A knowledge-level model for concurrent design.

BARKER, Robin.

-9 AUG 2005
5pm
29 SEP 2006 6pm
06 Jun 085pm

A knowledge-level model for concurrent design

Robin Barker BEng (Hons), MSc

A thesis submitted in partial fulfilment of the requirements of
Sheffield Hallam University for the degree of Doctor of
Philosophy

March 2002

Abstract

The concurrent approach to engineering design, concurrent design, implies that expert knowledge regarding a number of different downstream life-cycle perspectives (such as assembly, manufacture, maintainability etc) should all be considered at the design stage of a product's life-cycle.

Extensive and valuable work has been done in developing computer aids to both the design and concurrent design processes. However, a criticism of such tools is that their development has been driven by computational considerations and that the tools are not based on a generally accepted model of the design process. Different models of design have been developed that fall into a number of paradigms, including cognