intended to resolve a bottleneck reduce process performance in terms of both process efficiency and robustness to task selection policy.

However, closer examination reveals that resolving this bottleneck would cause another to be introduced between tasks '3' and '5' (Configuration 2). By the measure of resource utilisation, this is a less efficient configuration. Furthermore, the altered configuration would no longer be robust to task selection policy; whereas Configuration 1 is insensitive to the ordering of attempting tasks in the bottleneck, a poor decision in Configuration 2 leads to an increase in overall duration. Although continuing to add resources would resolve all bottlenecks in this example, the process would become increasingly inefficient for no reduction in duration (*e.g.*, Configuration 3).

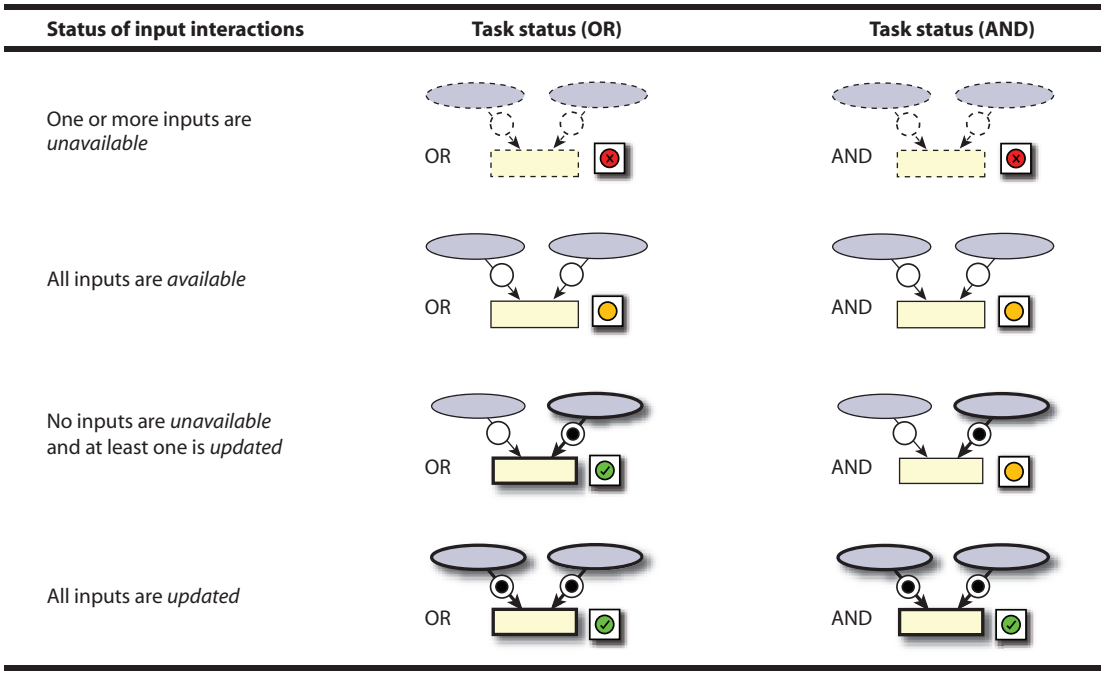


Figure 5.11 The execution status of a task is determined by examining its execution conditions and the states of each input interaction.

5.4.5 Determining the consequences of rework

To recap, rework is required when completion of a task affects information which has already been incorporated into the design. In the ASM simulation this may be detected when an interaction's state changes from *available* to *updated*. Once identified, the consequences of rework are determined by considering the three factors identified in Section 3.4.3.4 (p. 82):

5.4.5.1 Determining which tasks must be re-attempted

The status of a task following updates to input information is governed by the *execution conditions* for that task. The appropriate execution conditions are dependent on the task's context and the desired behaviour of the process model. For example, most design activities require rework following the modification of any input information. Tasks representing these activities should be specified as requiring *updates to any* input interactions to begin. In contrast, activities which integrate parallel work streams cannot be attempted until all streams are complete and all input information has been updated. Integration tasks therefore require *updates to all* input interactions (Figure 5.11) before rework may begin. Note that, regardless of execution conditions, a task may never be attempted

unless every input interaction is either *available* or *updated*. Tasks which require rework compete for resources on an equal basis with tasks attempted for the first time.

5.4.5.2 Determining how much rework is required on each task

A task is likely to involve different amounts of work on the first and subsequent attempts. Furthermore, the outcome of a task may also depend upon the number of iterations of that task. These differences are not explicitly modelled in the ASM. They should be incorporated on a case-by-case basis using process variables and functions, as outlined above.

5.4.5.3 Determining how the rework will be managed

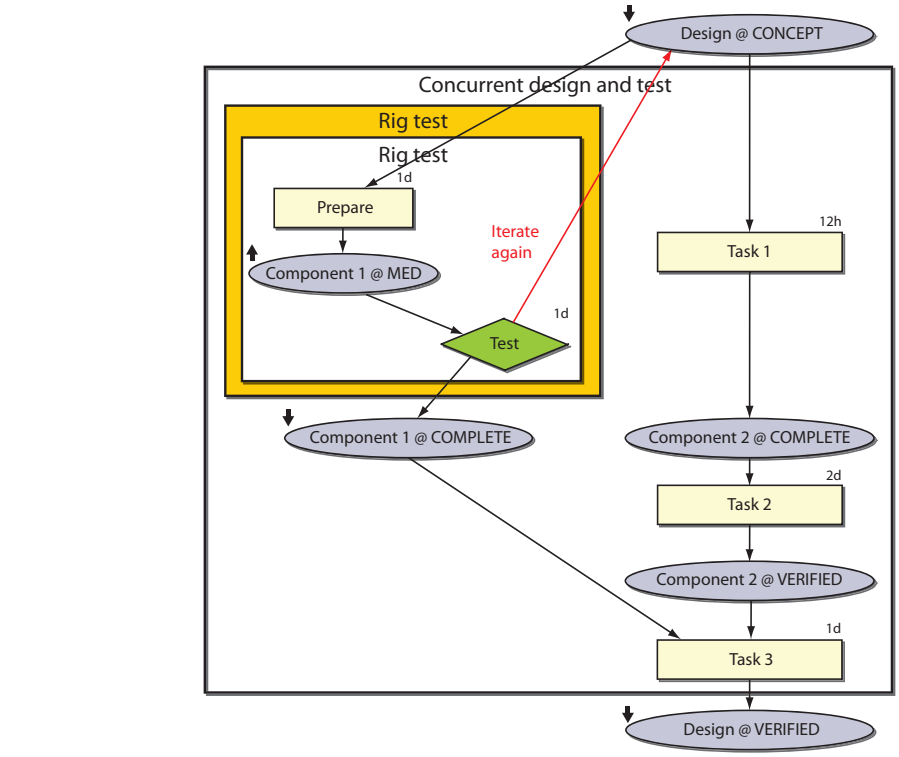
In practice, tasks are often attempted based on incomplete information due to the need to compress project schedules. Rework of one task may invalidate assumptions and ultimately require many downstream tasks to be re-attempted. For example, rig tests are usually carried out concurrently with design work in anticipation of success. If successful, the test will meet contractual obligations and provide information which may feed forward into later design work. However, the test may also indicate problems in design work which has already been completed, invalidating assumptions and requiring complete or partial rework of many tasks. In this scenario, the outputs of any directly or indirectly dependent tasks in progress, or which have already been completed may be invalidated. When this is detected, work on any potentially invalidated tasks should be reconsidered immediately. The invalidated tasks cannot be fully completed until their input information has been regenerated by rework of invalidated predecessors. Management decisions must be made regarding the most appropriate actions in such cases, *i.e.*, which tasks should be re-attempted and in what order.

The ASM simulation represents management actions in response to rework using plug-in modules which implement alternative *rework management policies*. Three policies are provided by the current implementation (Figure 5.12):

- **No action** assumes that explicit management actions are never taken following rework discovery. This represents the policy that work on dependent tasks and their successors will continue regardless of the invalidated assumptions, although they must ultimately be attempted again.
- **Optimistic** assumes that evaluation tasks are the only activities which can cause propagated rework. In anticipation that all such evaluations will be successful, only the failure of an iteration construct forces downstream information to be invalidated.
- **Pessimistic** assumes that all indirectly-dependent tasks may become invalid after any information update and are therefore immediately interrupted to prevent unnecessary effort. Interrupted tasks will only be reattempted once their input information is updated following the completion of rework upon their predecessors.

The rework management policy has no effect on a model in which no task may be started until all input information has been finalised, unless that task is part of an explicit iteration cycle. In other cases, the most effective policy depends on the behaviour of each task on subsequent attempts. For example, if a task's duration reduces upon each attempt, and if sufficient resources are available, it may be best to continue executing the task even when it is known that rework will later be required. Although continuing to work would increase the total time spent on that task, the learning factor would require less effort to be expended on the later attempt and possibly shorten the critical path of the project.

The recommended rework management policy is *optimistic* because this most closely reflects observed practice.



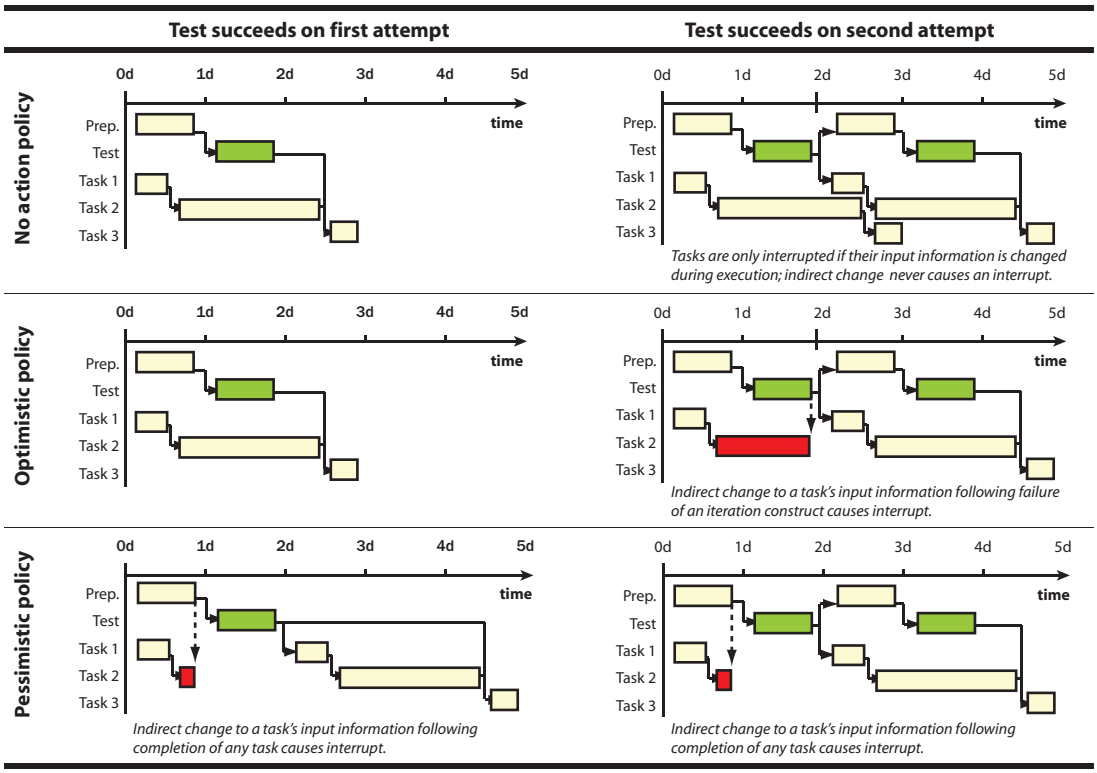


Figure 5.12 An example illustrating the *optimistic*, *pessimistic* and *no action* rework management policies.

5.4.6 Applications of the ASM simulation

The discussion of simulation in this thesis focuses on support for project management. However, the flexible infrastructure of the ASM simulation can be applied to explore a broader range of problems. These include:

- **Generating widely-applicable insights from system dynamics models.** System dynamics models are useful to explore the behaviour of general classes of process. As such analyses can be extremely sensitive to the underlying assumptions, it is usually important that models comprise few enough tasks and controlling process variables to allow a review of all influencing factors.

- **Generating specific insights from detailed models.** Due to the dependency of a process' behaviour on domain-specific factors such as the structure of information flows, insights and heuristics derived from analysis of a specific model may not be valid for all processes. Simulation of detailed models is useful to investigate the behaviour of a particular process and to pose questions at a level of detail suitable for suggesting improvements. However, the large number of tasks and influencing variables in such a model can cause difficulties in verifying that the simulation behaviour reflects the modeller's intention. It is therefore important that this behaviour is relatively straightforward.

- **Developing improved configurations.** Many authors have proposed that process simulation can be used to identify improved process configurations, based on perturbation analysis or other computational techniques (*e.g.*, O'Donovan 2004). However, such methods assume that the model's response to structural or parametric changes would reflect an equivalent change to the underlying process. As discussed in Chapter 3, this level of fidelity may be difficult to achieve in practice. The validity of these computational methods should therefore be carefully considered, especially where processes depend on socio-technical interactions such that there are many 'unknown unknowns'. Despite these limitations, simulation can still be a useful tool to support process design — in the same way that limited-fidelity analyses are useful to support decision-making in engineering design.

5.5 Summary

This chapter has introduced the Applied Signposting Model, a computable formalism for modelling and simulation of engineering design processes. The discussion addresses the first research question outlined in Chapter 1. To summarise, the key attributes of the ASM are:

- Intuitive graphical notation. A diagrammatic representation allows ASM models to be readily understood by individuals unfamiliar with the approach, thereby facilitating the elicitation of process knowledge from domain experts and the validation of models.
- Complex definition. The ASM provides a large number of descriptive elements to support the development of precise models and aid in their interpretation.
- Multiple hierarchical structures. Product development processes in industry are typically extremely complex. The hierarchical structures provided by the ASM framework support representation of such processes by allowing the development and manipulation of very large models.
- Flexible representation of dynamic behaviour. Process variables, task selection policies and rework management policies allow a wide range of behaviours to be modelled using the ASM simulation.

The dissertation has thus far focused on process modelling in the context of management support. Chapter 6 discusses the implementation of the ASM in a configurable software platform, which was subsequently applied to support a broader range of process improvement activities (Chapter 7). This ultimately allowed validation of the approach against the original research questions and success criterion stated in Chapter 1.

—— This page is intentionally blank ——

Chapter 6

Modelling and simulation software

This chapter describes the *P3 Signposting* software platform. This software was initially developed to implement the process modelling and simulation framework described in Chapter 5. It was later extended to form a more general modelling platform which may be used to support further research. The chapter therefore addresses the second research question outlined in Chapter 1.[1]

Discussion proceeds in six sections. Firstly, the research objectives and methods used to develop the software are outlined. Secondly, the platform architecture is discussed and the facilities provided to extend the software are summarised. Thirdly, the interface for constructing ASM models is briefly discussed. Fourthly, the simulation functionality and interface for exploring simulation results are introduced. Fifthly, the ability to configure the platform to develop new modelling frameworks is outlined using a product modelling framework as an example. Finally, dissemination of the tool is discussed to indicate its relevance to academic researchers and industry practitioners.

The chapter provides a research-oriented overview of the software. User-oriented documentation is available in the 100-page *P3 Signposting Users' Guide* (Wynn and Clarkson, 2006). Further detail is provided in Appendix C.

[1] *P3 Signposting* was conceived and developed within this research project. It is not based on previous 'Signposting' software developed in the Cambridge Engineering Design Centre.

6.1 Overview

This section reviews the research objectives and methods prior to discussing the software architecture and features.

6.1.1 Research objectives

The research reported in this chapter was undertaken to address two objectives. The first arose from the second research question outlined in Chapter 1. To recap:

- What are the requirements for design process modelling and simulation software to support academic research and industry practice?

This led to the following research objective:

> Develop a software tool which allows construction and simulation of process models based on the Applied Signposting Model.

Although this objective is stated in implementation terms, the resulting software forms a significant contribution to research as it has allowed other researchers to apply and extend the ASM modelling framework. This is discussed further in Chapter 7.

The second objective was derived from a methodological observation generated during the method development reported in Chapter 4. The study incorporated prototype software development as an important component of the research methodology, undertaken to allow continuous evaluation of the emerging method and thereby ensure the research direction was appropriate to industry requirements. Although this approach proved successful, the majority of the author's effort was ultimately expended in software development. It was subsequently hypothesised that a configurable platform could have significantly reduced the time spent programming and therefore allowed effort to be concentrated on the research aspects of method development. This led to the following research objective:

> Develop a configurable software platform to reduce the effort required to construct linkage-based modelling tools.

6.1.2 Research methods

Software development formed an integral part of the research reported in this dissertation. Following development of the Excel prototype during the eight-month case study discussed in Chapter 3, around 36 additional developer-months were expended to design and implement the software discussed in this chapter. Software development therefore forms a significant part of the research project, in terms of both effort expended and influence on the research direction. Figure 6.1 summarises key features of subsequent versions of the software.

The P3 Signposting software was developed by the author in conjunction with Seena Nair, a programmer in the author's research group. The author was responsible for defining the major requirements and architecture of the software and was closely involved in all aspects of the code development. The author accounted for approximately 50% of the programming time expended, weighted towards the end of the development period.

The following aspects were developed entirely by the author: the ASM framework; the software concept and requirements; network filtering and managed layout code; process simulation and scriptlet code; profile explorer; process selector; all linkage meta-model features; and the plug-in architecture. Other code was developed by or in conjunction with Seena Nair, namely: the drag and drop network interface; network generation, trees, dialogs, data structures, file handling and process execution for the Applied Signposting Model.

6.2 Platform architecture

P3 Signposting is dynamically assembled at run-time from modular software components. Each module requires a globally unique identifier, depends upon a specified set of modules and libraries and attaches to a specified *plug-in point*. New modules may in turn define new plug-in points, thereby allowing extension of the software for purposes beyond the scope of the original research. An integral versioning system is provided so that modules may be individually updated and released with a minimum of co-ordination. New or updated modules may be easily installed using a graphical update wizard provided as part of the core package.

	Excel prototype 1	Excel prototype 2	Java prototype	Release 0.7.0	Release 0.7.2
Overview	Development and initial implementation of the modelling and simulation approach		Refining the modelling and simulation approach to improve its flexibility, together with enhancing the analytical tools and user interface of the modelling software		
Hierarchies	Flat models		Lattice-structured models, classification schemes and process library		Hierarchical parameters
Modelling interface	Domain-mapping matrix indicating input and output parameters for each task	Multiple domain-mapping matrices (one per sub-process)	Direct manipulation of network representation, with context-sensitive operations		
Visualisation	Matrices and automatically generated GraphViz network layouts.	Matrices, GraphViz network layouts and experimental network editor	Multiple, configurable and interactive GraphViz layouts with network filters	Managed layout algorithm allowing user to manipulate node positioning, with network filters that maintain relative node locations	
Interoperability	Office VBA implementation interacts directly with standard productivity software		Cross-platform Java implementation exports data to common formats		
Decision modelling	Task outcome and duration selection modelled as stochastic process	Iteration construct allows specification of planned numbers of convergence/refinement iterations		Incorporation of process variables enables task outcome and duration to be modelled in detail, *e.g.*, to incorporate conditional probability distributions.	
Parameter interactions	Tasks transform parameters between discrete qualifiers			Parameter interactions are separated into two flavours: flow interactions driving task selection; and data interactions driving task execution.	

Figure 6.1 A summary of the features in subsequent versions of the P3 Signposting software.

6.2.1 Core modules and plug-in points

The *p3-core* and *asm-core* modules encapsulate the core functionality of P3 Signposting and define a number of plug-in points. For example, it is possible to provide modules which extend the user interface by adding items to toolbars, menus or dialogs. Other modules — such as the *executor* which governs the order in which tasks are attempted during simulation — provide functionality which is more tightly integrated with the system and do not affect the user interface.

The set of plug-in points in the core modules was designed to allow implementa-

tion of many interesting extensions with a small amount of code. For instance, the module *asm-core* defines the plug-in point *task-selection-policy*. New task selection policies may be developed by implementing an abstract method in a class which is part of the *asm-core* module package; this method simply returns a single *Task* object which is selected from the list of arguments.

Module code must be compiled and packaged in a *.zip* file with a *module descriptor* which provides information required to load the module, together with any documentation. The software distribution includes a directory containing many such packages. A bootstrapper reads this directory when the application is started, using the descriptors to ensure that all dependencies are satisfied and to determine the order for loading modules. At the time of writing, the main P3 Signposting distribution consists of 74 modules and libraries (Figure 6.2).

6.3 Applied Signposting modelling interface

This section discusses the modelling interface provided to construct Applied Signposting models. Modelling is primarily conducted using the network diagrams introduced in Chapter 5. The model is manipulated using intuitive click-and-drag operations in this view. Right-clicking an element shown in a network, tree, or tabulation opens a context menu from which that element's properties are configured. A search facility allows elements to be located by name, connectivity or other properties (*e.g.*, find all tasks which use a specified parameter as output, find all tasks with uncertain duration).

6.3.1 Multiple views

The software provides multiple views of the model to support manipulation of large data sets. The primary network view allows ASM process elements to be viewed individually or within the context of their parent process(es). Various tabulation views are available, including a matrix tabulating tasks against resources, a task precedence DSM in which connectivity is determined by flow interactions alone, and a task dependency DSM which indicates possible connections between tasks that include data or flow interactions with the same parameter. Additional views of ASM processes may be provided as plug-in modules.

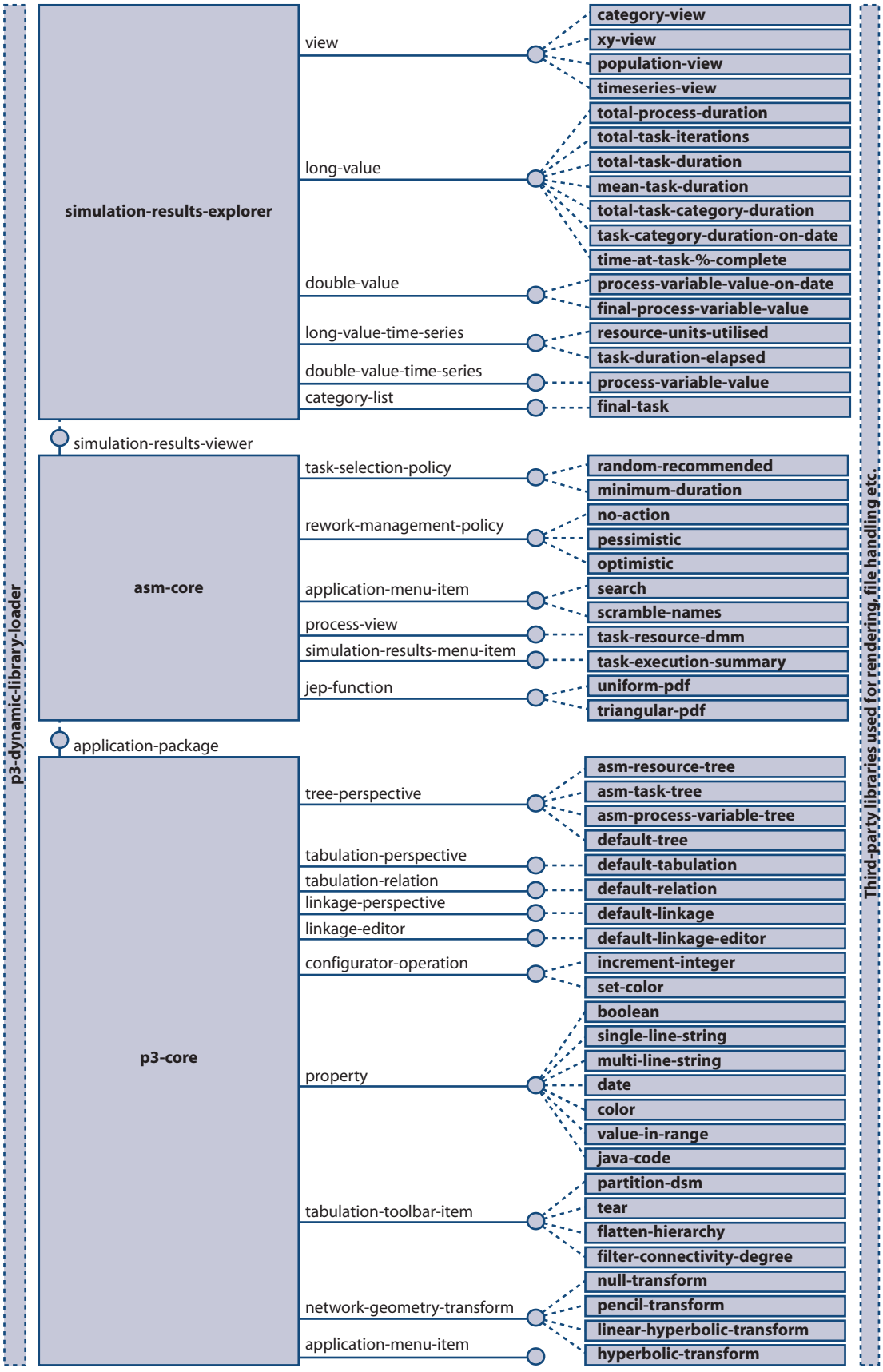


Figure 6.2 Modules and plug-in points in the core distribution.

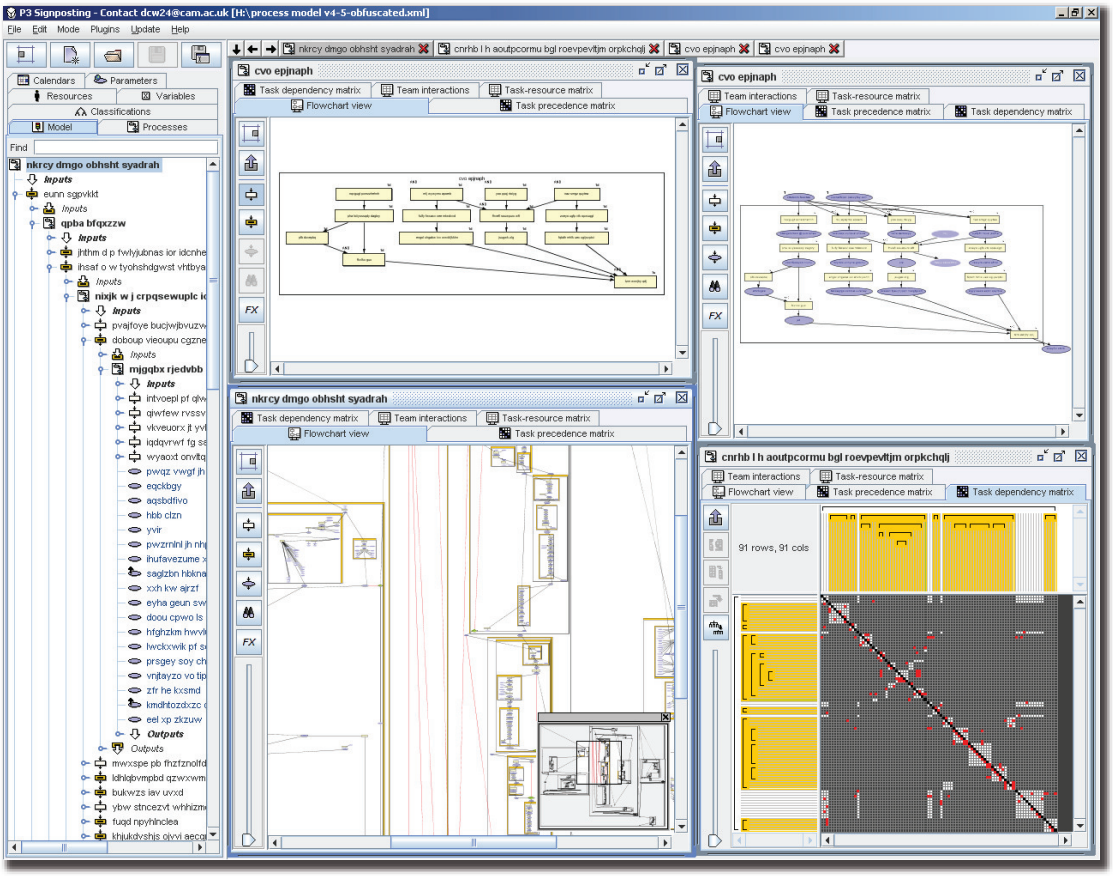


Figure 6.3 The *P3 Signposting* software provides multiple tabulation- and network-based views to support manipulation of large models.

6.3.2 Filtering the network view

It is often useful to filter a process network to view a subset of a large model (Chapter 5). For example, expanding or collapsing a sub-process allows the user to view either the detail or context of an activity; concealing all data interactions highlights the flow of information in a process; and focusing the view to show only those tasks and parameters in a certain category can be used to highlight the contribution of certain participants to a process.

View filters operate by matching and replacing nodes in the default network view. Filters are repeatedly applied until no more matches are found. For instance, a filter to conceal parameters in a given category matches nodes which represent interactions with those parameters. Those nodes are then replaced by edges which link the source and sink nodes directly, thus providing a simplified visualisation while maintaining the structure of information flows in the diagram.

6.3.3 Managed network layout

Diagrammatic representations provide an intuitive interface for describing processes and are informally used for this purpose in industry (Chapter 3). However, the suitability of a general-purpose diagramming interface depends upon models being relatively small and weakly connected such that elements may be manually arranged to attain a pleasing layout. In practice, however, process models often consist of a large number of elements whose connectivity must be continuously manipulated as the model is developed. In consequence, the overhead of layout manipulation limits the utility of many diagrammatic modelling tools.

The prototype modelling software developed during the case study automatically generated layouts using the general-purpose AT&T GraphViz *dot* algorithm to reduce the burden of manipulation (Chapter 4). Although this allowed rapid development of the prototype software a number of shortcomings became evident. In particular, to be effective for process modelling an automatic layout algorithm should consider more than aesthetic factors such as minimising edge intersections or screen area. This requirement arises from the tendency for modellers to organise their diagrams such that the layout supplements or reinforces information represented by the elements and relationships in the model.

To illustrate, a number of engineers commented during the method development that appropriate positioning of tasks relative to their input and output interactions was more important than achieving compact or aesthetically pleasing layouts. On large layouts the AT&T GraphViz *dot* algorithm often positioned tasks well above some of their input interactions. In these situations the misplaced inputs were usually overlooked when reading the diagram — suggesting that in addition to layout, the interpretation of formal diagrams may be further influenced by the semantics, or perceived semantics, of the underlying model. This is reinforced by Crilly *et al.* (2005) who suggest that in diagrammatic elicitation exercises it is necessary to consider the connotations of diagrams in addition to their direct denotations. Despite the importance of cognitive factors in influencing the effectiveness of network visualisations, few publications in the graph layout community address these issues (Ghoniem *et al.*, 2004).

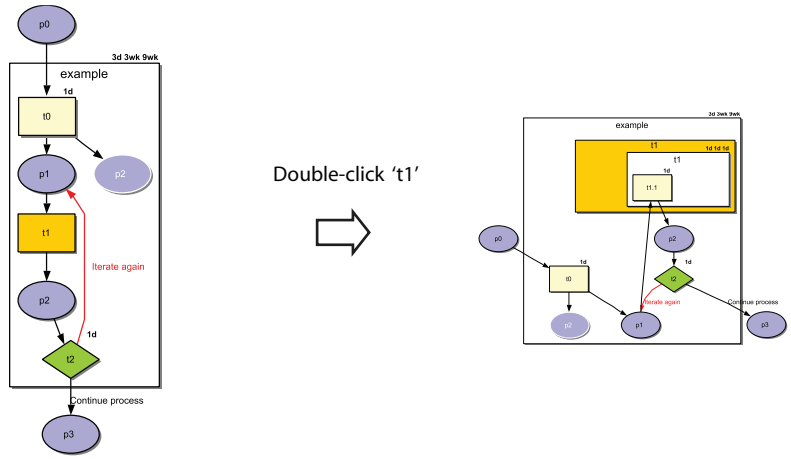


Figure 6.4 The GraphViz 'dot' algorithm produced significantly different layouts following minor changes to the model or view configuration.

An additional shortcoming was that a minor change in the underlying model or view configuration often resulted in a significantly different network layout (see Figure 6.4 for one example). This did not prove a significant obstacle to the author. However, it was later found that many individuals encountered great difficulty using this system.

Based on these findings, the following requirements regarding network visualisation were developed:

- Allow the user to influence the diagram layout.
- Minimise the overhead of manipulating large layouts.
- Support network filtering while maintaining the spatial relationships between individual diagram elements.

The P3 Signposting software incorporates a managed layout algorithm to address these requirements. The algorithm maintains a dimensionless grid for each process in an ASM model, which comprises the relative positioning of all nodes in the primary network view. When a new layout is required (*e.g.*, after opening a new window, opening/closing a container, applying a filter, moving a node using the mouse, or creating a new task or interaction) the algorithm proceeds as shown in Figure 6.5. The algorithm recurses over all open container tasks, *i.e.*, the size of the container is calculated by first laying out its wrapped process.

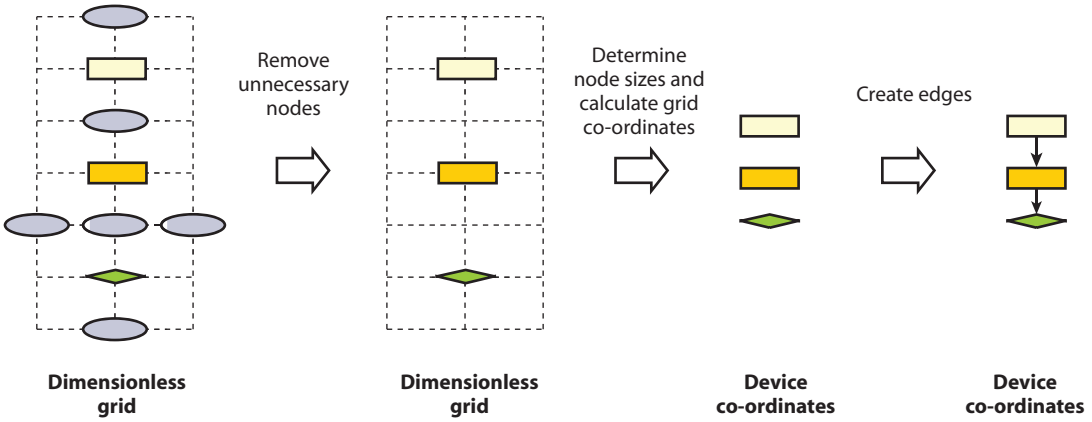


Figure 6.5 Layouts are based on a dimensionless grid which is maintained for each process. When a network view is required, unnecessary nodes are removed, node sizes and co-ordinates are calculated and edges are created.

As all nodes are aligned along their centre-lines, their spatial relationships remain constant when containers are expanded or collapsed. When a node is moved in the user interface, it snaps to a new position on the dimensionless grid and the entire layout is re-generated.

6.4 Process simulation interface

This section describes the process simulation interface under four headings: the process debugger for verifying simulation behaviour; the profile explorer for graphing simulation results; the process selector for identifying subsets of process outcomes; and the scriptlet interface, which provides a flexible method to extend the behaviour of the ASM simulation model.

6.4.1 Verifying simulation behaviour

The software provides a debugging mode to support the verification of simulation behaviour.[2] This mode allows the user to step through the process model and examine the state of the process following the start or finish of each task. Traffic-light symbols are used to represent the state of tasks and interactions during debugging, as indicated in Figure 5.7. Once a model has been verified, Monte-Carlo simulation may be used to generate a profile of process outcomes.

[2]Verification aims to ensure the simulation behaves as intended by the modeller. Verity does not imply *validity* — the more difficult question of whether a model adequately represents reality.

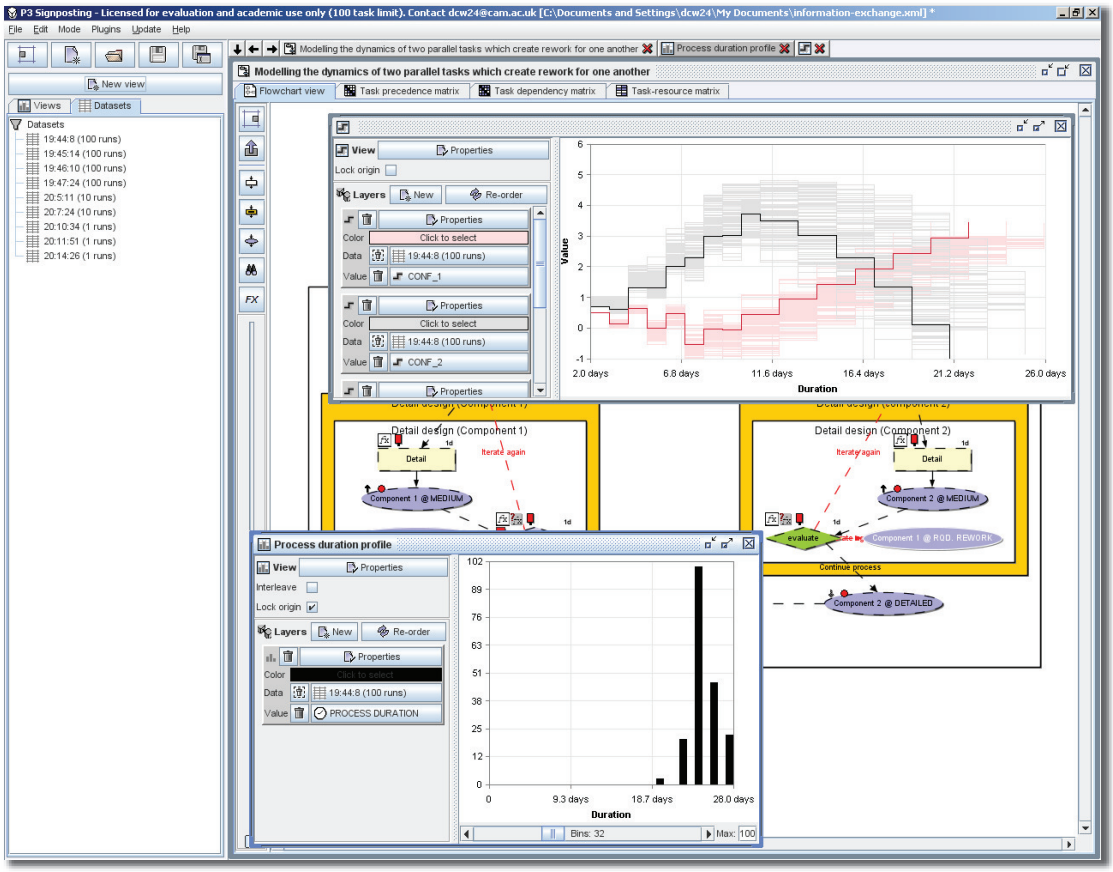


Figure 6.6 The profile explorer. Plots comprise one or more layers, where each layer consists of one or more value providers (number and type depending on plot) and a dataset.

6.4.2 Profile explorer

The *profile explorer* provides functionality for graphing simulation results. The explorer uses *value providers* which analyse the log of each run in a simulation dataset to generate a value or set of values for graphing. Value providers may return: a *single value* (*e.g.*, *total-process-duration* returns the total duration of the simulation run); a *time series* (*e.g.*, *process-variable-value* returns the value of a process variable at each time during the simulation run); or a *category list*, returning a list of categories of which the simulation run is a member. Most value providers are configurable; for example, *process-variable-value* requires the user to select the process variable of interest. A number of such value providers are included in the core distribution, and more may be developed as plug-in modules.

A number of charts are provided to explore the variables returned by value providers. At the time of writing these include: *histograms*, used to indicate

the expected values of a variable; *x-y scatter plots* for identifying correlations between two variables; *time series plots* which may be used to identify the times at which certain values occur (*e.g.*, what is the expected cost accumulated on a given date? on what date is a certain task complete with 90% certainty?); and *category pie/bar charts*, to show the occurrence of each category (*e.g.*, how many processes exceed the project deadline?). Multiple layers may be added to each chart, allowing direct comparison of simulation results following changes to a model (Figure 6.6). A plug-in point allows implementation of additional chart types.

6.4.3 Process selector

The profile explorer incorporates a *process selector*. Selectors may be used to identify processes which satisfy specified constraints, without requiring an understanding of how the constraints are met (Section 4.3; p. 99). They are constructed from stages, each of which acts as a band-pass filter for a specified value provider. When a selector is applied to a dataset, another dataset is created representing the original set of simulation results filtered to remove those runs which are excluded by any stage. The new dataset may be plotted alongside the original and the band-pass filters modified to explore the result in real time.

In addition to identifying subsets of processes, selectors allow multi-variate and multi-criteria *what if?* analyses to be interactively performed. For example, to explore the impacts of certain tasks' durations on several process performance criteria, the duration probability density functions for all tasks of interest would be extended to encompass a wider range of values. Although x-y plots of the duration of each task against each criterion could be constructed, a large number of such charts would be necessary to identify the effect of each variable on each performance criterion. The effects of combinations of variables are even more difficult to visualise using static graphs. To explore this problem using the process selector, a selection stage would be created to filter the duration of each modified task. The performance criteria for the unmodified process would be plotted as histograms or x-y plots, and the selector results overlaid upon these charts (Figure 6.7). Moving the selector bars then clearly indicates the effect of a particular combination of the task durations. Performing a similar analysis without this

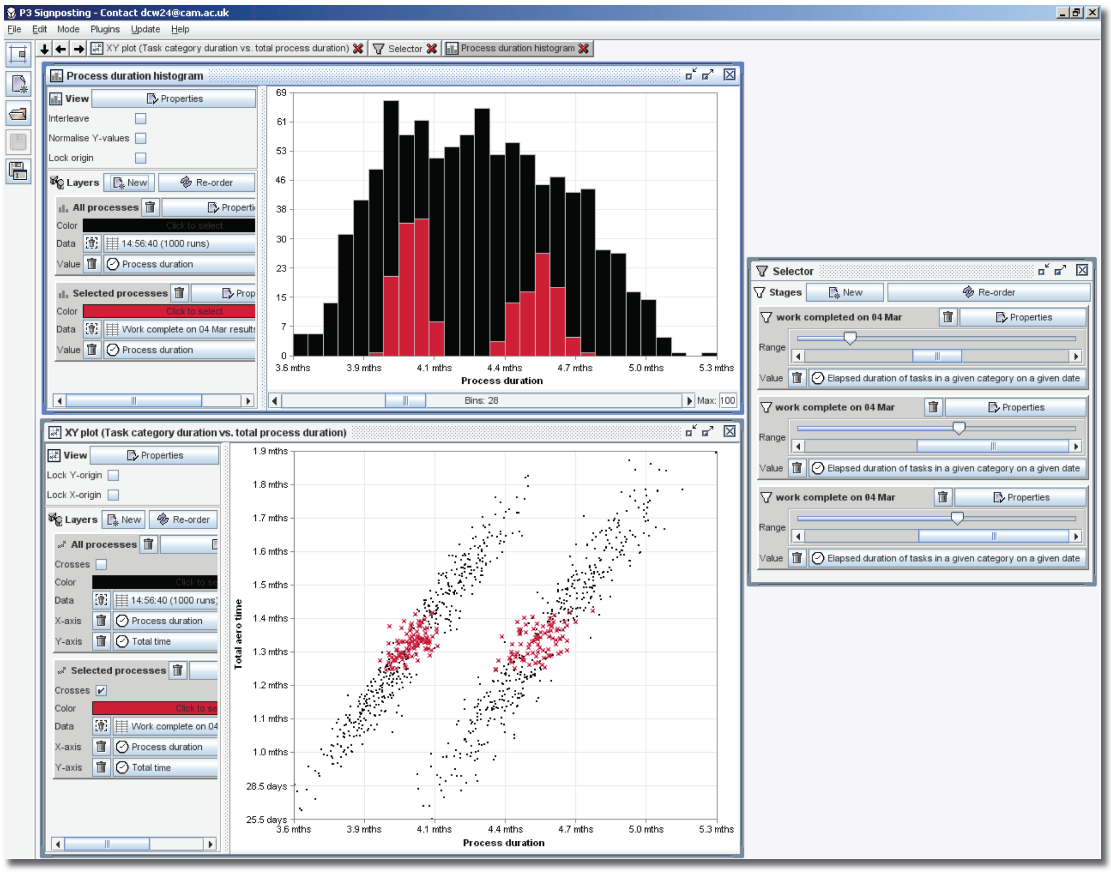


Figure 6.7 A process selector comprising three stages. Two plots show the source data in black and the selector results overlaid in red. When the scrollbars representing stage band-pass filters are manipulated all plots are updated in real time.

capability requires perturbing task durations, re-simulating and re-plotting. This is a non-interactive procedure due to the computational expense of simulation.

6.4.4 Scriptlets

Scriptlets are code fragments which may be attached to individual tasks in an ASM model. They are written in Java and may be developed and compiled from within the P3 Signposting environment. When a task is attempted during simulation any associated Scriptlets are executed in turn. Scriptlets have full access to the system and the model; they may be used to customise task behaviour or execute programs external to the modelling environment. They may also be used to generate the input data required by such codes and parse any output data, thereby synchronising their configurations with the process variables used to control execution routes. This allows external codes to describe the behaviour of tasks in the ASM simulation. Scriptlets therefore allow the ASM simulation

to be extended or integrated into heterogeneous simulation environments.

6.5 Linkage meta-models

The Applied Signposting Model is a general-purpose process modelling framework. It is often appropriate to extend the framework to better suit a particular modelling domain or application. For example, a modelling project might be undertaken to identify the relationships between customer requirements and individual design tasks, with the aim of developing a requirement-centric design process. Capturing this additional information would require extension of the ASM modelling framework and the P3 Signposting software. Other model-based approaches are based on frameworks significantly different to the ASM. For example, Jarratt proposes that a model of the components in a product and their interconnectivity may be used to support the management of engineering change (Jarratt, 2004).

Developing a new or extended modelling approach involves several activities, including:

- Understand the purpose(s) for modelling and the stakeholders in the modelling project.
- Understand the domain to be modelled.
- Develop a modelling framework and support method.
- Develop a software (or paper) implementation allowing construction of models.
- Elicit knowledge from domain experts and develop a coherent perspective.
- Develop a model to address the stated purpose(s).

In practice these activities can involve a great deal of design iteration. For instance, it is common that new requirements which emerge during method development necessitate extensive changes to the modelling framework and software implementation.

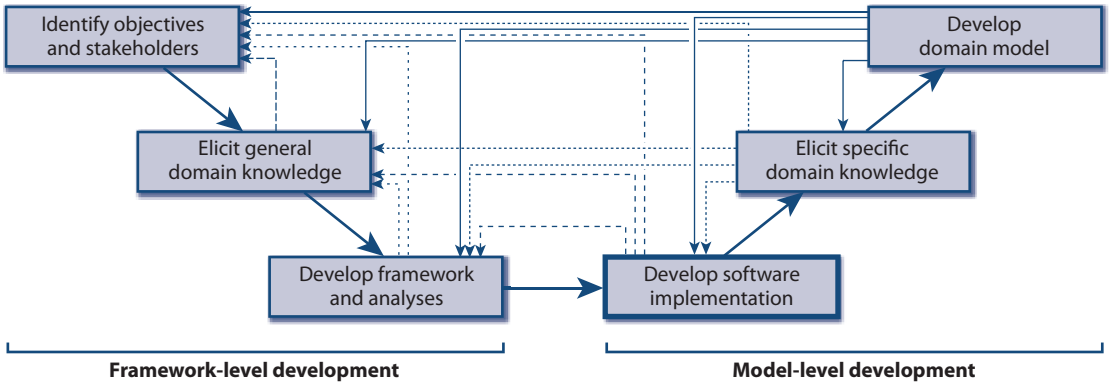


Figure 6.8 Developing a model-based approach to support process improvement can involve many iterations between framework- and model-level activities.

The most time-consuming of these activities is often software development. In particular, while algorithm implementation is usually straightforward, the development of user interfaces which are sufficiently robust and usable for evaluation or distribution outside the academic environment requires specialist skills which many researchers do not possess. As a result, many modelling tools developed in this context never reach sufficient maturity for deployment in industry. The linkage meta-modelling approach introduced below was developed to address this issue by significantly reducing the time and expertise required to develop new modelling tools.

6.5.1 Linkage meta-models in P3 Signposting

A P3 Signposting linkage meta-model is a formal, computable 'model of a linkage model'. Linkage meta-models express modelling frameworks such as the ASM by constraining the elements and relations which are allowed in model instances. They also configure the modelling environment used to manipulate this data.

The linkage meta-modelling approach is based on the observation that many modelling frameworks may be expressed as directed graphs of elements and the linkages between them. Furthermore, most modelling approaches use similar visual representations based on node-link diagrams, matrices, and trees (refer to Chapter 2 for many examples). These views may also be described in digraph form. The data structures underlying model visualisations may therefore be generated from a model instance by topological transformations. This approach forms the basis of a software platform which uses general digraph data structures

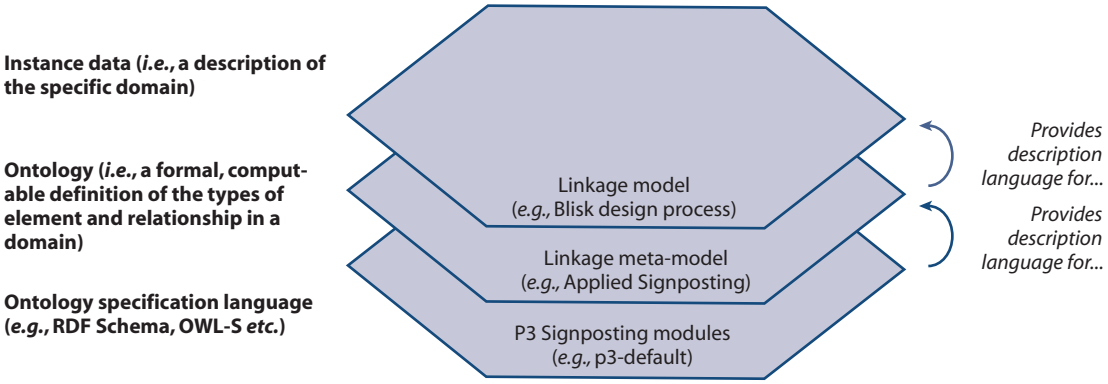


Figure 6.9 The P3 Signposting linkage meta-model and its relationship to formal ontologies.

to represent models constructed using any linkage-based modelling framework and which presents this information in a form suitable for manipulation.

Linkage meta-models are defined using an ontology specification language similar to the *OWL* (Bechhofer *et al.*, 2004) or *RDF-Schema* languages used by ontology modelling tools such as *Protégé*. However, whereas most ontologies are intended for the management of textual data, P3 Signposting linkage meta-models are designed to express linkage-based modelling frameworks in a form suitable for manipulation using network, matrix and tree views.

Linkage meta-models consist of *classes*, *properties*, *relations*, *perspectives* and *views*, all of which are developed using a graphical configurator (Figure 6.10). These are introduced below prior to an illustrative example.

6.5.2 Classes, properties and relations

Classes, properties and relations are defined in a similar way to other approaches which describe the elements allowed in a domain, such as UML and the ontology specification languages discussed above. In brief:

- **Classes** are formal descriptions of the types of element which can be modelled. For example, the Applied Signposting Model defines classes entitled 'Task', 'Parameter' and 'Resource' (and many more — over 26 in total). Classes may be hierarchically defined, such that they inherit properties and relations from their super-classes.

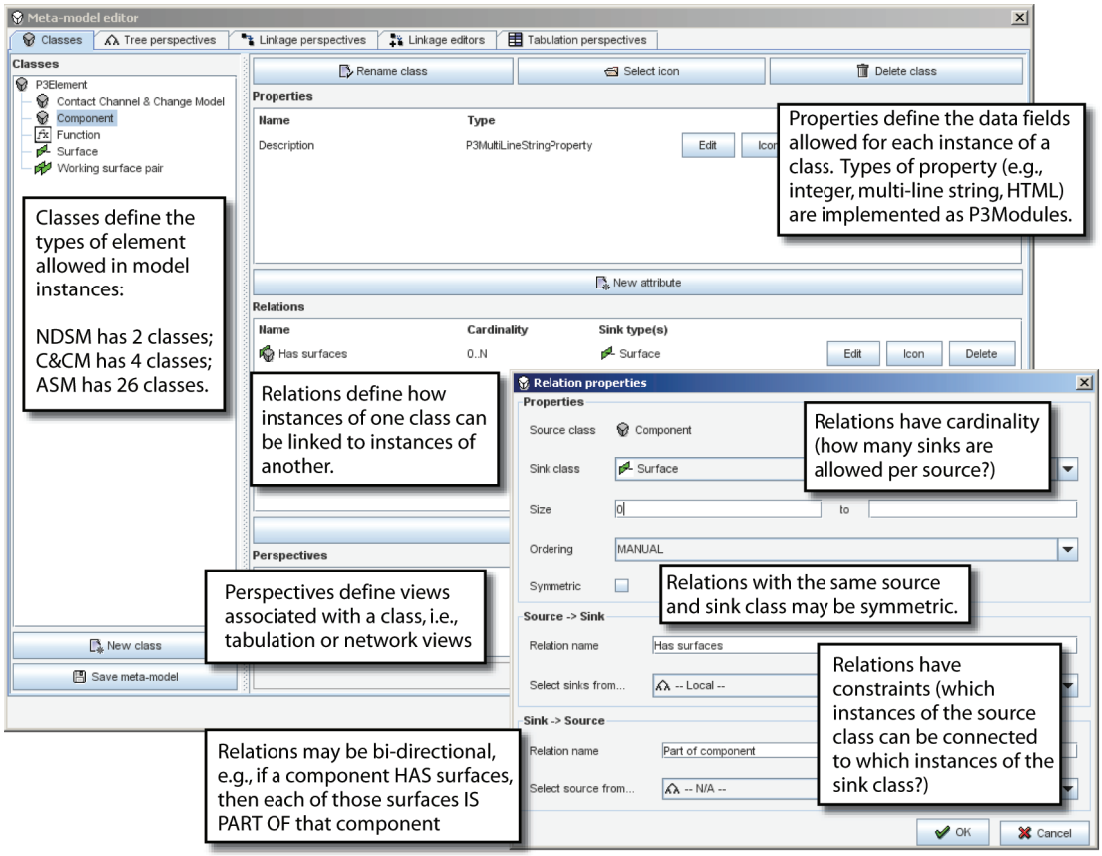


Figure 6.10 The graphical configurator for linkage meta-model development.

- **Properties** define data fields such as 'Name' and 'Description' which qualify the instances of a class. Each property has a type, *e.g.*, *single-line-string*, *boolean*, *etc.* Property types are implemented as modules which are responsible for creating the UI components to manipulate and validate their values.

- **Relations** define how instances of a class can be linked to instances of other classes. Relations have the following attributes:

  - **Source and sink classes.** These must be specified for every relation.
  - **Description.** A relation may be assigned a bi-directional name and icon. For example, if the forward description of a parent-child relation is specified as A *is child of* B, then the reverse description may be specified as B *is parent of* A.
  - **Symmetry.** A relation which links instances A to instances B of the same class may be optionally specified as symmetric. When a symmet-

ric linkage A -> B is added or removed by the user of the modelling software, the converse linkage B -> A is automatically updated to maintain symmetry.

– **Cardinality.** The cardinality of a relation can be specified, *i.e.*, the minimum and maximum numbers of sink elements allowed per source element.

– **Scope.** A relation may have *direct* or *indirect* scope. Direct relations provide slots in each instance of the source class within which new instances of the sink class may be created. Indirect relations connect a source element to sink elements defined elsewhere in the model.

With the exception of a single root element, every element *must* be a sink of exactly one direct relation and *may* be a sink of any number of indirect relations. The root element represents the system being modelled and cannot be a sink of any direct relation. A P3 Signposting model is therefore structured as a tree of elements linked to their children by direct relations. The elements in this tree may be interconnected by a network of indirect relations.

Namespace constraints may be optionally specified for any indirect relation. A namespace constraint determines which instances of the source class may be connected to which instances of the sink class. For instance, ASM *task* elements can only interact with *parameter* elements which are children of the same *process* element, or which are specified as global. If no namespace constraints are specified, any instance of the source class may be connected to any instance of the sink class that exists in the model. Namespace constraints are described using linkage perspectives or tree perspectives, as described below.

6.5.3 Perspectives and views

In general, views of a model cannot be automatically generated from the class definitions because modelling tools usually require multiple visualisations designed to perform specific manipulation tasks. For example, the ASM network visualisation shows only a subset of the model data; resource calendars are not represented

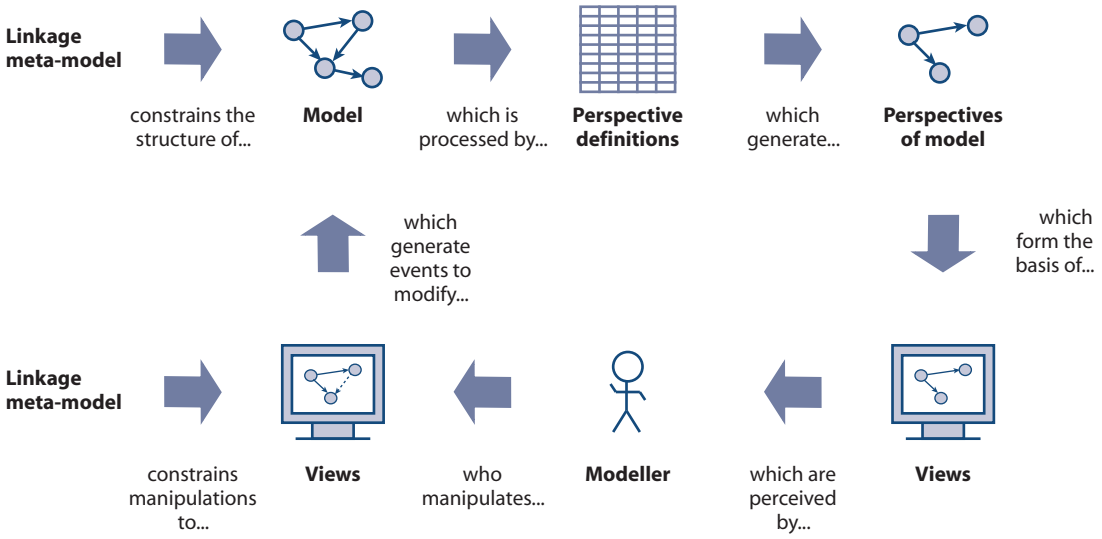


Figure 6.11 Perspectives and views in P3 Signposting.

as this would detract from the clarity of the view. The linkage meta-modelling approach is thus based on the concept that each view provides a *perspective* of the model. Perspectives are implemented as algorithms which process the digraph structure of a model to generate another digraph, usually comprising a subset of the elements and relationships in the model. This data structure forms the basis of a *view*, *i.e.*, an interactive user interface component which is presented to the modeller for manipulation. The relationship between linkage meta-models, models, perspectives and views is depicted in Figure 6.11. This figure also illustrates the high-level operation of P3 Signposting.

6.5.3.1 Perspectives

The algorithms used to generate *tree perspectives* and *linkage perspectives* are the key components of the linkage meta-modelling approach. Each algorithm is applied to a specific element in a model — this is the *context* of the perspective. For instance, either a tree- or linkage perspective may be used to define the namespace constraint for an indirect relation. When the modeller opens a dialog to add an element to that relation, the perspective is applied in the context of the relation's source element to identify the sinks which satisfy the constraint.

The perspective algorithms are implemented as a set of plug-in modules called the *default modules*. The default modules are appropriate for specifying linkage modelling frameworks with relatively simple constraints, as is the case for

Name	Match element	Recurse over relation (*italic font indicates recursion over source elements*; roman font indicates recursion over sink elements)	Create node?	Configurator operations	Operation parameters	Recurse over relation's sinks using perspective	Recurse over relation's sources using perspective
T8. Linkage types	C&CM	Linkage types	Y			T8. Linkage types	
	Linkage type		Y				

```
FUNCTION T8_LinkageTypes ( Element aElement, TreeNode aParent )
    if ( aElement instanceof C&CM )
        TreeNode relationNode = new TreeNode ( aElement.LinkageTypes );
        aParent.addChild ( relationNode );

        for ( Element element : aElement.LinkageTypes.sinks )
            T8_LinkageTypes ( element, relationNode );

    else if ( aElement instanceof LinkageType )
        aParent.addChild ( new TreeNode ( aElement ) );
```

Figure 6.12 An example of a default tree perspective and its interpretation as pseudo-code. In the software, perspective rules are created using a graphical configurator and interpreted each time the perspective is applied.

many matrix-based modelling approaches. Both the tree- and linkage perspective algorithms operate by recursively processing the graph structure of a model instance according to a set of rules which form part of the meta-model definition. Each rule incorporates a pattern which is compared against the model element being examined by the algorithm. Rules may specify an action which must be performed when a match is found (*e.g.*, create a node to represent the element). They may also specify that the algorithm should recurse over any elements linked to the current element via specified relations. The perspective for recursion may be different from that applied to the current element. This allows the treatment of an element to be dependent on the route by which it was reached, which is often necessary to prevent infinite recursions. Figure 6.12 illustrates the definition of a default tree perspective and its interpretation as pseudo-code. Linkage perspectives are defined using the same notation; however, nodes are added to a list instead of a tree as they are created.

Although the flexibility of the default perspectives may initially appear limited, it is in practice possible to create a broad range of useful views by carefully

considering the order in which the model structure is traversed. Additionally, in cases where the default modules are insufficiently expressive it is possible to write customised extensions. The ASM implementation makes extensive use of this approach, creating network, matrix and tree views by processing the model using algorithms which are coded in Java rather than configured using rules.

6.5.3.2 Configurators

In addition to transforming the structure of a model, the perspective algorithms may optionally qualify each node as it is created by modifying the node's *configurator*. Configurators comprise sets of fields which are subsequently parsed by the user interface code to determine how each node should be displayed. For instance, the configurator field FILL_COLOR is used to determine the background color of the node when displayed in a tree view. This field could be set explicitly when the perspective creates each node (*e.g.*, all elements of class *Linkage type* have a yellow background) or bound to a property of type Color, thereby allowing the modeller to specify individual elements' appearances in the interface.

6.5.3.3 Tree views

Tree views are defined as a single tree perspective and a cell renderer, which parses configurator values to determine the color and label of each node in the tree.

6.5.3.4 Tabulation views

Tabulation views are constructed from four components: a source tree perspective defining the column headings; a sink tree perspective defining the row headings; a linkage perspective that is applied in turn to every source element in the tabulation, returning a list comprising each connected sink element together with its configurator that qualifies that linkage; and a *linkage renderer* which determines how cells in the matrix are rendered. For example, the default linkage renderer allows matrix cells to be filled and labelled only. It accepts configurator fields entitled COLOR, LABEL_COLOR, and LABEL, of types Color, Color and String respectively.

Finally, a *linkage factory* may optionally be specified to determine how the model is modified when the user clicks an unpopulated cell in the tabulation. When applied to a specific source and sink element — *i.e.*, the column and row headings of the tabulation cell which was clicked — the linkage factory identifies a chain of elements and relations which could be created to link those elements, in the context of that tabulation. When a chain is then created the view is subsequently updated, the new linkage is identified by the tabulation's linkage perspective and will become visible.

Mouse-clicks on cells which are already populated are not handled by the linkage factory. Instead, a field entitled LINKAGE_ELEMENT is consulted to identify the elements which lie on the path between the source and sink — this is populated by the linkage perspective when the tabulation is created. According to the selection tool in use, the properties of these elements are either displayed for manipulation or the listed elements are deleted to remove the linkage.

6.5.3.5 Model manipulation

Models may be modified by clicking an element in any tree or tabulation to obtain a context pop-up menu. This allows new elements to be added to any direct relations, existing elements to be added to any indirect linkages (subject to constraints), and dialogs to be opened allowing the element's properties to be manipulated. All pop-up menus and dialogs are dynamically generated from the linkage meta-model. When the model changes following a user action, the perspective algorithms are used to incrementally update any open views.

6.5.4 Example configuration

This section illustrates the definition of a linkage meta-model using the *Contact and Channel Model (C&CM)* proposed by Albers *et al.* (2003). This example was chosen because it includes elements in multiple domains, requires consideration of namespace constraints and of indirect linkages. It therefore illustrates key features of the approach while remaining relatively straightforward to explain.

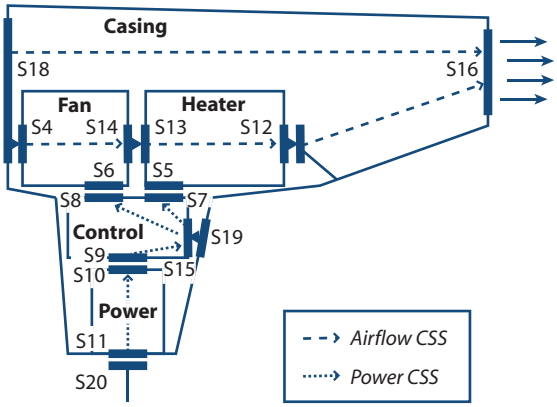


Figure 6.13 Conceptualisation of a hairdryer using the Contact and Channel Model (adapted from Alink, 2005).

6.5.4.1 System conceptualisation

The C&CM is a product modelling framework which conceptualises mechanisms as collections of *Components* which have *Surfaces*. Each surface is linked to one or more surfaces of the same component via a *Channel and Support Structure (CSS)*. For example, the bottom surface of a beam under vertical compression transfers force to the top surface via a CSS. The framework assumes that interactions between components may be described in terms of *Working Surface Pairs (WSPs)* which transfer force, material, energy *etc.* between the surfaces of two adjacent components. An example Contact and Channel Model of a hairdryer is shown in Figure 6.13 (adapted from Alink, 2005).

6.5.4.2 Classes, properties and relations

The *C&CM* linkage meta-model comprises a total of ten classes. Five of these represent the main concepts of the modelling framework: the *C&CM* itself; *Component*; *Surface*; *CSS*; and *WSP*. Since Channel and Support Structures and Working Surface Pairs may be directional, *e.g.*, bulk airflow, or non-directional, *e.g.*, structural support, the CSS and WSP classes are defined as abstract and each has two concrete sub-classes. When creating an instance of one of these classes in the modelling interface, the user is prompted to select one of the two concrete sub-classes. Finally, a *Linkage type* class is provided to allow further qualification of each CSS and WSP in a model.

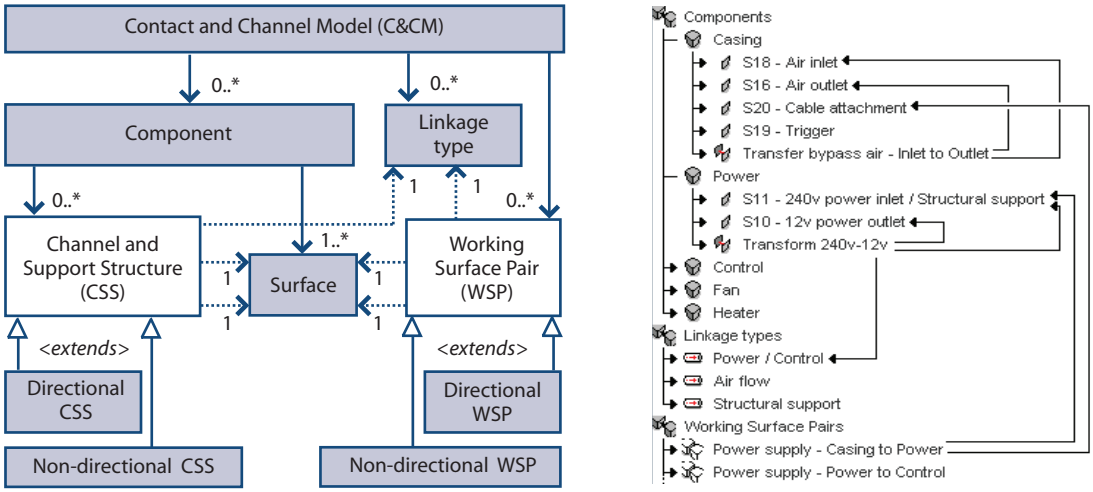


Figure 6.14 A simplified representation of the C&CM framework in a UML-style class diagram (left) and a section from the digraph of the hairdryer model as represented within P3 Signposting (right). Dotted lines in the class diagram indicate indirect relations and solid lines indicate direct relations.

The *Component* class has one direct relation entitled *Surfaces* which may contain 1..* elements of class *Surface* and one direct relation entitled *CSSs* which contains 0..* instances of class *CSS*. The *CSS* class has one direct relation entitled *CSS's surface (1)* and one entitled *CSS's surface (2)*, each containing exactly one instance of the *Surface* class which is constrained for selection from the CSS's parent component's surfaces. Similarly, the *WSP* class comprises two indirect relations which each contain one surface selected from the set of all Components' surfaces. Finally, the *C&CM* class has one direct relation containing 0..* instances of *Component*, one containing 0..* instances of *WSP* and one containing 0..* instances of *Linkage type*.

This configuration allows *C&CM* models to be formally represented as a directed graph. A UML-style class diagram of the framework and the corresponding digraph of the hairdryer model is shown in Figure 6.14.[3] The complete definition is tabulated in Figure 6.17 (p. 181).

[3]Note that alternative configurations may often be used to represent the same system model; as outlined in Section 6.5, identifying the most appropriate representation is an iterative process.

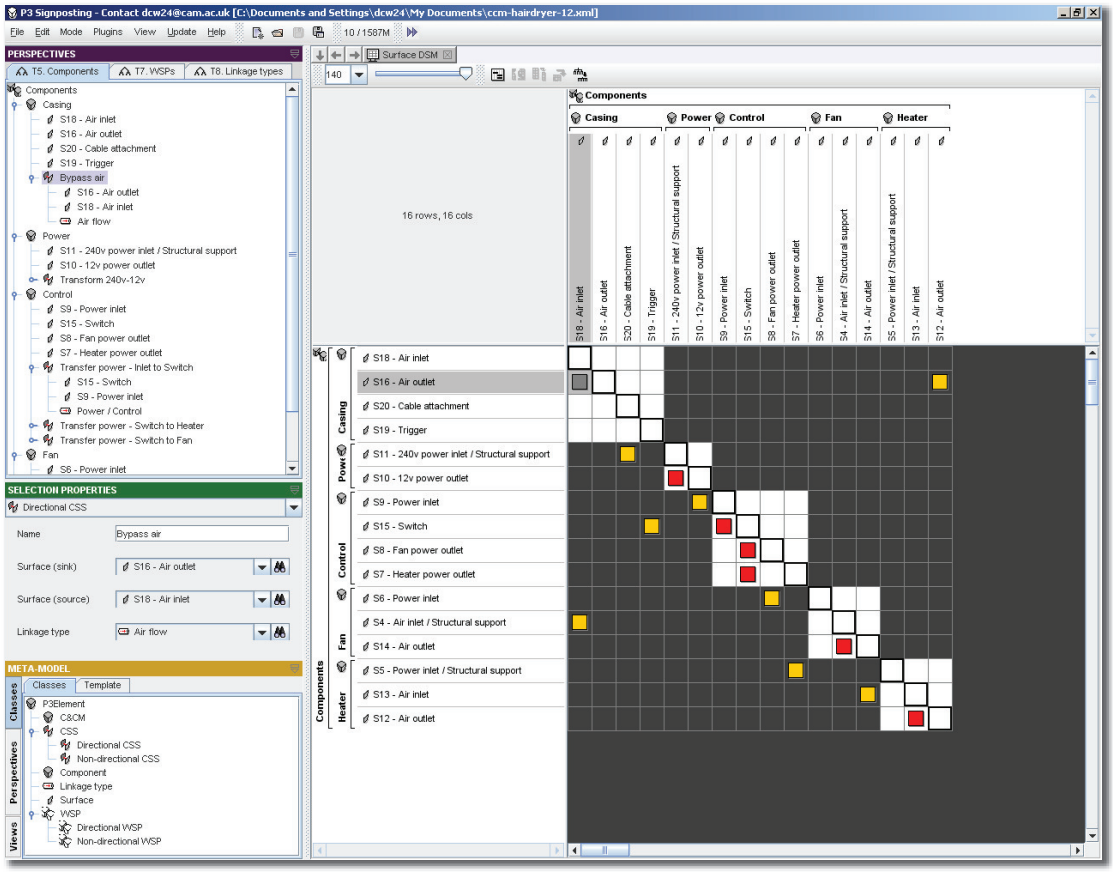


Figure 6.15 The Contact and Channel modelling application developed by configuring a P3 Signposting Linkage Meta-model.

6.5.4.3 Perspectives and views

The C&CM linkage meta-model comprises four views: a tree view indicating the breakdown of the mechanism into Components, their Surfaces and Channel and Support Structures; a tree view comprising the list of Working Surface Pairs; the tree view of linkage types, which may be created by the user and assigned to specific CSSs and WSPs; and a Dependency Structure Matrix which shows the linkages between components' surfaces via CSSs (within components) and WSPs (between components). The definitions of these views are tabulated in Figure 6.18 (p. 181). The perspectives upon which they are based are defined in Figures 6.19, 6.20 and 6.21 (pp. 181–182). Although these definitions are not well suited to representation on paper, they are extremely compact in comparison to the code which would be required to define a modelling application of equivalent functionality using a conventional programming language.

6.5.5 Summary and opportunities for further work

To recap, the linkage meta-modelling approach was developed to allow rapid customisation of the P3 Signposting software to better support particular modelling requirements and thereby reduce the effort required to develop new support approaches. The approach was illustrated by example configuration of the software to develop a C&CM modelling tool. This tool, which is shown in Figure 6.15, was subsequently used by C. Pfeiffer to construct the model of a robot arm mechanism reported in Keller *et al.* (2007).[4]

Linkage meta-models may be used to develop new modelling frameworks, as illustrated by the C&CM example, or may be used to extend the ASM framework for representing information not supported by the standard configuration of Chapter 5. For instance, an additional property entitled 'Priority' might be added to the 'Task' class. This could then be used as the basis of a task selection policy which extended the simulation to account for priority in task scheduling.

At the time of writing, additional implementation work is required to fully exploit the possibilities which arise from the approach. For example, network views are not supported by the current implementation.[5] Another interesting opportunity for further work is to implement the ASM process simulation and other linkage analysis algorithms in a form suitable for application to digraphs generated through perspective algorithms. This would extend the linkage meta-modelling approach to support rapid development of systems analysis tools as well as modelling frameworks.

[4] C. Pfeiffer conducted research in Cambridge as part of his Diploma thesis at Universität Karlsruhe.

[5] Network views could be specified using a single tree perspective to generate the node hierarchy and a linkage perspective to generate the arcs in the network.

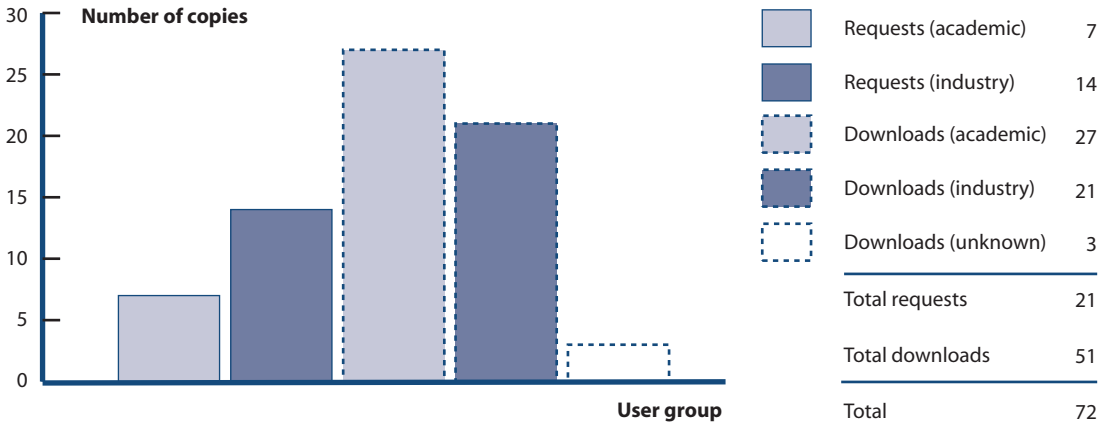


Figure 6.16 72 individuals requested and received copies of the *P3 Signposting* software during the three-month period between 7th September 2006 and 7th December 2006.

6.6 Dissemination

Towards the end of this research project a limited version of P3 Signposting was made available for academic and evaluation use. Seven researchers in the author's group have used or extended the approach, which is also being evaluated by members of the healthcare group in the Cambridge Engineering Design Centre to evaluate its applicability for improving healthcare processes in the NHS (Clarkson *et al.*, 2006). The software has also generated significant interest within Rolls-Royce. In addition to its use by the control account manager following the case study described in Chapter 3, the software has been requested by: a process improvement expert; a knowledge management expert; an intern employed by Rolls-Royce who applied the approach to model the company's DfX processes; and ten members of the turbine design team to support the development of a new global turbine design framework which aims to more efficiently utilise the company's existing computational design and analysis tools.

Additionally, 48 individuals who were not associated with the research project downloaded the software or requested a copy in person during the three month period between 7th September 2006 and 7th December 2006. 24 (50%) of these users were from academic or research institutions, 21 (44%) from industry, and 3 (6%) did not reveal their affiliation.

In total, 72 copies of the P3 Signposting software were disseminated during the first three months of its availability. This is summarised in Figure 6.16.

6.7 Summary

This chapter has discussed the *P3 Signposting* software platform and thereby addressed the second research question outlined in Chapter 1. In summary, the software:

- Implements the *Applied Signposting* process modelling and simulation framework described in Chapter 5, thereby allowing application by other researchers and industry practitioners.
- Provides multiple network- and tabulation-based views of model data and analytical tools to support the exploration of simulation results.
- Supports the development and evaluation of new modelling approaches by enabling rapid development of linkage-based modelling tools.
- Is based on a modular architecture allowing other researchers to extend the software, requiring only limited knowledge of the existing code.
- Has been disseminated following requests from 72 users in industry and academia.

Classes			Properties		Relation properties			Source-sink properties			Sink-source props.	
Name	Extends	Abstract?	Property name	Property type	Source class	Sink class	Cardinality	Name	Namespace constraint	Constraint origin local?	Name	Namespace constraint Constraint origin local?
C&CM					C&CM	Component	0..*	Components				
CSS		Y	Name	String	C&CM	Linkage type	0..*	Linkage types				
- Directional CSS	CSS				C&CM	WSP	0..*	WSPs				
- Non-directional CSS	CSS				CSS	Surface	1	CSS's surface (2)	T2. CSS's surfaces	Y		
Component			Name	String	CSS	Surface	1	CSS's surface (1)	T2 CSS's surfaces	Y		
Linkage type			Name	String	CSS	Linkage type	1	CSS's type	T8. Linkage types			
Surface			Name	String	Component	Surface	1..*	Surfaces				
WSP		Y	Name	String	Component	CSS	0..*	CSSs				
- Directional WSP	WSP				WSP	Surface	1	WSP's surface (2)				
- Non-directional WSP	WSP				WSP	Linkage type	1	WSP's type	T1. Surfaces			
					WSP	Surface	1	WSP's surface (1)	T8. Linkage types			

Figure 6.17 Class, property and relation definitions in the C&CM linkage meta-model.

Tree view name	Generated by tree perspective
T5. Components	T5. Components
T7. WSPs	T7. WSPs
T8. Linkage types	T8. Linkage types

Tabulation	Source tree perspective	Sink tree perspective	Linkage perspective	Linkage factory
Surface DSM	T1. Surfaces	T1. Surfaces	L1. Surface-surface	F1. Surface-surface

Figure 6.18 View definitions in the C&CM linkage meta-model.

Name	Match element	Recurse over relation (*italic font indicates recursion over source elements*; roman font indicates recursion over sink elements)	Create linkage?	Configurator operations (for matched element)	Operation parameters	Recurse over relation's sinks using perspective	Recurse over relation's sources using perspective
L1. Surface-Surface	Surface	*CSS's surface (2)* *CSS's surface (1)* *WSP's surface (2)* *WSP's surface (1)*					*L2. CSS-Surface 2-1* *L3. CSS-Surface 1-2* *L5. WSP-Surface 2-1* *L4. WSP-Surface 1-2*
L2. CSS-Surface 2-1	Non-directional CSS	CSS's surface (1)		set-color bind-element	COLOR, 255-0-0 LINKAGE_ELEMENT	L6. Surface node	
L3. CSS-Surface 1-2	CSS	CSS's surface (2)		set-color bind-element	COLOR, 255-0-0 LINKAGE_ELEMENT	L6. Surface node	
L4. WSP-Surface 1-2	WSP	WSP's surface (2)		set-color bind-element	COLOR, 255-204-0 LINKAGE_ELEMENT	L6. Surface node	
L5. WSP-Surface 2-1	Non-directional WSP	WSP's surface (1)		set-color bind-element	COLOR, 255-204-0 LINKAGE_ELEMENT	L6. Surface node	
L6. Surface node	Surface						

Figure 6.19 Linkage perspective definitions in the C&CM linkage meta-model.

Name	Match element	Recurse over relation (*italic font indicates recursion over source elements*; roman font indicates recursion over sink elements)	Create node?	Configurator operations	Operation parameters	Recurse over relation's sinks using perspective	Recurse over relation's sources using perspective
T1. Surfaces	C&CM	Components	Y			T1. Surfaces	
	Component	Surfaces	Y			T4. Surface node	
T2. CSS's surfaces	CSS	*CSSs*					*T2. CSS's surfaces*
	Component	Surfaces	Y			T4. Surface node	
T3. CSSs	CSS	CSS's type CSS's surface (2) CSS's surface (1)				T8. Linkage types T4. Surface node T4. Surface node	
T4. Surface node	Surface		Y				
T5. Components	C&CM	Components	Y			T5. Components	
	Component	CSSs Surfaces	Y			T3. CSSs T4. Surface node	
T6. Components only	C&CM	Components	Y			T6. Components only	
	Component		Y				
T7. WSPs	C&CM	WSPs	Y			T7. WSPs	
	Surface	*Surfaces*					*T6. Components only*
	WSP	WSP's type WSP's surface (2) WSP's surface (1)	Y			T8. Linkage types T7. WSPs T7. WSPs	
T8. Linkage types	C&CM	Linkage types	Y			T8. Linkage types	
	Linkage type		Y				

Figure 6.20 Tree perspective definitions in the C&CM linkage meta-model.

Name	Match element	Recurse over relation (*italic font indicates recursion over source elements*; roman font indicates recursion over sink elements)	Create linkage?	Recurse over relation's sinks using factory	Recurse over relation's sources using factory
F1. Surface-surface	CSS	CSS's surface (2)	Y		
	Surface	*CSS's surface (1)* *WSP's surface (1)*			*F1. Surface-surface* *F1. Surface-surface*
	WSP	WSP's surface (2)	Y		

Figure 6.21 Linkage factory definitions in the C&CM linkage meta-model.

Chapter 7

Applications and reflection

This chapter discusses application of the Applied Signposting Model and P3 Signposting software tool to support a number of modelling and research projects. This allows evaluation of the research outputs against the success criterion stated in Chapter 1.

Discussion proceeds in three sections. Firstly, three industry-related modelling projects are discussed to highlight application of the approach to support knowledge capture and to validate its utility to industry. Secondly, two Ph.D. projects which have included significant use of the work will be discussed, thereby highlighting the application to process analysis and validating the contribution to support novel research. Thirdly, key lessons learned during these applications are highlighted.

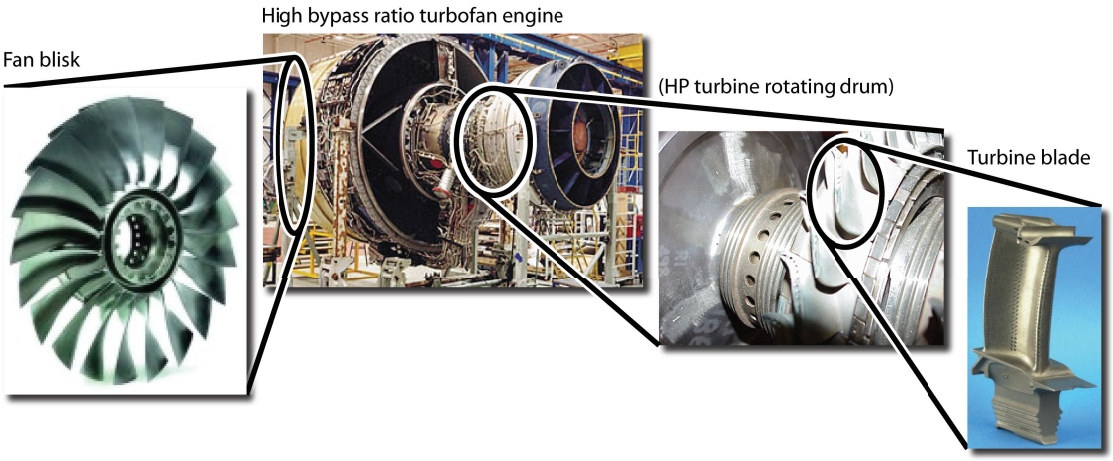


Figure 7.1 The approach was applied to model Rolls-Royce aero-engine design processes at three levels of the sub-system hierarchy.

7.1 Modelling applications

Three modelling projects are presented here as application case studies. These projects were selected for discussion since they concern design processes at different levels of the product hierarchy (Figure 7.1) and since the author was familiar with each case. The work was extensively discussed with the modellers during their projects and in two cases this was supplemented by one-hour informal interviews. The applications are summarised in Figure 7.2 and discussed below.

7.1.1 Turbine cooling system design process

This modelling project was undertaken by a Ph.D. student seconded from the Turbine Systems Engineering group within Rolls-Royce after seven years' employment. He had previously specialised in modelling the fluid dynamics of turbine blade internal cooling systems. The objective of his doctoral research was to improve the company's process for designing these systems.

7.1.1.1 Modelling objectives

At the outset of the Ph.D. project there was a common perception in the turbines group that the cooling system design process was aligned to detailing. In other words, the available tools and the framework in which they were used limited the breadth of concepts that could be explored early in the process (Bell *et al.*, 2007). Bell's Ph.D. research was intended to explore and ultimately address this issue.

Modelling	Fan blisk design I	Fan blisk design II	Turbine cooling design	Engine design
Primary modelling purpose	Scheduling	Visualisation	Simulation	Visualisation
Primary P3 Signposting user	Author	RR design manager	RR designer	Researcher
Modelling period recorded	N/A	N/A	26 days	54 days
of which continuous usage	N/A	N/A	39 hours	39 hours

Model structure	Fan blisk design I	Fan blisk design II	Turbine cooling design	Engine design
Process definitions	16	17	44	69
Task definitions	73	79	155	430
Parameter definitions	59	36	269	501
Data Interaction definitions	22	1	217	344
Flow Interaction definitions	217	249	523	1038
Resource definitions	3	0	0	29
Task tree nodes	99	100	314	430

Size ratios	Fan blisk design I	Fan blisk design II	Turbine cooling design	Engine design
Mean tasks per process	4.56	4.65	3.52	6.23
of which simple tasks	54%	63%	57%	76%
of which compound tasks	0%	8%	1%	1%
of which iteration constructs	9%	0%	7%	6%
of which containers	35%	27%	32%	15%
Mean usages of each task	1.36	1.27	2.03	1
Mean interactions per task	3.27	3.16	4.77	3.21
of which data interactions	9%	~ 0%	29%	24%
Mean interactions per parameter	4.05	6.94	2.75	2.76

Figure 7.2 A summary of the modelling projects presented as evaluation case studies. 'Fan blisk design 1' refers to the model constructed by the author during the method development reported in Chapter 4.

As part of this research, an ASM process model of the cooling system design process was constructed to develop the overview of tasks and information flows necessary to identify potential improvements. It was intended that simulation could ultimately be used to support verification of this model and allow exploration of alternative process configurations.

7.1.1.2 Process characteristics

The turbine operating temperature is a primary limit on the efficiency and power output of a jet engine. At $1200 - 1300\,^{\circ}\mathrm{C}$ this is significantly higher than the melting point of the alloy from which blades are manufactured. Heat is removed by cooler air which flows through passages within each blade prior to exiting through small surface holes and forming protective exterior films (Figure 7.3).

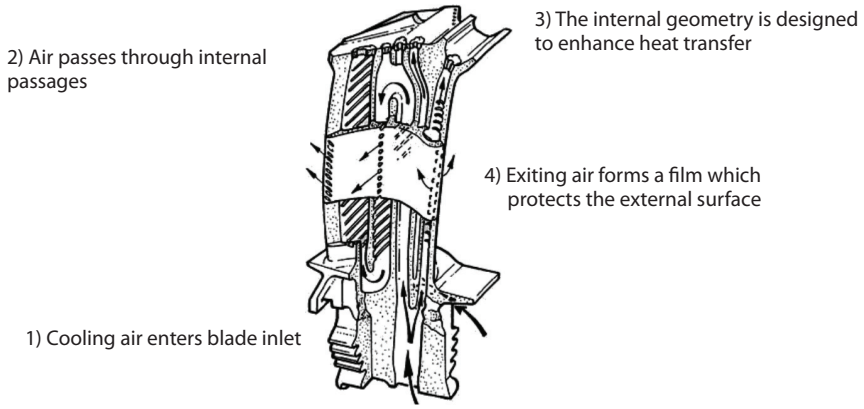


Figure 7.3 A cut-away view of a turbine blade which illustrates the internal cooling passages (Bell *et al.*, 2007).

The aerothermal and mechanical design of turbine blades is a difficult task. External and internal features interact in a coupled and nonlinear fashion to determine the performance of the cooling system and, ultimately, the turbine as a whole. These complex physics require that most design effort is based on detailed models. Although important design decisions are made prior to this phase, relatively few concepts are generated and evaluated due to the expense of modelling and the limited utility of low-cost approximations.

7.1.1.3 Modelling methods

To supplement his prior experience Bell conducted interviews with 18 stakeholders in the cooling system design process. These included cooling and aerodynamics experts, stress engineers, manufacturing engineers and turbine blade designers.

7.1.1.4 Model structure

The detailed process model is shown in Figure 7.4. Its structure is essentially sequential and incorporates many possibilities for rework. Extensive use is made of data interactions to capture the strong connectivity between tasks while maintaining a readable diagram. A key sub-process entitled 'Section design and analysis' is used in six locations. This is visible as a repeated pattern in Figure 7.4.

Development of this model allowed Bell to identify those aspects of the design process for which improvement would provide the greatest overall benefit. He subsequently used this knowledge to develop a new conceptual design process.

Figure 7.4 An overview of the cooling system design process model developed by C. P. Bell.

7.1.2 Fan blisk design process

The P3 Signposting software was given to a design manager responsible for timely delivery of fan blisks. This individual was familiar with the research in the context of the support method discussed in Chapter 4. Following conclusion of that study, he independently applied the approach to re-model the blisk design process.

7.1.2.1 Modelling objectives

The primary modelling objective was to develop a comprehensive description of the blisk design process to support improvements to the 'Fan Key System', a suite of analysis codes used extensively throughout the design process. A detailed model capturing the design tasks and parameters was constructed using a significantly different structure to that reported in Chapter 4. Modelling was conducted entirely by the design manager; apart from about 5 telephone conversations the author was not involved.[1]

7.1.2.2 Model structure

The resulting model is shown in Figure 7.5. It follows a similar structure to that used by McMahon and Xianyi to represent the design process as concurrent work streams that are integrated in a process of iterative refinement (McMahon and Xianyi, 1996). To illustrate, consider the top-most sub-process of Figure 7.5. Initial data preparation tasks (A1-A4) lead to four concurrent work streams, represented as processes that are independent during execution (B1-B4). Upon completion of these streams a compound task representing a co-ordination meeting is undertaken (D). This task has four output scenarios representing the result of a decision taken during the meeting regarding the focus of design and analysis for the next iteration, and one to indicate process completion. The first four scenarios lead to tasks which modify the design definition to incorporate recent advances (E1-E4).

A similar structure is exhibited by the lower sub-process, which represents iterative development of the aero-mechanical design into a more detailed scheme prior to tooling. Processes A1, A2, A3 and B1 are re-used as part of this sub-process.

[1] Since the conversations were not recorded and no records were kept it is not possible to be more specific.

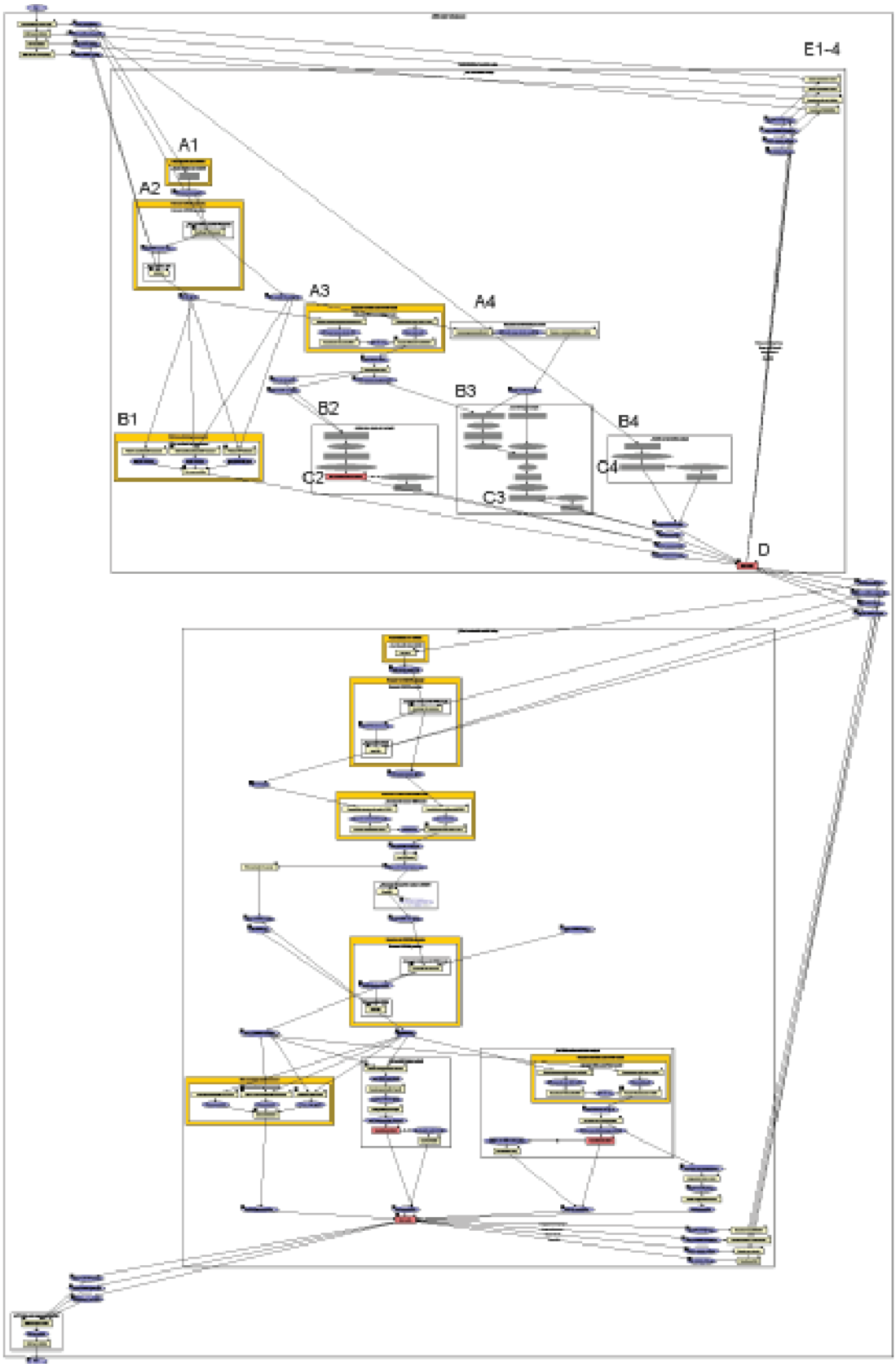


Figure 7.5 An overview of the fan blisk design process model developed by the blisk design manager. In this visualisation, the orange container task borders are only rendered in cases where the contained sub-process is used in multiple contexts.

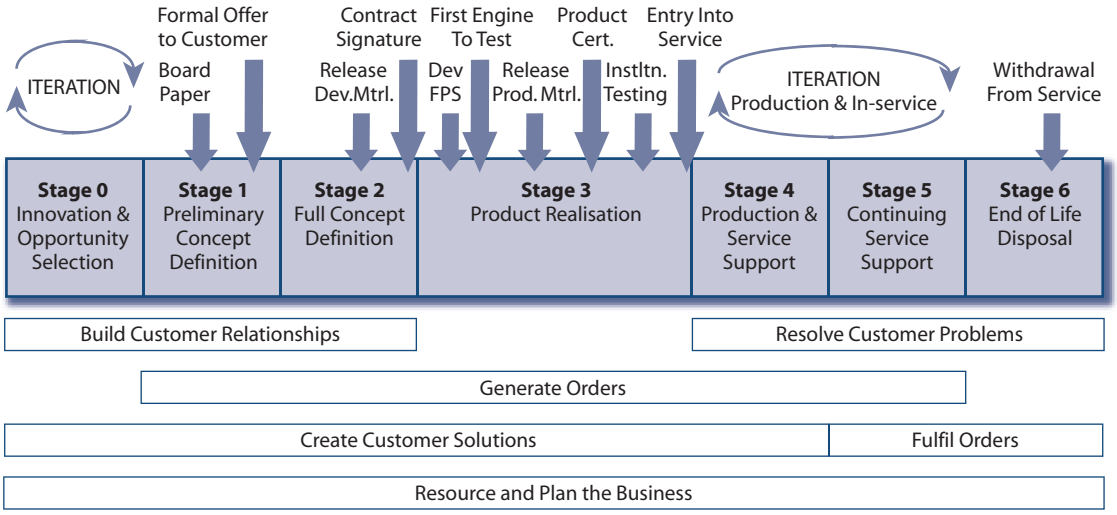


Figure 7.6 The Rolls-Royce 'Derwent' product introduction and lifecycle management process (Rolls-Royce, 2005).

7.1.3 Whole engine development process

P3 Signposting forms a core component of a Rolls-Royce/DTI funded research project entitled *Integrated Products and Services (IPAS)*. As part of this project, a post-doctoral Research Associate was employed at Cambridge University to develop an ASM model of the whole engine development process.

7.1.3.1 Modelling objectives

The modelling exercise aimed to develop a detailed description of the whole engine design process with a particular focus on capturing the consideration of service information. The ultimate aim of the project is to prescribe an improved process which takes better account of serviceability goals; however, at the time of writing it was not clear how this would be achieved. The detailed modelling exercise focused on the early phase design work for the turbine components and on the Trent 1000 design project.

7.1.3.2 Process characteristics

Rolls-Royce have published a stage-based model which describes their process for product introduction and lifecycle management. This *Derwent process* comprises six stages and covers a total timespan of up to 50 years (Rolls-Royce, 2005). Modelling covered stages 0–3 with a primary focus on stages 0 and 1.

In overview, aero-engines are developed by finalising high-level parameters such as basic geometry and numbers of blade rows during early concept design. This determines the requirements for the engine sub-systems and components which are designed by Integrated Product Teams (IPTs). IPTs consist of personnel from the functional areas which hold a stake in the component. Individuals may be permanently assigned to an IPT or may be part of many IPTs depending on their roles. Additional IPTs are responsible for design activities which encompass several components, such as systems integration and air systems.

7.1.3.3 Modelling methods

The IPAS researcher conducted 32 interviews with 27 designers from the Advanced Propulsion Systems Design (responsible for conceptual design) and Turbine Systems Engineering groups. Based upon these interviews, he developed an initial process model which is currently undergoing a second phase of verification interviews conducted by another IPAS researcher in the Cambridge Engineering Design Centre.

7.1.3.4 Model structure

The resulting model is shown in Figure 7.7. The focus on high-level product detail is reflected in the representation of individual tasks and parameters. For example, the turbine cooling system design process was represented using 314 tasks in the dedicated model of Section 7.1.1, but by a single task in the whole engine model. The model is unique among the case studies since it incorporates many possibilities for task concurrency. These arise from the low-level network structure as parallel work streams are not explicitly incorporated in the model.

The model was considered to be significantly less complete and less consistent than the three component-level design process models discussed above.[2] For example, most tasks were not distinguished from the information they generated, usually producing a single parameter of the same name.

[2]This conclusion was drawn from discussions between the author and other members of the research group, all of whom were familiar with the ASM and conversant with Rolls-Royce design processes.

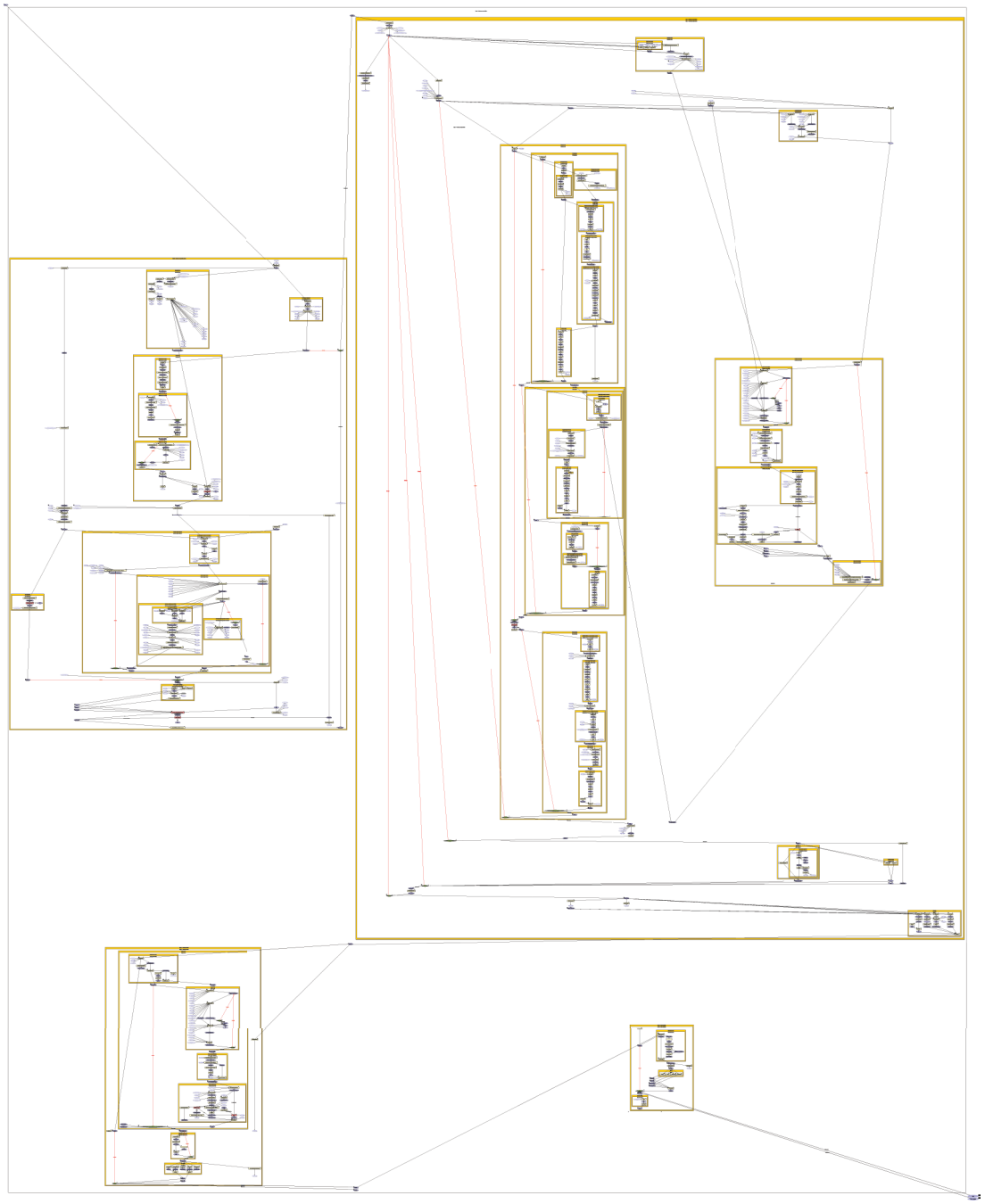


Figure 7.7 An overview of the whole engine design process model developed by various researchers in the author's group.

One contributing factor may have been the approach to knowledge elicitation. At the outset of the IPAS study the modeller was unfamiliar with the domain — having no prior knowledge of the product or company; a relatively small number of interviews were conducted — less than 35 hours in total; in contrast to the other applications the model was not presented to stimulate discussion and refinement during interviews; and few personnel were interviewed more than once.

The process to be modelled was also extremely complex. As discussed in Chapter 3, even experienced company personnel do not claim a detailed understanding of the entire design process. In addition, certain characteristics of the process proved difficult to model using the task precedence representation. For instance, the IPAS researcher indicated that the opportunistic and responsive nature of many design activities could not easily be modelled — this is a recognised limitation of the ASM which arises from its task precedence representation. He also identified that integrative functions such as air systems could not be easily represented in the main workflow.

7.2 Research applications

This section describes two research-oriented applications of the ASM and of P3 Signposting. The applications apply and extend the tool to investigate how simulation analysis can be used to improve process performance. They were undertaken by two Cambridge students as part of their Ph.D. research.

7.2.1 Optimising design strategies

Following development of the cooling system design process model discussed above, Bell *et al.* (2007) proposed an approach to improve component design processes based on the ASM framework. As discussed in Chapter 3, these processes consist of many *convergence/refinement* cycles. On each cycle, decisions are made to determine whether inexpensive low-fidelity analyses will be undertaken or more lengthy and higher fidelity tasks will be used. One goal of the research project was to optimise these decisions, *i.e.*, to determine the most appropriate strategy for task selection at each stage in the design process.

The approach is based on the assumption that *process meta-data* influences the selection of tasks when designing, and that representing this meta-data can enhance the fidelity of simulation models. In this context, meta-data refers to data about the process, such as the number of times a task has been attempted, the confidence in the design, the time remaining, *etc.* An ASM simulation model was constructed based on the case study of the cooling system design process discussed above. This simplified model used process variables to represent meta-

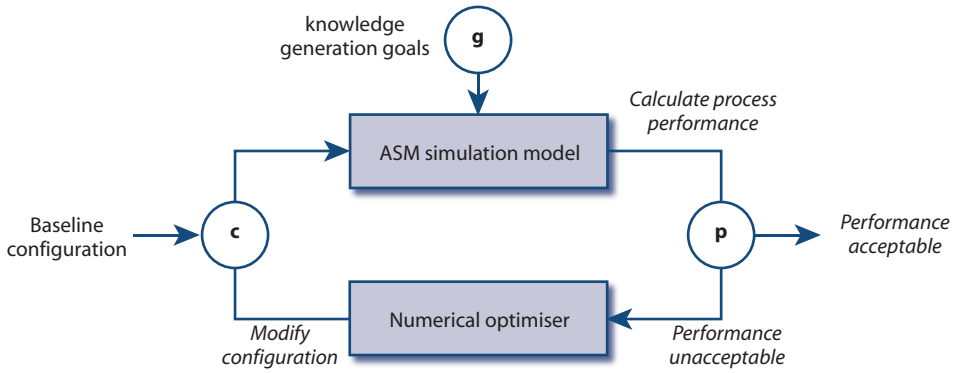


Figure 7.8 Bell *et al.* (2007) propose that the ASM simulation could be used as the basis of a process configuration optimiser.

data and process variable functions to model the effect of each design task in terms of the meta-data. Task selection decisions were modelled by evaluating the meta-data against a set of process variables which encode the point at which design moves from low- to high-fidelity analyses. Process variables were also used to define design maturity goals (iterations must continue until a specified amount of knowledge is generated) and measure process performance in terms of total cost and duration.

The resulting ASM simulation model may be viewed as a 'black box' for predicting the process performance of a given strategy and design maturity goal (Figure 7.8). Bell *et al.* propose this could be used as the basis for a numerical process optimisation implemented as a *P3 Signposting* plug-in (Bell *et al.*, 2007).

7.2.2 Identifying robust processes

Chalupnik *et al.* (2007) used the ASM simulation to investigate robustness in design processes. The ultimate goal of this research is to identify process configurations which can deliver expected performance in an uncertain environment.

A *P3 Signposting* plug-in was developed to explore the process robustness concept. The plug-in utilised the ASM simulation code to conduct one-factor-at-a-time perturbation analyses. A laboratory experiment was then conducted to explore the properties of the 12-task mechanical design process previously modelled by Clarkson *et al.* (2000). This led to a number of general insights regarding process robustness analysis. The research is published in Chalupnik *et al.* (2007).

7.3 Reflection

This section briefly reflects upon application of the approach under four headings: lessons learned during modelling; cost of modelling; perceived strengths; and model re-use.

7.3.1 Lessons learned during modelling

Modelling approaches used during the case studies included recorded interviews which were subsequently used to identify key activities and sequences, group workshops in which the model was presented and discussed, and informal discussions with individuals or small groups in which print-outs of the model were reviewed and annotated. Of these approaches, the small-group discussions were found to be the most effective. Workshops were found to be less productive due to difficulties in maintaining focus. Interviews in which the model was not reviewed were helpful to improve the interviewer's understanding of the domain, but did not allow experts to critique and suggest improvements to the representation.

Each of the models described above changed substantially as its structure and purpose was explored during development. In each case, construction followed a highly iterative process of critique and refinement, with the modeller's understanding of the process developing in parallel with their model. Although each case had a well-defined process improvement goal to guide initial modelling, additional uses for the models suggested themselves during construction. This led to refinements of the structure, nomenclature and perceived fidelity of the models. It was further observed that modelling did not proceed through incremental improvements. In each case leaps in understanding led to substantial restructuring and upon some occasions to the development of a new model. This iterative process may be as characteristic of process modelling as it is of designing, and highlights the importance of effective user interfaces to facilitate model manipulation.

The studies have highlighted that modelling processes — even those which are considered well understood and well documented — is an effort- as well as knowledge-intensive activity. This is the case even for an insider provided with appropriate

tools. The difficulty of process modelling was usually underestimated, as highlighted by a comment from one Rolls-Royce manager:

> *"If you understand your process already it shouldn't take long."* — Design manager.

7.3.2 Perceived strengths

In each study, the approach has enabled development of models exhibiting significantly more detail than was shown in previous process descriptions. Furthermore, users have indicated a desire to continue adding detail in terms of both breadth and depth. This highlights that a key strength of the approach is the hierarchical structure and user interfaces which allow development of very large models:

> *"You could capture down to a very low level what was going on, and yet package all that detail away. That was actually important, because in order to understand the detail you had to look at the interactions between sub-processes. How the tasks at the bottom of those loop nests interacted was not as people perceived it to be — as you talked to people in the chain there was no one picture of how all the bits fitted together."* — C. P. Bell.

Another perceived strength of the approach is its intuitive, graphical notation. Although the ASM contains many features to allow development of sophisticated models, interpreting the graphical notation does not require a full appreciation of the modelling framework. This allowed models to be discussed with process participants who could easily interpret the information and thereby offer suggestions for improvement. In addition, the formality of the notation was thought particularly useful in developing a consistent representation of a complex and subjective domain. The blisk design manager suggested this was particularly useful, not just for describing an existing process but to support the development of new processes:

> *"I actually view [P3 Signposting] as an ideal tool for doing design process design."* — Blisk design manager.

7.3.3 Cost of modelling

A more objective analysis of the modelling activity was undertaken by configuring the software to log each action performed by the user, together with the time at which they were executed. Only those actions leading to modification of model data were logged; no record is available of the users' navigation within the tool. Furthermore, only those sessions which included data saved to disk were stored.

Detailed logs were captured for the construction of three models — the whole engine design process model (Dataset 1), the turbine cooling system design process model (Dataset 2), and a model of the decision process for accommodating design changes at the interface between software and hardware in the control systems group at Rolls-Royce (Dataset 3). The latter model was constructed by M. Kilpinen, a Ph.D. student in the author's research group. It comprises 122 tasks and 386 parameters.

7.3.3.1 Effort expended in modelling

A 54 day period of the whole-engine modelling was captured in Dataset 1, comprising 39 hours of continuous activity and 3912 individual modelling actions. Dataset 2 represents a 26 day period of the cooling system modelling, comprising 3010 actions in 39 continuous hours. Dataset 3 comprises 1715 commands in 27.5 continuous hours over 31 days.[3]

This supports the informal observation that process modelling is a time-consuming activity. A significant time period was spent using the software, at a mean activity rate of one activity every 36 seconds, 47 seconds and 58 seconds for experiments 1, 2 and 3 respectively. Discussions with each user indicated that this 'active' modelling time represented a small proportion of the overall activity; a significant period of time was also spent reflecting on how the model should be structured to most effectively represent the knowledge modellers had gained during data gathering. Users believed they had developed a much deeper understanding of the design process and its behaviour as a result of this activity.

[3]The continuous activity period was calculated by subtracting all periods of inactivity greater than 30 minutes from the total recorded time. Since only modification actions were recorded, it is not possible to determine whether the tool was being used between logged commands. Therefore, time spent navigating the model or running simulations cannot be distinguished from inactivity.

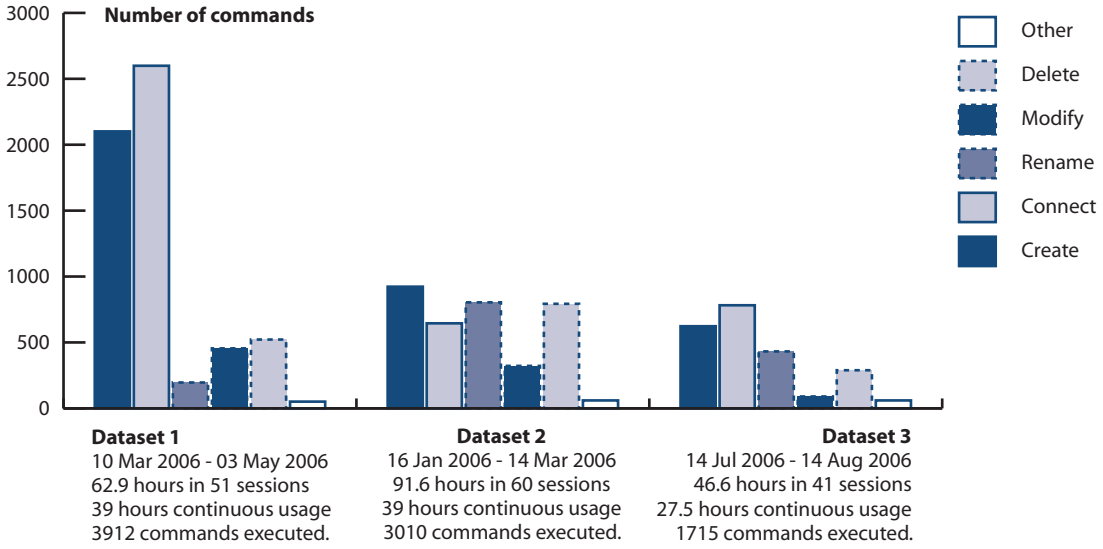


Figure 7.9 Summary of the numbers and types of commands executed during the monitoring periods. Note that some commands are counted in multiple categories, *e.g.*, the single drag-and-drop action which creates a new interaction and connects it as output from a task is categorised as both *create* and *connect*.

7.3.3.2 Profile of modelling activities

Figure 7.9 indicates the number of commands of each type executed by each user. Although there are insufficient samples to draw statistically significant conclusions across the datasets, the large proportion of *modify*, *rename* and *connect* tasks visible in all three cases highlights that a substantial portion of the modelling effort was expended in modifying existing information. This supports the informal observation that process modelling is a rework-intensive activity.

The data also show that the fractions of *Create* and *Connect* commands in Dataset 1 differ from those in the other two datasets. Dataset 1 showed these action types accounting for almost 80% of the total, whereas Datasets 2 and 3 indicated that only 44% and 51% of actions respectively involved these types. One contributing factor is that the IPAS researcher chose to start again with a new model on two occasions, whereas the other users continually refined one model over the entire recorded period. When questioned, the IPAS researcher indicated that he started again due to the difficulty of reconfiguring the hierarchical structures he had constructed. This highlights the importance of an effective user interface which supports the reconfiguration of models.

7.3.4 Flexibility of models and process elements

The flexibility of any modelling framework to represent different processes may be considered from various perspectives. From a framework perspective, many models may be constructed using a particular approach, each of which may represent different processes for different purposes. A particular model may also be flexible on an interpretation level – in that models mean different things to different people – and on the level of the target system. The latter refers to the ability of a model to be instance-independent. For example, the process models discussed in Section 7.1 above are both specific and generic representations. They provide a description of a specific instance of the design process, but also have potential to represent aspects of future or similar processes.

This flexibility raises the possibility of developing a process library which can represent not only similar aspects within one model but from which many models may be composed. Such a library could reduce the effort required to construct process models and thus facilitate application of the research in industry. Elsewhere, this strategy has been successful in the ADePT (Austin *et al.*, 1999) and ProModeller (Vajna, 2005) tools applied to the construction and automotive industries respectively.

However, experience with Rolls-Royce has indicated that while high-level task descriptions might change little from project to project, details can vary significantly. In terms of products, some components or systems are adapted and carried across, or modified on a parametric level, while others require more innovative solutions. In terms of processes, Case Studies 1–3 summarised in Figure 7.2 focused on modelling specialised technical design processes, each of which is conducted using specific tools, analyses and solution principles. Sub-processes are not recognisably similar across these models. Although in each case company personnel indicated that their processes remain similar across product generations, there is insufficient evidence to determine whether a generic library would have provided an effective starting point for developing models in these studies.

Nevertheless, in every modelling application process elements were re-used within the model. This experience indicated that successful re-use hinges on achieving an appropriate form and level of abstraction. In particular, interface definitions must be carefully considered to ensure that the exposed parameters are appropriate to the process and to all contexts in which it is used. For example, the same process cannot represent activities at the assess customer needs level and the parametric optimisation level, since the abstraction of interface parameters must be significantly different in each case. The ability to highlight such interface mismatches was generally considered a benefit of the ASM since it encouraged development of well-structured and self-consistent models. However, the need for interface consistency also limits the possibilities for process re-use, since processes are inevitably tailored to fit their original contexts. In practice, several iterations of a process and its interface definition were found necessary before re-use was possible.

In summary, each modeller who applied the approach believed that the reusability of process elements was a useful feature of the modelling framework. Although this was originally intended to reduce the cost of modelling, the greatest benefit ultimately arose from the need to carefully consider the context and nomenclature of revisited processes.

7.4 Summary

This chapter has described application of the Applied Signposting Model and P3 Signposting software to a number of modelling and research projects. To summarise:

- Application of the modelling approach and software tool to develop three large process models was discussed. Two of these are ongoing research projects which have included significant industry interaction; the third was undertaken entirely by an industry user.

- The research contribution of this thesis is validated by the acceptance of the modelling approach in these studies. This acceptance is indicated by the substantial effort each modeller invested in applying the approach. In each case, the approach allowed the modeller to develop significantly more detailed models than were previously available.

- Two research-oriented applications were discussed, in which the modelling approach and software tool were extended by other researchers.

- It was demonstrated that the research contribution can be applied to support knowledge capture and process analysis, in addition to the management support application introduced in Chapter 4.

—— This page is intentionally blank ——

Chapter 8

Conclusions

This chapter concludes the thesis. The research questions introduced in Chapter 1 are revisited and the research contributions summarised. The contributions are evaluated against the success criterion stated in Chapter 1. Limitations are discussed and opportunities for further work highlighted.

8.1 Review of research contributions

This section reviews the research contributions under two headings: answers to the research questions and additional contributions.

8.1.1 Research questions

To recap, this dissertation argues that:

> The processes by which complex products are developed may be modelled using a formal approach. Such models may be used to support a range of process improvement activities.

Chapter 1 decomposed this argument into two research questions. Firstly:

- What attributes are necessary in a design process modelling approach used for knowledge capture, management support and process analysis?

This question was addressed by developing the *Applied Signposting Model* based on an extended case study in Rolls-Royce. In summary, the key attributes of the ASM are:

1. **Task precedence representation.** This was identified as an effective approach to modelling component design processes. Distinguishing between the role of information that drives task selection and that which drives task execution allows representation of more complex dependency relationships than can be easily incorporated in most task precedence models, thereby addressing a key limitation of these approaches.

2. **Rich graphical notation.** Individuals do not possess both the overview and detailed knowledge required to construct a useful process model. A key challenge during modelling is therefore the negotiation of a common perspective. This is supported by an intuitive, graphical notation. Further, a rich representation which enforces logical constraints provides a 'common language' for modelling, supporting communication and therefore validation of models.

3. **Hierarchical structures.** The research revealed that very large models are often required to describe processes in industry. The ASM provides hierarchical structures to support manipulation of large volumes of process information.

4. **Flexible specification of simulation behaviour**. A flexible approach allows simple graphical simulation models to be developed using the same notation as those exhibiting more sophisticated behaviour. This is useful to support research in particular, as it allows investigation of a wide range of models based on a single, familiar notation.

The second research question was:

- What are the requirements for design process modelling and simulation software to support academic research and industry practice?

This question was addressed by developing and validating the *P3 Signposting* software described in Chapter 6. To summarise, the key requirements are:

1. **Incorporating the benefits of popular descriptive tools**. Familiar and intuitive visualisations should be provided to encourage use alongside existing tools.

2. **Developing a knowledge repository.** Process modelling initiatives are often criticised in industry for their expense and lack of relevance outside the original context. To address this concern, tools should support development of a knowledge base which may be re-used for future modelling and continuous improvement projects. Furthermore, since models are inevitably refined in application tools should provide for incremental development and maintenance of the knowledge base.

3. **Querying the knowledge base.** Process modelling can quickly result in a large volume of potentially useful information. In addition to facilitating the entry of this data, modelling frameworks should provide structures for annotation and tools must provide functionality for effectively browsing and searching the knowledge base. Due to the large number of variables and potentially complex behaviour of simulation models, analysis tools should support the identification of 'levers' for process improvement.

4. **Configurability.** Concurrent development of methods, theories and tools with regular feedback from users is an effective approach to developing model-based approaches for process improvement. Software development can play a key role in guiding the research direction as well as supporting continuous evaluation and refinement. However, it is a time-consuming activity requiring expertise which many researchers do not possess. A configurable software platform can reduce the cost of developing research prototypes (Chapter 6).

8.1.2 Additional contributions

The following additional contributions were made by this research project:

1. **A framework to organise models of product development.** Modelling the design and development process is a well-established research area. Although a number of reviews thoroughly examine aspects of this field, no recent publications frame the breadth of literature. A framework which begins to address this was presented in Chapter 2.

2. **An analysis of modelling design iteration.** Iteration is a major driver of uncertainty in the design process. The analysis of design iteration presented in Chapter 3 illustrated that task network models cannot reflect the full complexity of iterative behaviour as observed in practice. It was concluded that studying the characteristics of iteration in a given design domain and the modelling objectives can guide selection of an appropriate modelling approach.

3. **A model-based approach to support project management.** Chapter 4 introduced an approach to support the management of design programmes using process modelling and simulation. This approach illustrates that design process simulation can support practice despite its limited fidelity.

4. **Software to support process improvement.** P3 Signposting is a robust and extensible platform which implements the Applied Signposting approach and facilitates the rapid development of new linkage-based modelling frameworks. This provides a significant contribution to research by lowering the barrier to developing new model-based approaches.

5. **A contribution to Rolls-Royce.** Modelling the blisk design process provided benefit to Rolls-Royce by integrating existing documentation into a rigorous and significantly more detailed description (Chapter 4). The blisk design manager who supported the study chose to continue this work by independently applying the modelling software following its conclusion. In addition to this direct contribution, many Rolls-Royce personnel have encountered the research results during application of P3 Signposting by other researchers (Chapter 7).

8.2 Evaluation

This section shows that the research project met all objectives stated in Chapter 1 and validates the thesis against the original success criterion.

8.2.1 Applications of process modelling

Chapter 1 identified the three process improvement applications considered by this thesis. To recap, these were: knowledge capture; management support; and process analysis and reconfiguration. The discussion of applications in this dissertation focused on management support as this was the original context for the research project. Chapter 7 illustrated that the modelling framework and software tool have been successfully applied to support knowledge capture and process analysis, thereby indicating their utility for all three applications.

8.2.2 Stakeholders

Three main stakeholders in these applications were identified: technical engineering and management personnel; project management personnel; and academic researchers. The project successfully engaged with the first two of these groups in the context of the management support approach (Chapter 4) and the third in the context of the research applications (Chapter 7).

8.2.3 Research motivation

This research was motivated by the proposal that a single modelling approach to address all these applications and stakeholders could provide three benefits: reducing the cost of modelling by allowing adaptation of existing models; supporting communication between stakeholders; and supporting the development of novel approaches. The latter two of these benefits were respectively illustrated by: the management support approach which supports synthesis of a common perspective of design plans from engineers' and managers' viewpoints (Chapter 4); and integration of the research into two Ph.D. projects (Chapter 7). Adapting existing models for new purposes was discussed in the context of re-using process elements (Chapter 7).

8.2.4 Success criterion

Chapter 1 stated that this research would benefit those seeking to improve design processes, both in academic research and industry practice. The success criterion therefore required the research outputs to be validated in industry and to contribute to other academic research projects. This was comprehensively achieved:

1. **Application in industry.** The *P3 Signposting* software was used by researchers in Cambridge and by a design manager in Rolls-Royce to develop descriptive process models which were significantly more detailed than previous descriptions (Chapter 7). It was shown that these users had invested significant time in applying the research, thereby indicating its perceived relevance and the maturity of the software implementation.

2. **Contribution to research.** The *Applied Signposting Model* forms a substantial component of two published research projects at the time of writing, thereby validating the contribution to research (Chapter 7).

3. **Dissemination of research results.** In the first three months of its release for academic and evaluation use, 72 copies of P3 Signposting were distributed following requests from industry and academic users (Chapter 6). This further indicates the relevance of the contributions to academic research and industry practice.

8.3 Limitations

The primary limitation of this thesis is its focus on a single company in the aerospace industry. Generality of the research results thus cannot be proven. However, many discussions with industry and academic experts have supported the findings and indicated the potential for applicability outside Rolls-Royce. Exploring the generality of the findings through additional case studies is an opportunity for further research.

A number of specific limitations to the ASM modelling framework arise from the task precedence representation on which it is based. These are:

1. **Modelling concurrent tasks which exchange information.** Although overlapping tasks may be decomposed into multiple interleaved tasks, this is not appropriate for modelling many concurrent streams of work which frequently exchange information. The approach is therefore unsuitable for modelling systems that incorporate overlapping tasks that cannot be further decomposed. This limitation could be addressed by enhancing the simulation to include a time-stepping element, potentially based on functions describing the rate of releasing information in a manner similar to Carrascosa *et al.* (1998).

2. **Modelling strongly connected networks.** Process models which incorporate many failure points, failure modes and/or interdependencies between tasks cannot be easily modelled using the approach. This shortcoming is common to all modelling frameworks based on a graphical notation. Although the P3 Signposting software does provide matrix views for manipulating ASM models, the network view was almost exclusively used during the application studies due to its greater accessibility. Further research regarding process model visualisation is necessary to address this limitation.[1]

3. **Modelling adaptive processes.** The ASM is unsuited to represent adaptive task selection based on the current state of design information.[2] Although a number of dynamic task models have been proposed to address this issue, the difficulty of validating such approaches was highlighted in Chapter 3. Surmounting these issues and incorporating adaptive task selection into the ASM provides an interesting opportunity for further work since adaptive behaviour is an important characteristic of many design processes.

[1] Although complex information flows cannot be easily represented, this issue can be alleviated by using data interactions to represent redundant feed-forward dependencies. Additionally, behavioural interdependencies between tasks may be modelled using process variables.

[2] Although limited state-contingent selection may be incorporated using process variables to represent state and compound tasks to represent decisions, this is inappropriate where many alternatives are available at every step.

8.4 Opportunities for further research

A number of research opportunities arose from this thesis. These are:

1. **Customised modelling frameworks.** A short-term opportunity for further work is to refine the meta-model technology discussed in Chapter 6 and implement a range of modelling frameworks within P3 Signposting. This would firstly comprise the task network models reviewed in Chapter 2 and would eventually include a wider range of frameworks such as component connectivity models, design rationale models, and frameworks which integrate multiple domains.

   Access to multiple modelling frameworks within the same software tool will allow a comparative analysis to understand the appropriateness of each framework for particular process domains and modelling objectives, resulting in a 'framework wizard' to help modellers identify the most appropriate approach in each case. This could also compose a modelling framework from individual meta-model class structures. This would require theoretical as well as implementation research, since some framework structures pre-suppose alternative conceptualisations of the systems they represent.

2. **Iteration.** Further research is necessary to fully classify the modes of iteration which can occur, their importance and impact on process behaviour, and the factors which influence this. Related opportunities for further research include investigation of how to model the dependencies between product, process and personnel which influence iteration and how iteration impacts upon the outcome of process simulation. Because design process simulation models often cannot be calibrated due to insufficient historical data, it is especially important to ensure the assumptions underlying treatment of iteration are appropriate.

3. **Synthesising design automation and process simulation.** Design of high-performance products and components often relies upon specialised computational tools. In such cases, tool integration can lead to substantial efficiency improvements by reducing the time spent in data transfer and conversion tasks. Task-based models can guide the designer in selecting the

most appropriate task at each step (*e.g.*, Clarkson and Hamilton 2000) or can directly drive selection and execution of design codes (*e.g.*, McMahon and Xianyi 1996). More sophisticated automation frameworks use a model of the emerging design space to guide activity selection (*e.g.*, Jarrett 2000). Since automated design can be computationally expensive, it would be useful to extend these frameworks to optimise process performance alongside design performance. An interesting opportunity therefore exists to synthesise consideration of *product* information with *process* knowledge to enhance automation tools. Two complementary research questions arise from this:

- (How) can complete or partial information about the evolving design state be incorporated to improve the fidelity of simulation models?
- (How) can process simulation be used to inform task selection in design automation frameworks?

The scriptlets and process variables within P3 Signposting provide suitable infrastructure for exploring the use of parametric design information to influence process behaviour. Although some preliminary research was conducted in this area it was ultimately considered beyond the scope of this thesis.

4. **Representing and reasoning about uncertainty in simulation.** Due to the uncertainty surrounding design process behaviour and its modelling, it is important to evaluate the validity of advice derived from process simulation. Although uncertainty in estimating simulation parameters could be estimated and treated separately from the uncertainty inherent to the process, current design process simulation approaches do not account for this. An opportunity for further research is therefore to explore how epistemic uncertainty can be identified, represented in design process simulation models and utilised to explore the validity of analysis results.

8.5 Conclusion

The performance of product development processes is important to the commercial success of new products. This dissertation has introduced an approach to model such processes and thereby to support their improvement. The approach has been applied to investigate the representation, simulation and management of design processes in a major U.K. aerospace manufacturer. It was implemented in a configurable software platform, which is now being used by a number of ongoing research projects in the Cambridge Engineering Design Centre.

Appendix A

Case study background

This appendix lists the informal interviews conducted and meetings observed during the case study reported in Chapter 3. A brief description of the Rolls-Royce organisational structure is provided to aid interpretation of individuals' roles.

A.1 Organisational structure

At the time of writing, Rolls-Royce is a matrix organisation structured into Operational Business Units (OBUs) and Customer Facing Business Units (CFBUs). Each OBU is responsible for designing a particular sub-system in the engine; for example, the Turbine Systems OBU employs specialists in turbine engineering. These personnel further specialise into functions, for example in the aerodynamics or materials issues specific to turbine design. OBU personnel may be assigned to one project or may work on several projects concurrently. In addition to this project work, they are responsible for maintaining and improving the design technology specific to their function.

OBUs are complemented by CFBUs, which are responsible for the delivery of specific projects and which include many non-engineering personnel. For example, a CFBU may include several tiers of programme managers and support personnel.

A.2 Interviews conducted

Personnel in OBU roles included technical designers, aerodynamicists and stress engineers as well as technical team leaders. Personnel in CFBU roles included the Control Account Manager (CAM) responsible for timely delivery of the F136 fan rotative components and the Programme Manager sponsoring the research, who was responsible for delivery of the entire fan module. Specialist CFBU personnel such as the Earned Value Management Systems (EVMS) Specialist and Project Accountant were interviewed.

A number of discussions were also held with personnel not associated with the F136 fan blisk, such as a Knowledge Management Specialist and members of the Design for Process Excellence team, a Rolls-Royce initiative similar to Design for Six Sigma. The interviews are summarised in Figure A.1, together with the details of a number of meetings and workshops which were attended during the case study.

Role	Affiliation
Programme Manager	CFBU
Earned Value Management Systems Specialist	CFBU
Bidding and Proposal Manager	CFBU
Transmissions IPT Lead	OBU/ CFBU
Test Engineer	OBU
Rotatives Control Account Manager	OBU/CFBU
Aerodynamics Team Leader	OBU
Fan Module IPT Lead	OBU
Project Accountant	CFBU
Technical Aerodynamicist	OBU
Manufacturing Engineer	OBU
Stress Engineer	OBU
Design Engineer	OBU
Design Engineer	OBU
Stress Engineer	OBU
Design Engineer	OBU
-- JOB DESCRIPTION UNAVAILABLE --	CFBU
Design Engineer	OBU
Programme Management Specialist	Other
Process Excellence Team Leader	Other
-- JOB DESCRIPTION UNAVAILABLE --	Other
Knowledge Management Specialist	Other

Scheduled meeting	Frequency	Purpose
Management Review	Weekly	Monitoring component-level progress and risks
Hardware Review	Weekly	Monitoring prototype manufacturing
Product Change Board	Weekly	Stage-gate reviews for individual components
Business Review	Monthly	Review of major project deliverables

One-off workshop	Duration	Purpose
Cost/Weight Reduction	2 days	Identify opportunities to reduce cost and weight
Architecture Review	1 day	Senior RR technical personnel review architecture
GE Leadership Visit	2 days	GE/RR project leadership review formal processes

Figure A.1 Personnel interviewed and meetings observed during the study.

Appendix B

Prototype software

This appendix details the process modelling and planning support tool described in Chapter 4.

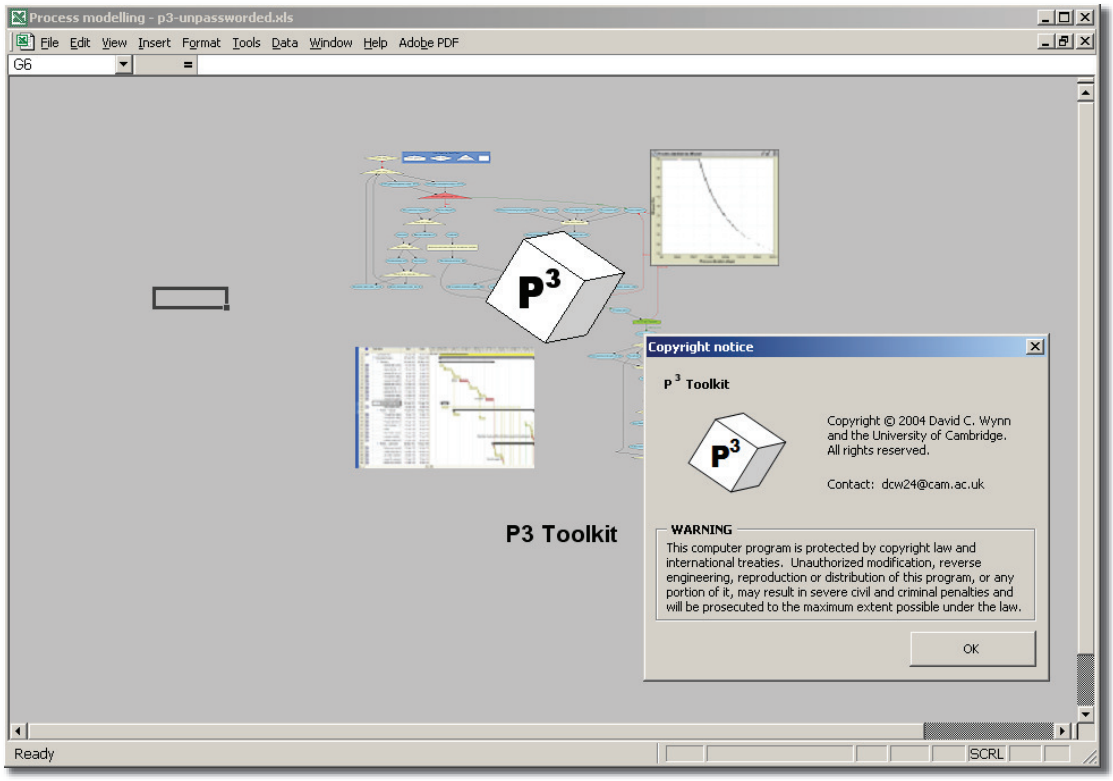


Figure B.1 The prototype software.

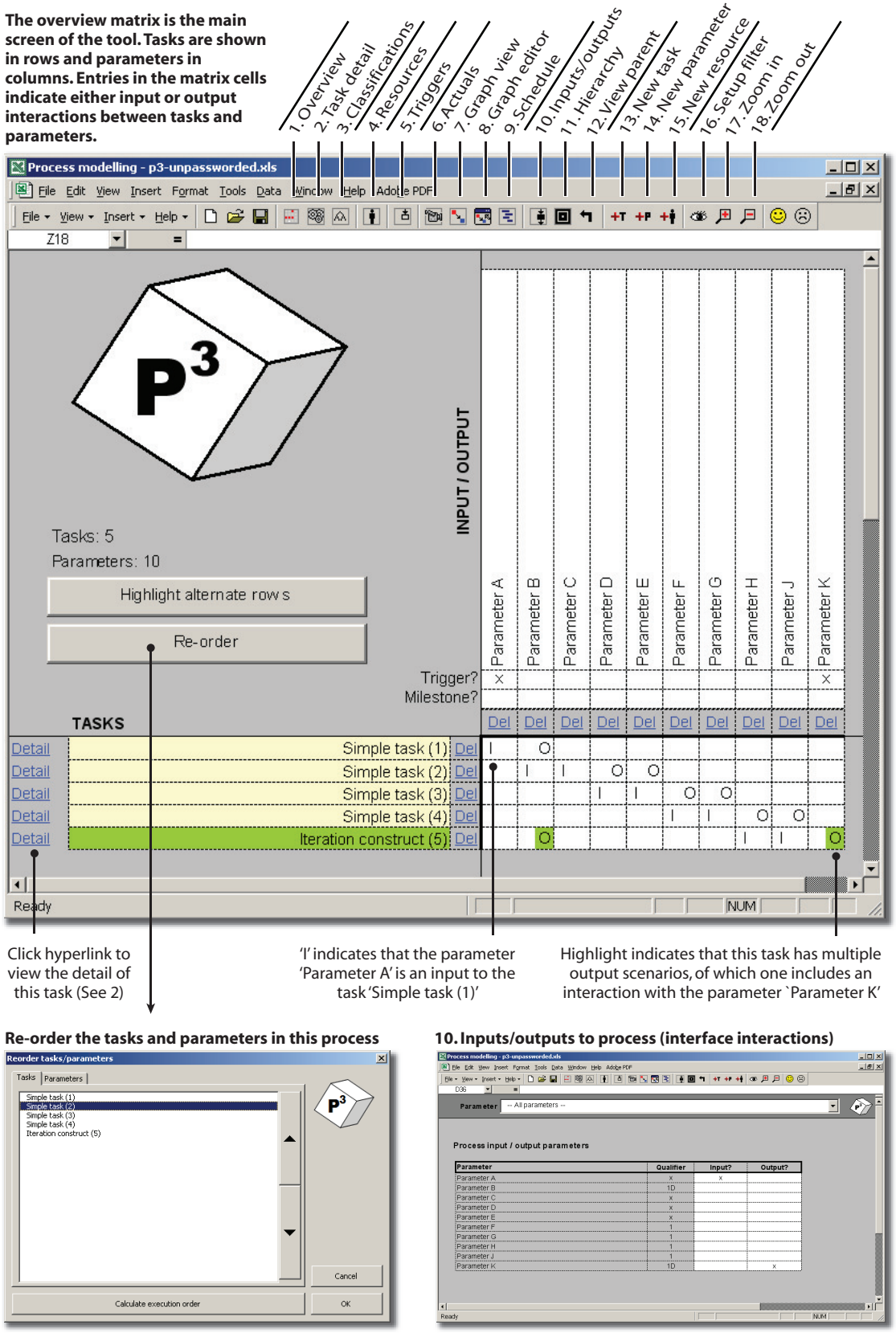


Figure B.2 The modelling interface is constructed from domain-mapping matrices for each sub-process.

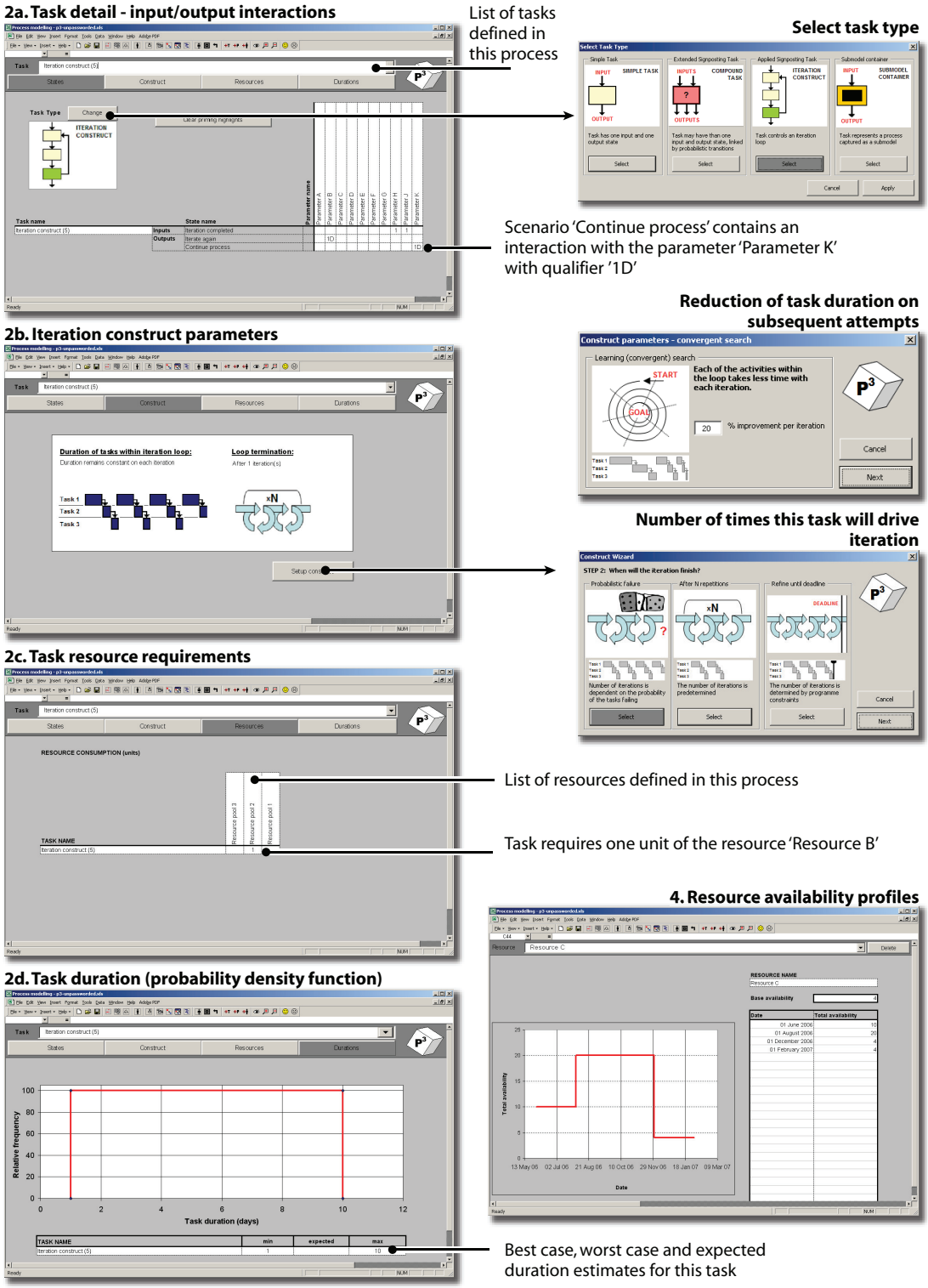


Figure B.3 Clicking a row in the domain-mapping matrix produces screens allowing the properties of individual tasks to be manipulated.

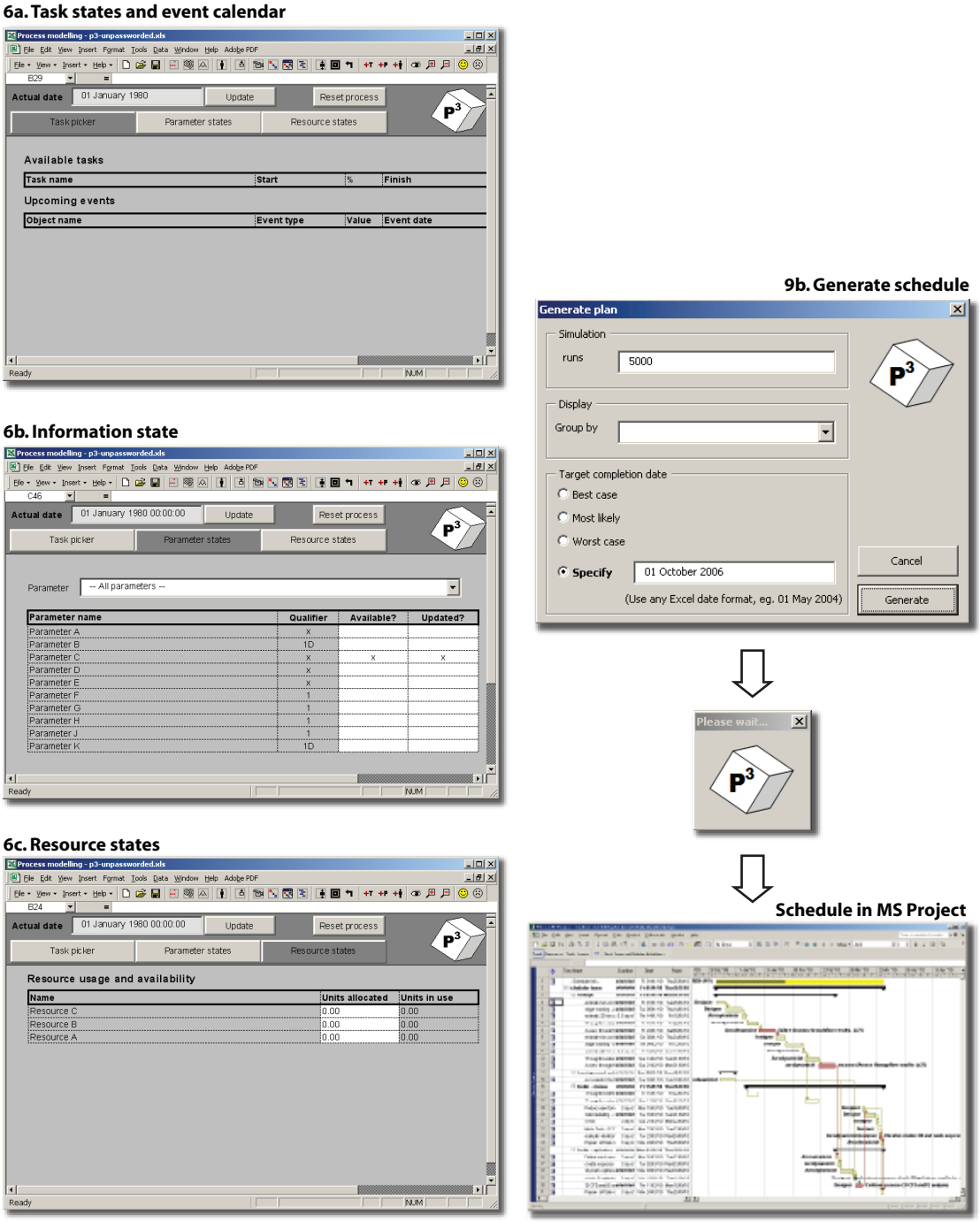


Figure B.4 The process simulation may be debugged by stepping through tasks and viewing the state of information and resources at each step (left). Following simulation, a Gantt chart of a single outcome may be selected and opened for viewing in MS Project (right).

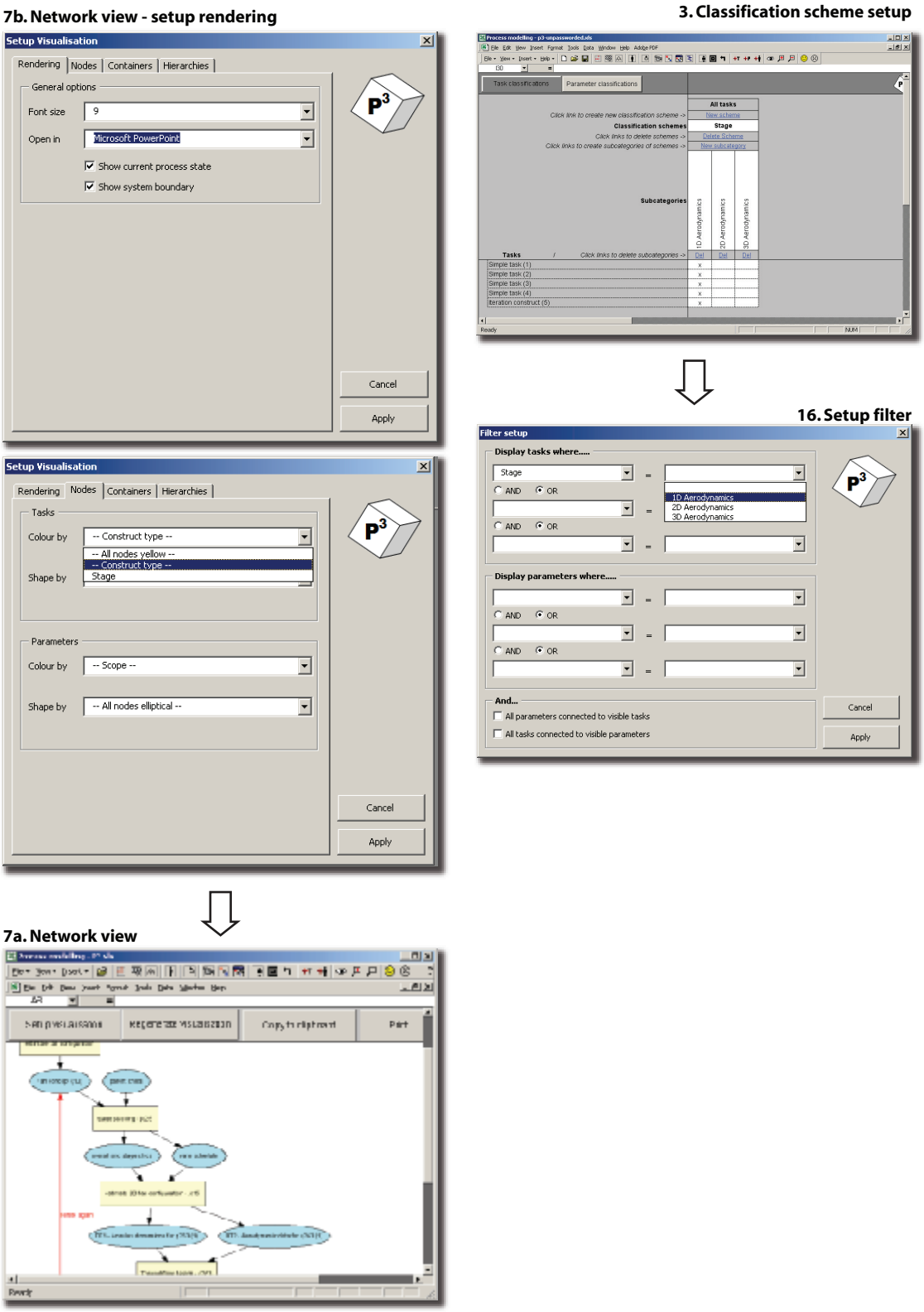


Figure B.5 Network layouts are automatically generated using the AT&T GraphViz 'dot' algorithm (left). They may be configured to conceal tasks or parameters which are members of specified classification categories (right).

Appendix C

P3 Signposting software

This appendix provides additional detail regarding the *P3 Signposting* software discussed in Chapter 6. User-oriented documentation is available in the 100-page *P3 Signposting Users' Guide* document (Wynn and Clarkson, 2006).

Figure C.1 The P3 Signposting software.

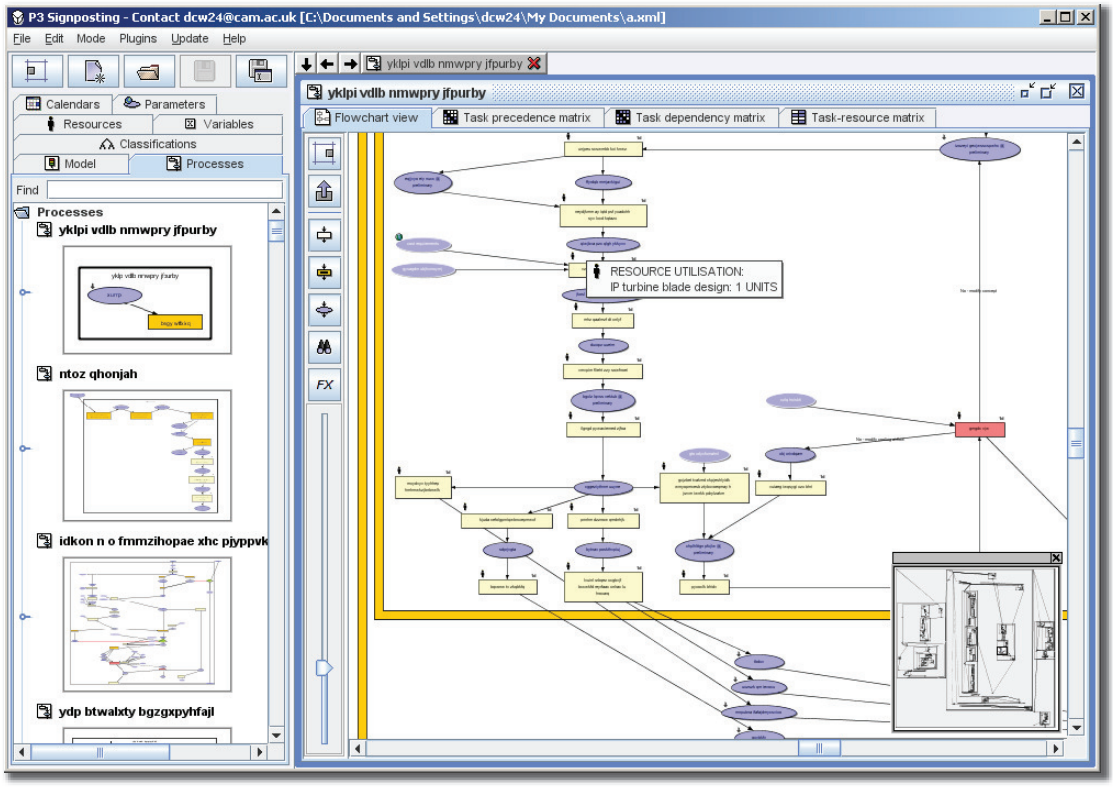


Figure C.2 Element properties are summarised using sets of interactive icons.

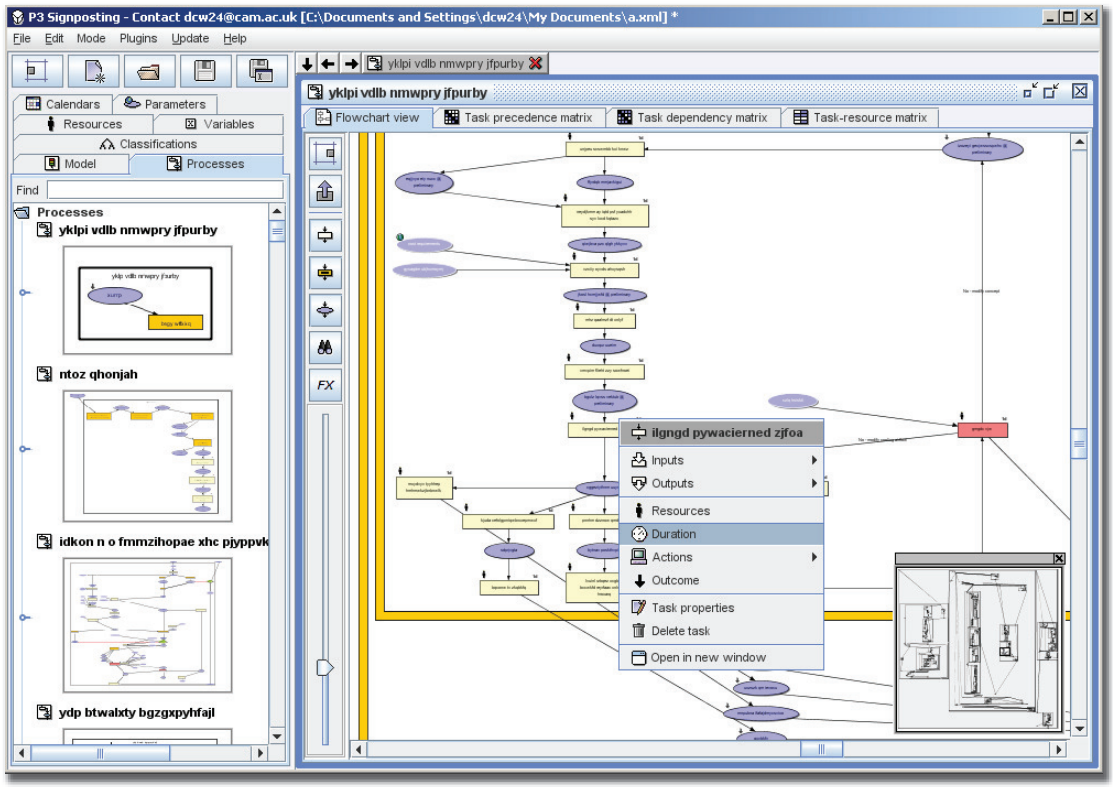


Figure C.3 Element properties may be modified by right-clicking that element in the modelling interface.

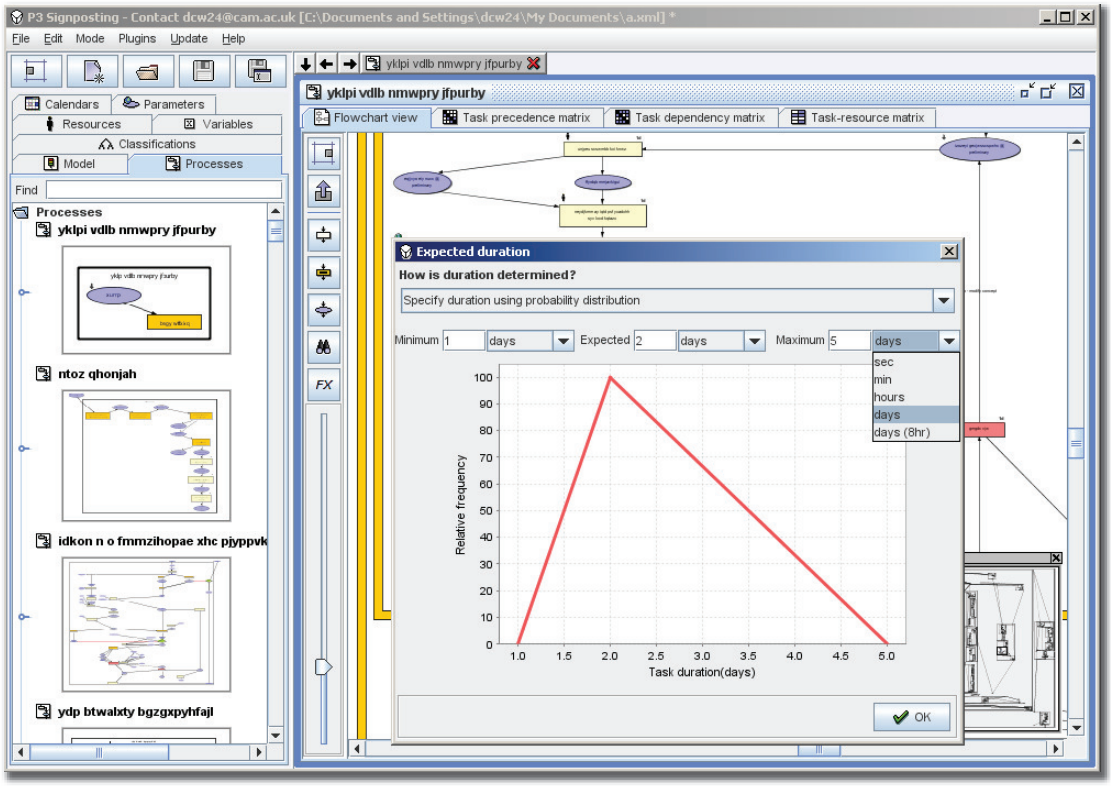


Figure C.4 Uncertainty in tasks' durations may be modelled using triangular or uniform probability density functions.

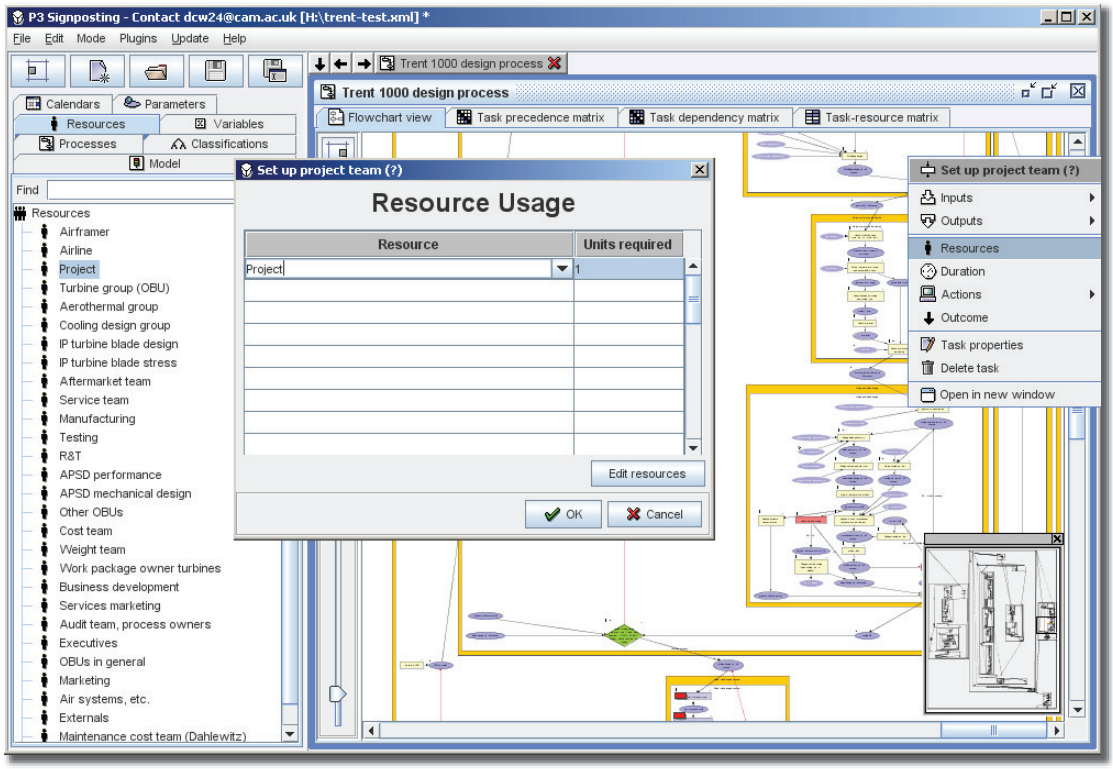


Figure C.5 Tasks may require units from specified resource groups to begin.

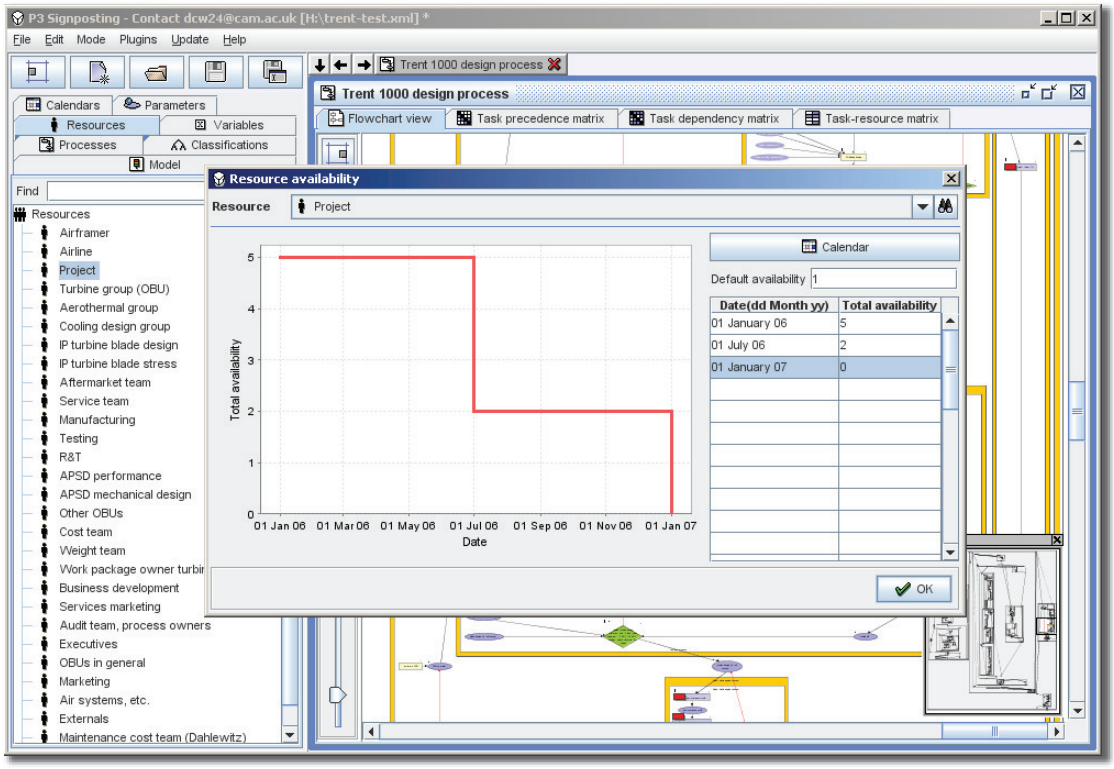


Figure C.6 Each resource has an availability profile and a calendar which determines its working hours.

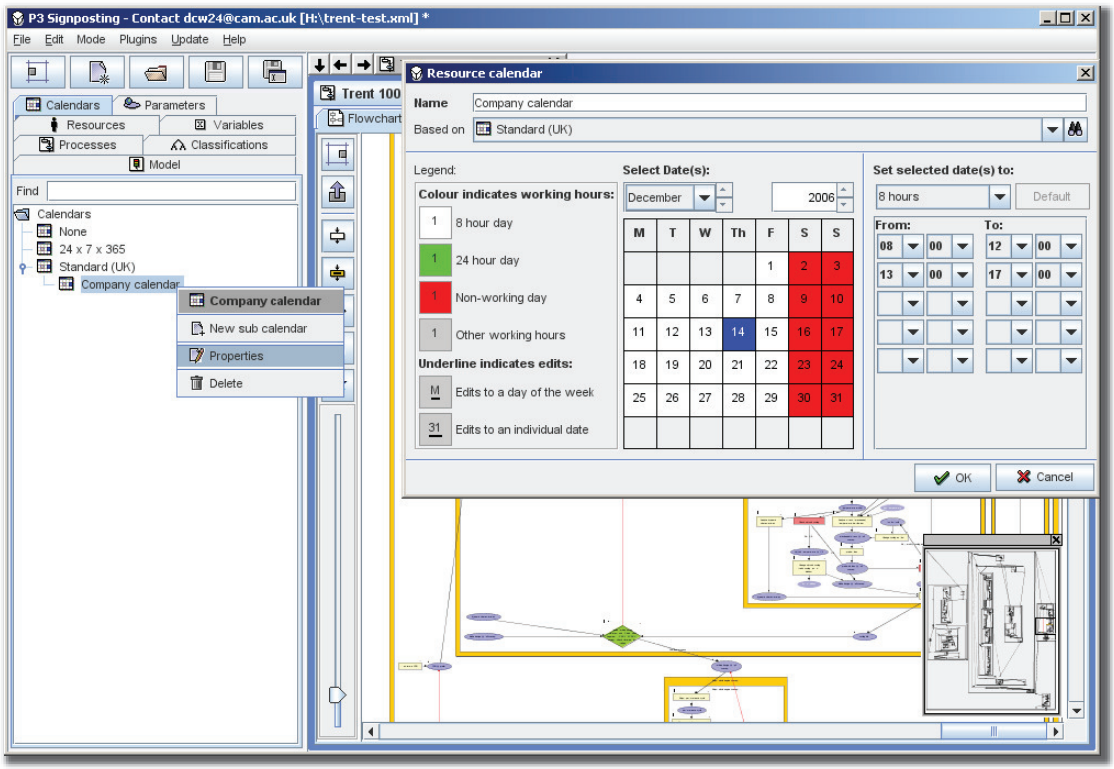


Figure C.7 Each calendar inherits its configuration from its parent and overrides the working hours on specified days.

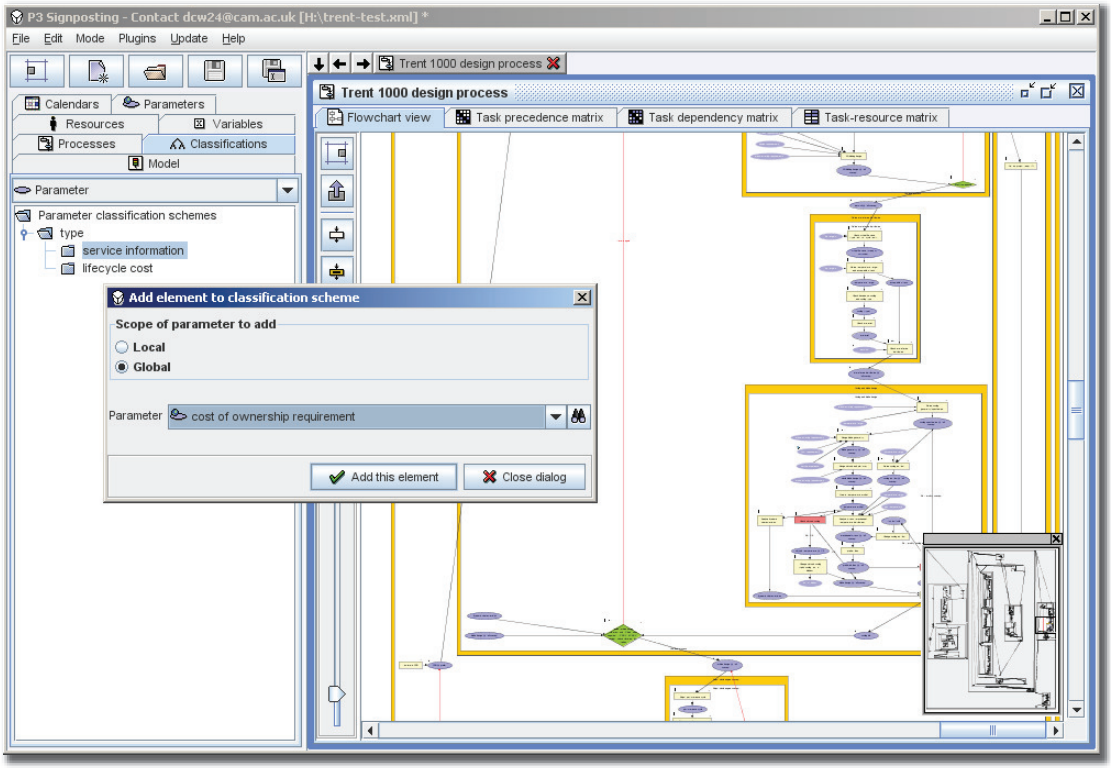


Figure C.8 Elements may be added to one or more categories in one or more classification schemes. Categorising elements allows the display to be filtered.

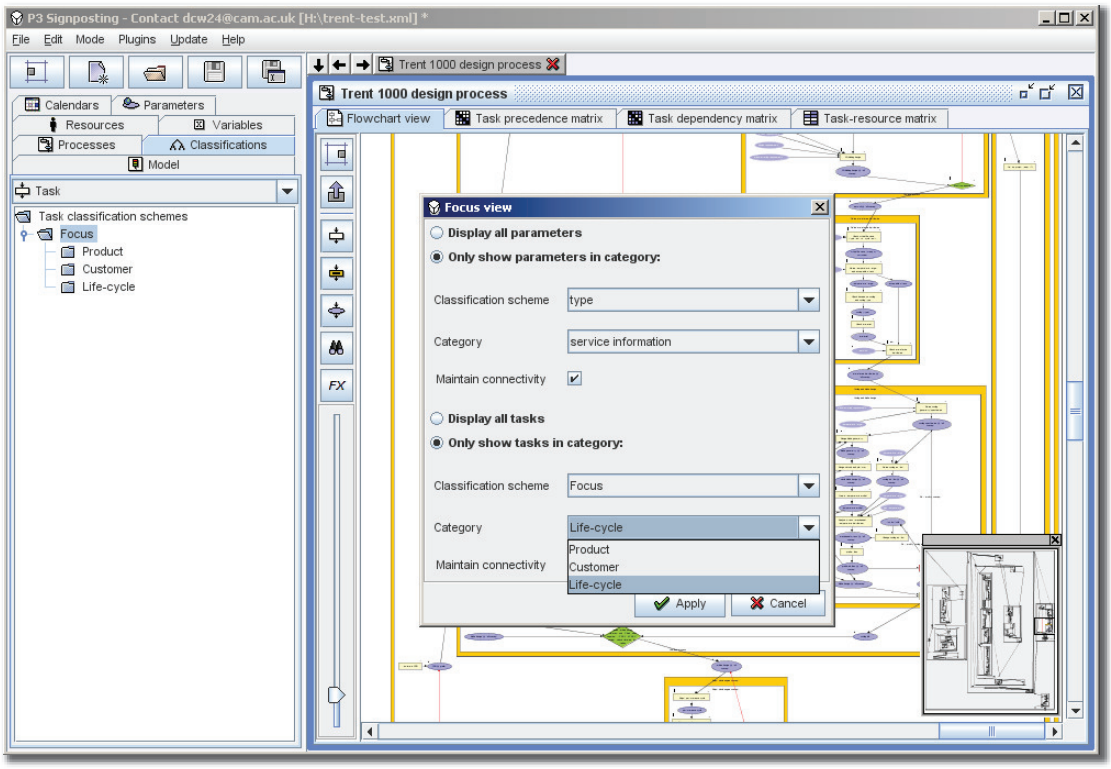


Figure C.9 Configuring the network view to focus on tasks and parameters which belong to specified categories.

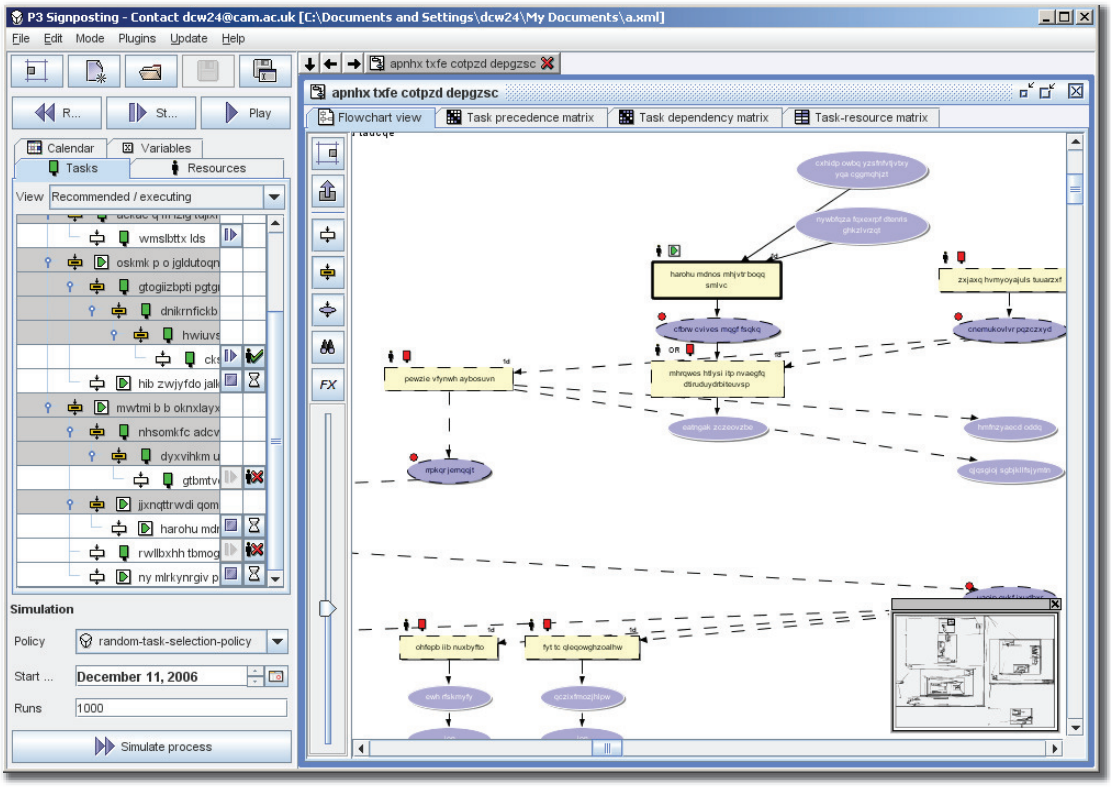


Figure C.10 The software provides a 'debug mode' for manually stepping through the simulation algorithm to verify its behaviour.

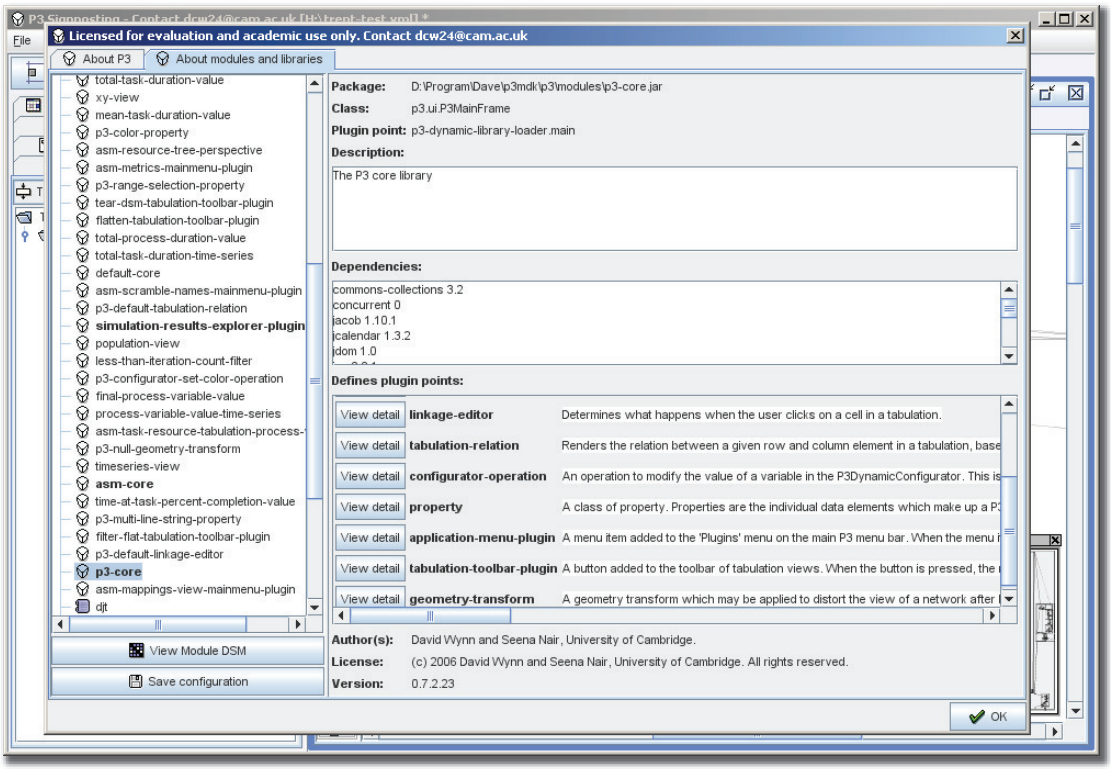


Figure C.11 The software is dynamically constructed from the set of modules found at run-time. Each installation may have a unique set of modules.

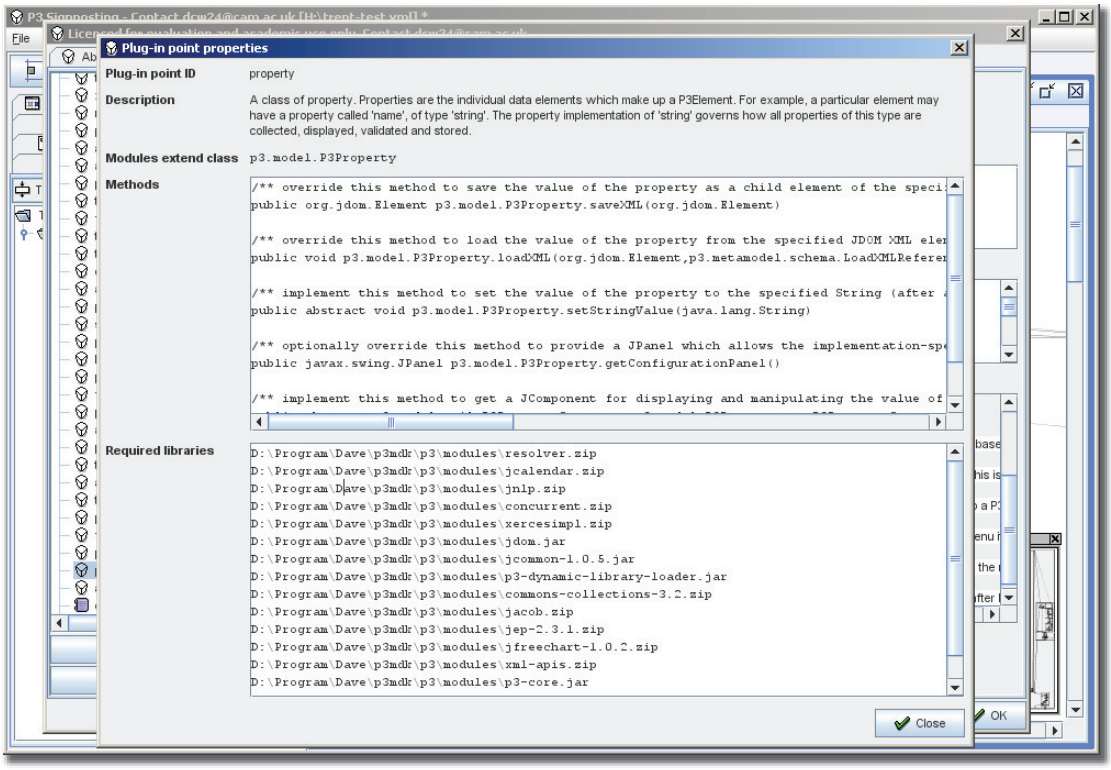


Figure C.12 All information required to implement a new module can be located by browsing the plug-in point definitions. Any module may define new plug-in points.

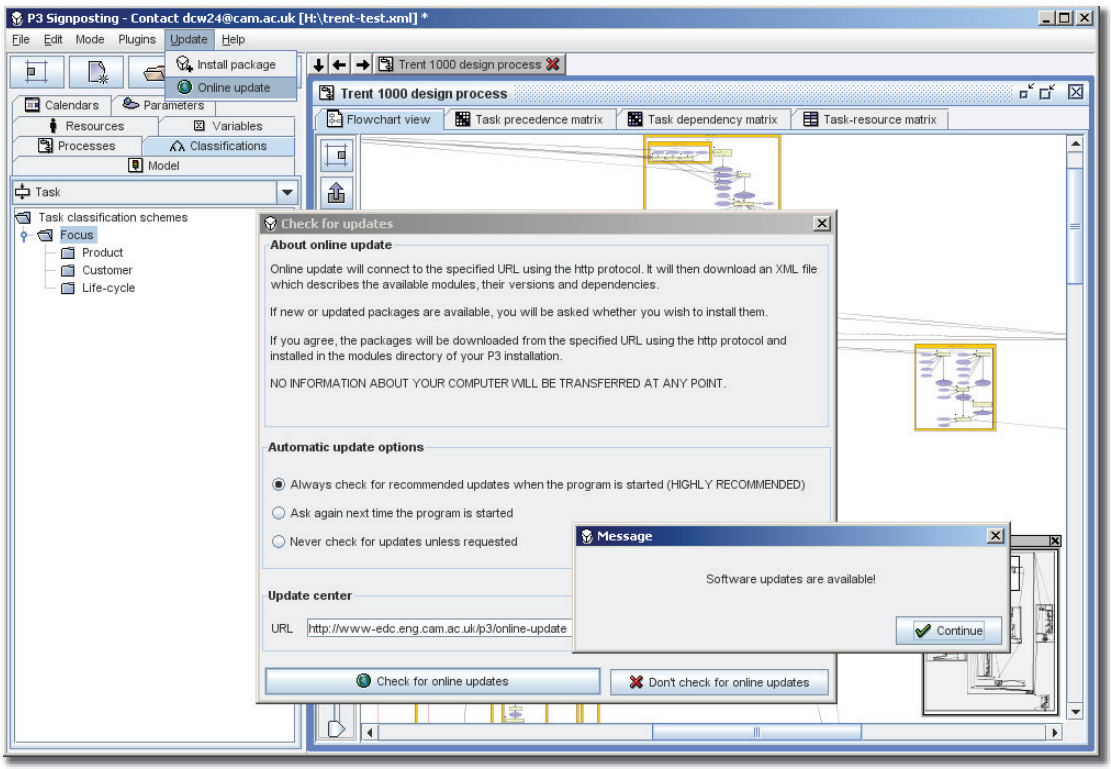


Figure C.13 A versioning system allows modules to be released, updated and installed individually, either by distributing packages or by providing an online update site.

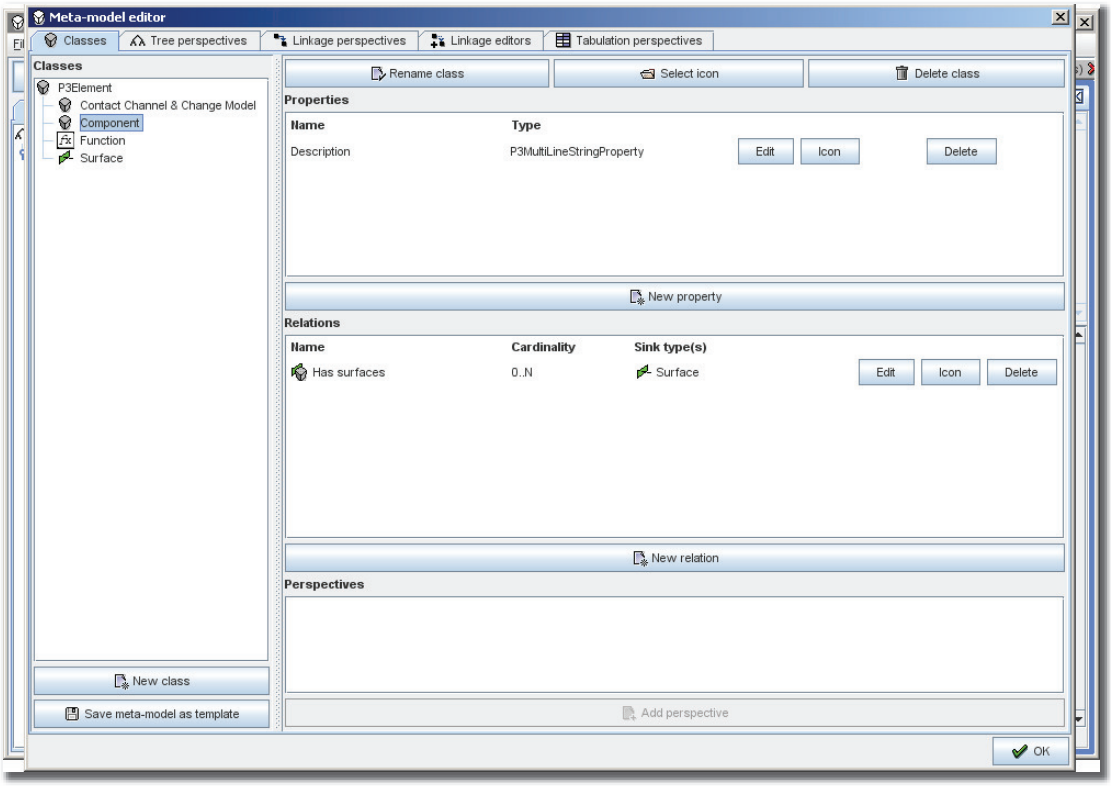


Figure C.14 The meta-model configurator allows custom modelling frameworks to be constructed without writing additional code.

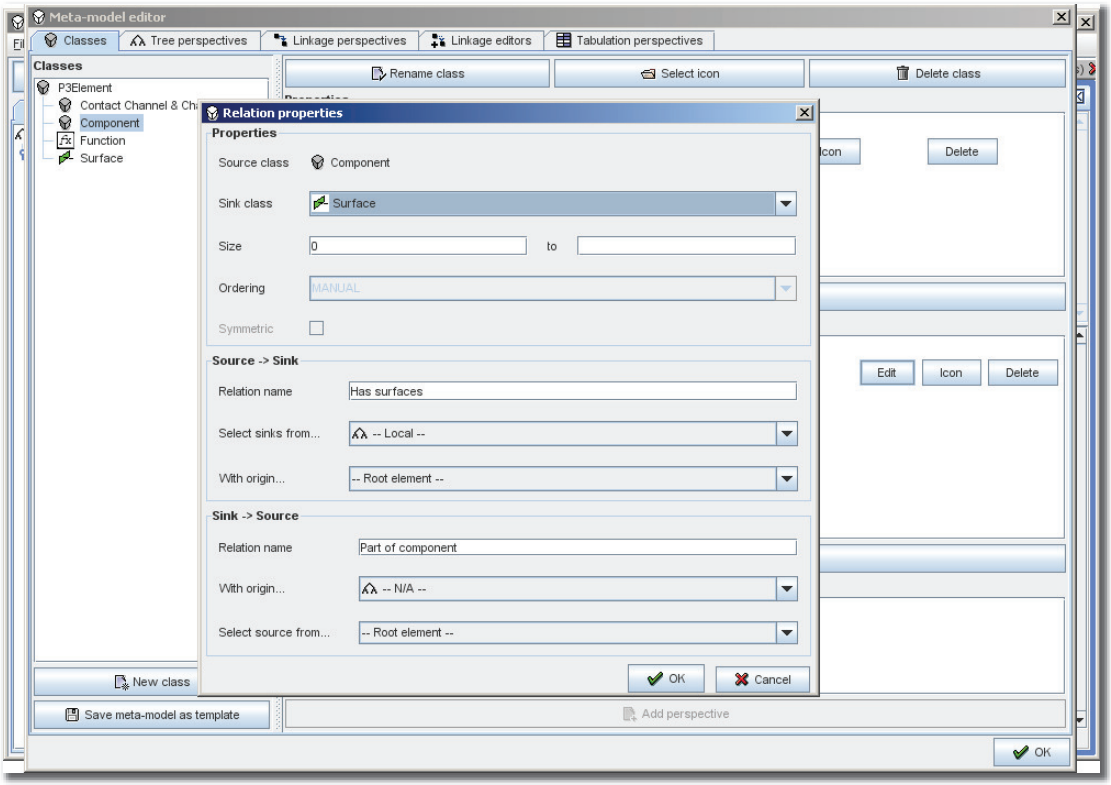


Figure C.15 Meta-models consist of classes, defining the types of element allowed in the model, and relations which define how elements can be connected.

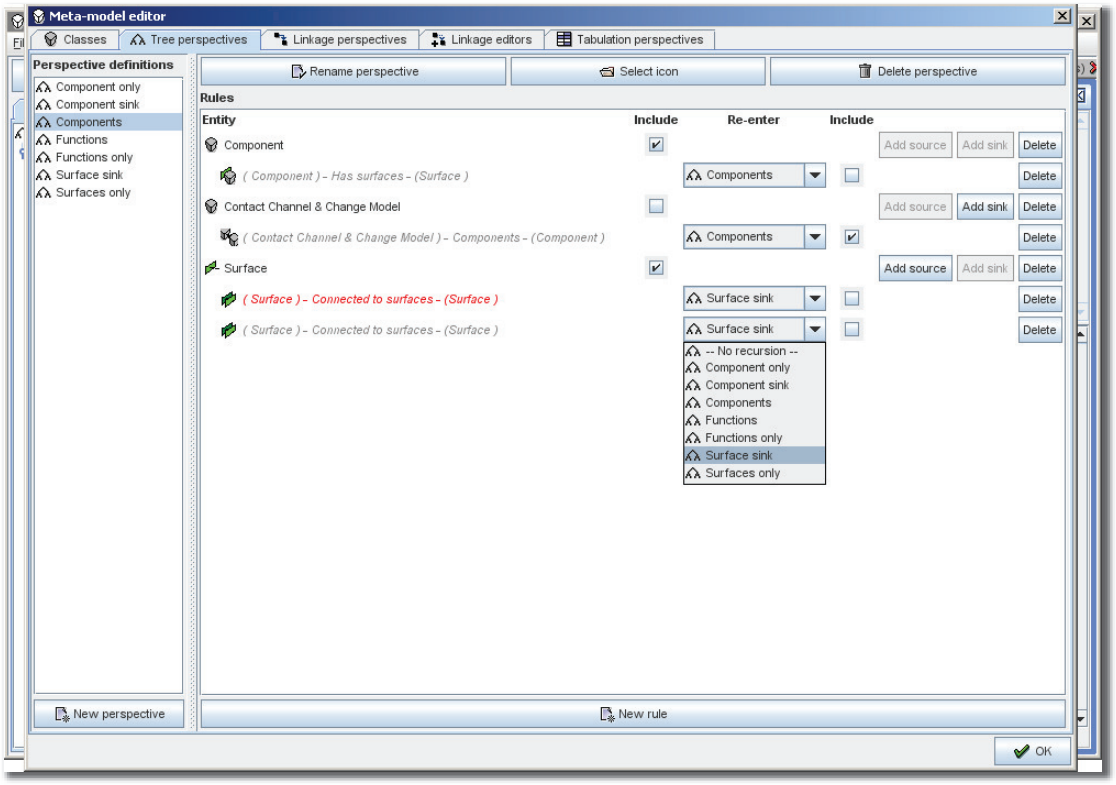


Figure C.16 Perspectives are algorithms which process the graph structure of a model to construct views. Configuration options depend upon the implementing module.

Bibliography

Acsian (2007) Plexus Modeller. http://www.acsian.net *Acsian Ltd., 28 February 2007.*

Adler PS, Mandelbaum A, Nguyen V, Schwerer E (1995) From project to process management: An empirically-based framework for analyzing product development time. *Management Science, 41(3), pp. 458-484.*

Albers A, Matthiesen S, Ohmer M (2003) An innovative new basic model in design methodology for analysis and synthesis of technical systems. *Proceedings of the 14th International Conference on Engineering Design (ICED '03), Stockholm.*

Alink T (2005) The contact and channel model in the change prediction method. *Diploma thesis, Institut für Produktentwicklung, Universität Karlsruhe, Germany.*

Andersson J, Pohl J, Eppinger SD (1998) A design process modeling approach incorporating nonlinear elements. *Proceedings of the ASME Design Theory and Methodology Conference (DETC 1998), Atlanta.*

Andreasen MM, Hein L (1987) Integrated product development. *IFS (Publications)/Springer.*

ANSI/EIA-748-A-1998 (1998) Earned value management systems. *American National Standards Institute (ANSI).*

APQP (1995) Advanced product quality planning and control plan (APQP). *http://www.aiag/publications/quality/APQP2.asp*

Archer LB (1965) Systematic method for designers. *The Design Council.*

Ashby MF, Bréchet YJM, Cebon D, Salvo L (2004) Selection strategies for materials and processes. *Materials and Design, 25(1), pp. 51-67.*

Asimow M (1962) Introduction to design. *Prentice Hall.*

Austin S, Baldwin A, Li B and Waskett P (1999) Analytical design planning technique: A model of the detailed building design process. *Design Studies, 20(3), pp. 279-296.*

Ayton P, Pascoe E (1995) Bias in human judgement under uncertainty? *Knowledge Engineering Review, 10, pp. 21-41.*

Bahrami A, Dagli CH (1993) Models of design processes. *In: Concurrent engineering: contemporary issues and modern design tools. Chapman and Hall.*

Basu A, Blanning R (1997) Metagraph transformations and workflow management. *Proceedings of the 30th Annual Hawaii International Conference on System Sciences, IEEE.*

Baxter MR (1995) Product design: practical methods for the systematic development of new products. *Chapman and Hall.*

Bayazit N (2004) Investigating design: A review of forty years of design research. *Design Issues, 20(1), pp. 16-29.*

Bechhofer S, van Harmelen F, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA (2004) OWL web ontology language reference. http://www.w3.org/TR/owl-ref/ *World Wide Web Consortium (W3C), 10 February 2004.*

Bell CP, Wynn DC, Dawes WN and Clarkson PJ (2007) Using meta-data to enhance process simulation and identify improvements. *16th International Conference on Engineering Design (ICED'07), Paris, France (forthcoming).*

Birmingham R, Cleland G, Driver R, Maffin D (1997) Understanding engineering design: Context, theory and practice. *Prentice Hall.*

Blessing LTM (1994) A process-based approach to computer-supported engineering design. *PhD thesis, University of Twente, The Netherlands.*

Blessing LTM, Charkrabarti A, Wallace KM (1995) A design research methodology. *10th International Conference on Engineering Design (ICED'95), Prague, Czech Republic, 1, pp. 50-55.*

Boehm B (1988) A spiral model of software development and enhancement. *Computer, May 1988.*

Browning TR (2001) Applying the design structure matrix system to decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering Management, 48, pp. 292-306.*

Browning TR, Deyst JJ, Eppinger SD, Whitney DE (2002a) Adding value in product development by creating information and reducing risk. *IEEE Transactions on Engineering Management, 49(4), pp. 443-458.*

Browning TR, Eppinger SD (2002b) Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management, 49(4), pp. 428-442.*

Browning TR, Ramasesh RV (2007) A survey of activity network-based process models for managing product development projects. *Production and Operations Management, 2007 (forthcoming).*

Bucciarelli LL (1996) Designing engineers. *MIT Press.*

Carrascosa M, Eppinger SD, Whitney D (1998) Using the design structure matrix to estimate product development time. *Proceedings of ASME Design Engineering Technical Conferences, Atlanta, Georgia, USA.*

Chalupnik MJ, Wynn DC, Eckert CM and Clarkson PJ (2007) Understanding design process robustness: a modelling approach. *16th International Conference on Engineering Design (ICED'07), Paris, France (forthcoming).*

Cho S, Eppinger SD (2001) Product development process modeling using advanced simulation. *Proceedings of ASME Design Engineering Technical Conferences, Pittsburgh, Pennsylvania, USA.*

Chung MJ, Kwon P, Pentland BT (2002) Making process visible: A grammatical approach to managing design processes. *Journal of Mechanical Design, 124, pp. 364-374.*

Clarkson PJ, Hamilton JR (2000) Signposting: a parameter-driven task-based model of the design process. *Research in Engineering Design, 12(1), pp. 18-38.*

Clarkson PJ, Melo AF, Eckert CM (2000) Visualization techniques to assist design process planning. DETC'00.

Clarkson PJ, Wallace K, Ashby M, Johnson A, Parks G, Symons D (2006) Cambridge engineering design centre IMRC fifth annual report, October 2005 - September 2006.

Cooper KG (1993) The rework cycle: How it really works... and reworks... *PM-Network, February 1993.*

Cooper RG (1994) Third generation new product processes. *Journal of Product Innovation Management, 11, pp. 3-14.*

Cope M (2002) The seven Cs of consulting: The definitive guide to the consulting process. *FT Prentice Hall.*

Costa R (2004) Productive iteration in student engineering design projects. *MSc thesis, Montana State University, USA.*

Costa R and Sobek DK II (2003) Iteration in engineering design: inherent and unavoidable or product choices made? *Proceedings of ASME DETC'03, Chicago, Illinois, USA.*

Crilly N (2005) Product aesthetics: representing designer intent and consumer response. *PhD thesis, University of Cambridge.*

Crilly N, Blackwell A, Clarkson PJ (2005) Graphic elicitation: using research diagrams as interview stimuli. *Qualitative Research, 6(3), pp. 341-366.*

Cross N ed. (1984) Developments in design methodology. *John Wiley and Sons Limited.*

Cross N, Roozenburg N (1992) Modelling the design process in engineering and in architecture. *Journal of Engineering Design, 3(4), pp. 325-337.*

Cross N (1994) Engineering design methods: strategies for product design. *John Wiley and Sons Limited.*

Cross N, Christiaans H and Dorst K (1996) Analysing design activity. *John Wiley and Sons Limited.*

Danilovic M, Browning TR (2007) Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management, 2007 (forthcoming).*

Darke J (1979) The primary generator and the design process. *Design Studies, 1(1), pp. 36-44.*

Douglas DE (1978) PERT and simulation. *10th Winter Simulation Conference.*

DTI (1994) Successful product development. *Technical report, Department of Trade and Industry, HMSO.*

Dym CL, Little P (2000) Engineering design: A project-based introduction. *John Wiley and Sons Limited.*

Eckert CM (1997) Intelligent support for knitwear design. *PhD thesis, Open University.*

Eckert CM, Clarkson PJ (2003) The reality of design process planning. *ICED'03, Stockholm.*

Eckert CM, Stacey MK, Clarkson PJ (2004a) The lure of the measurable in design research. *Design 2004, Dubrovnik, Croatia, pp. 21-26.*

Eckert CM, Clarkson PJ, Zanker W (2004) Change and customisation in complex engineering domains. *Research in Engineering Design, 15(1), pp. 1-21.*

Eckert CM (2006) Generic and specific process models: Lessons from modelling the knitwear design process. *Tools and Methods for Concurrent Engineering 2006, Ljubljana, Slovenia.*

Eppinger S, Whitney D, Smith R, Gebala D (1994) A model-based method for organizing tasks in product development. *Research in Engineering Design, 6, pp. 1-13.*

Eppinger SD (1991) Model-based approaches to managing concurrent engineering. *Journal of Engineering Design, 2(4), pp. 283-290.*

Evans JH (1959) Basic design concepts. *Journal of the American Society of Naval Engineers, November 1959, pp. 671-678.*

Finger S, Dixon JR (1989) A review of research in mechanical engineering design, part I: descriptive, prescriptive and computer-based models of design processes. *Research in Engineering Design, 1(1), pp. 51-68.*

Flanagan T (2006) Supporting design planning through process model simulation. *PhD thesis, University of Cambridge.*

Ford DN, Sterman JD (1998) Dynamic modeling of product development processes. *System Dynamics Review, 14(1), pp. 31-68.*

French MJ (1999) Conceptual design for engineers. *Springer.*

Friesleben D, Vajna S (2002) Dynamic process navigation: Modeling, improving and review of engineering processes. *ASME Design Engineering Technical Conferences, Montreal, Canada.*

Frost RB (1992) A converging model of the design process: Analysis and creativity, the ingredients of synthesis. *Journal of Engineering Design, 3(2), pp. 117-126.*

Ghoniem M, Fekete J, Castagliola P (2004) A comparison of the readability of graphs using node-link and matrix-based representations. *IEEE Symposium on Information Visualisation 2004, Austin, Texas, USA.*

Grose DL (1994) Reengineering the aircraft design process. *5th AIAA/USAF/-NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Panama City Beach, Florida, USA.*

GraphViz (2007) Graph visualisation software. http://www.graphviz.org/ *AT&T Research, 28 February 2007.*

Hales C (2004) Managing engineering design. *Springer.*

Hall AD (1962) A methodology for systems engineering. *Van Nostrand Reinhold.*

Hamilton JR (1998) The capture and representation of knowledge to support aerospace design. *PhD thesis, University of Cambridge.*

Hillier B, Musgrove J, O'Sullivan P (1972) Knowledge and design. *In: Environmental design: research and practice. University of California, USA.*

Hubka V, Eder E (1980) Design science. *Springer-Verlag, Berlin.*

Hubka V (1982) Principles of engineering design. *Butterworth.*

ISO9000 (1994) BS EN ISO 9000-1: Quality management and quality assurance standards - guidelines for selection and use. http://www.bsi-global.com/ *BSI Group.*

Jarratt TAW (2004) A model-based approach to support the management of engineering change. *PhD thesis, University of Cambridge.*

Jarrett JP (2000) Technology or methodology? An approach to designing better turbomachinery. *PhD thesis, University of Cambridge.*

Jones JC (1963) A method of systematic design. *In: Conference on design methods. Pergamon Press.*

Jones JC (1970) Design methods: seeds of human futures. *Wiley.*

JSF (2007) F-35 Lightning II Joint Strike Fighter Program. http://www.jsf.mil/ *United States Government/Department of Defense, 28 February 2007.*

Keller R, Alink T, Pfeiffer C, Eckert CM, Clarkson PJ, Albers A (2007) Product models in design: a combined use of two models to assess change risks. *16th International Conference on Engineering Design (ICED'07), Paris, France (forthcoming).*

Kreimeyer M, Eichinger M, Lindemann U (2007) Aligning multiple domains of design processes. *16th International Conference on Engineering Design (ICED'07), Paris, France (forthcoming).*

Kusiak A, Larson TN, Wang J (1994) Reengineering of design and manufacturing processes. *Computers and Industrial Engineering, 26(3), pp. 521-536.*

Kusiak A, Yang H (1993) Modelling the design process with Petri nets. *Con-*

current Engineering: Contemporary issues and modern design tools. Parsaei HR and Sullivan WG (eds.) Chapman and Hall.

Lawson B (1980) How designers think. *The Architectural Press Limited.*

Lévárdy V, Hoppe M, Browning TR (2004) Adaptive test process - An integrated modeling approach for test and design activities in the product development process. *Proceedings of ASME Design Engineering Technical Conferences, Salt Lake City, Utah, USA.*

Lévárdy T, Browning TR (2005) Adaptive Test Process - Designing a project plan that adapts to the state of a project. *INCOSE 2005.*

Levitt RE, Thomsen J, Christiansen TR, Kunz JC, Jin Y, Nass C (1999) Simulating project work processes and organizations: towards a micro-contingency theory of organizational design. *Management Science, 45(11), pp. 1479-1495.*

Maffin D, Alderman N, Braiden PM, Hills W, Thwaites A (1995) Company classification: a new perspective on modelling the engineering design and product development process. *Journal of Engineering Design, 6(4), pp. 275-289.*

March L (1984) The logic of design. *In: Cross N (ed.) Developments in design methodology. John Wiley and Sons.*

Mayer RJ, Menzel PC, Painter MK, deWitte PS, Blinn T and Perakath B (1995) Information Integration for Concurrent Engineering (IICE) IDEF3 process description capture method report. *Knowledge Based Systems, Incorporated.*

McMahon CA, Xianyi M (1996) A network approach to parametric design integration. *Research in Engineering Design, (1996)8, pp. 14-32.*

Melo AF (2002) A state-action model for design process planning. *PhD thesis, University of Cambridge.*

Mihm J, Loch C and Huchzermeier A (2003) Problem-solving oscillations in complex engineering projects. *Management Science, 46(6), pp. 733-750.*

Narahari Y, Viswanadham N, Kumar VK (1999) Lead time modelling and acceleration of product design and development. *IEEE Transactions on Robotics and Automation, 15(5), pp. 882-896.*

NIST (1993) Integration definition for function modeling (IDEF0). *National Technical Information Service, U.S. Department of Commerce, Federal Information Processing Standards Publication 183 (FIPSPUB 183).*

O'Donovan BD (2004) Modelling and simulation of engineering design processes. *PhD thesis, University of Cambridge.*

O'Donovan BD, Eckert CM, Clarkson PJ (2004) Simulating design processes to assist design process planning. *Proceedings of ASME Design Engineering Technical Conferences, Salt Lake City, Utah, USA.*

Olsen J, Cagan J, Kotovsky K (2006) Unlocking organisational potential: a computational platform for investigating structural interdependence in design. *Proceedings of ASME Design Engineering Technical Conferences, Philadelphia, Pennsylvania, USA.*

Pahl G, Beitz W (1995) Engineering design: A systematic approach. *Springer.*

Peirce CS (1923) Chance, love and logic. *Kegan Paul.*

Peterson JL (1981) Petri net theory and the modelling of systems. *Prentice Hall.*

PMI (2004) A guide to the Project Management Body of Knowledge (PMBoK). *Project Management Institute.*

Pritsker AAB (ed.) (1966) GERT: Graphical evaluation and review technique. *The RAND Corporation, RM-4973-NASA, April 1966.*

Pritsker AAB (1979) Modelling and analysis using Q-GERT networks (2nd edition). *John Wiley and Sons.*

Pugh S (1991) Total design: Integrated methods for successful product design. *Addison-Wesley.*

Pulm U (2006) The meaning and coherence of process, organisation and products in engineering design. *SDPS Journal of Design and Process Science, 7(4), pp. 1-11.*

Rawson KJ, Tupper EC (1994) Basic ship theory. *Longman.*

Reich Y (1995) The study of design research methodology. *Transactions of the*

ASME Journal of Mechanical Design, 117(2), pp. 211.

Rolls-Royce (2005) The jet engine.

Rolls-Royce Group (2007) http://www.rolls-royce.com/ *28 February 2007.*

Roozenburg NFM, Cross N (1991) Models of the design process: integrating across the disciplines. *ICED'91, Zürich, Switzerland.*

Roozenberg NFM, Eekels J (1995) Product design: fundamentals and methods. *John Wiley.*

Safoutin MJ (2003) A methodology for empirical measurement of iteration in engineering design processes. *PhD thesis, University of Washington, USA.*

Scheer A-W (2000) ARIS - Business process frameworks. *Springer.*

Smith RP and Eppinger SD (1997) Identifying controlling features of engineering design iteration. *Management Science, 43(3) pp. 276-293.*

Steward DV (1981) The design structure matrix: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management, 28(3), pp. 71-74.*

Ullman DG (2003) The mechanical design process. *McGraw-Hill.*

Ulrich KT, Eppinger SD (2003) Product design and development. *McGraw-Hill.*

Vajna S (2005) Workflow for design. *In: Design Process Improvement - A Review of Current Practice. Clarkson PJ and Eckert CM (eds.), Springer.*

Wynn DC, Clarkson PJ (2005) Models of designing. *In: Design Process Improvement - A Review of Current Practice. Clarkson PJ and Eckert CM (eds), Springer.*

Wynn DC, Clarkson PJ, Eckert CM (2005) A model-based approach to improve planning practice in collaborative aerospace design. *Proceedings of ASME IDETC/CIE 2005, Long Beach, California, USA.*

Wynn DC, Eckert CM, Clarkson PJ (2006) Applied Signposting: A modelling framework to support design process improvement. *Proceedings of ASME IDETC/-CIE 2006, Philadelphia, Pennsylvania, USA.*

Wynn DC, Clarkson PJ (2006) P3 Signposting users' guide. *Draft document version 0.32, December 29th 2006. http://www-edc.eng.cam.ac.uk/p3.*

Wynn DC, Eckert CM, Clarkson PJ (2007) Modelling iteration in complex product development. *16th International Conference on Engineering Design (ICED'07), Paris, France (forthcoming).*

Yassine AA, Joglekar N, Braha D, Eppinger SD and Whitney D (2003) Information hiding in product development: the design churn effect. *Research in Engineering Design, 14(3), pp. 145-161.*

Zachman JA (1987) A framework for information systems architecture. *IBM Systems Journal, 26(3).*

Errata

The following errata have been corrected since this dissertation was approved for the Ph.D. degree on October 23, 2007:

Page v.	En-dash characters ("–") replaced by em-dash ("—").
Page 1.	"DTI (1992)" replaced by "DTI (1994)".
Page 52.	"in-situ" italicised.
Page 98.	"in-situ" italicised.
Page 95.	"convergence/refinement" italicised.
Page 100.	"convergence/refinement" italicised.
Page 110.	"convergence/refinement" italicised.
Page 113.	"convergence/refinement" italicised.
Page 193.	"convergence/refinement" italicised.
Page 199.	"...product to product..." replaced by "...project to project..."
Page 237.	DTI (1994) inserted.

DCW, November 30, 2007.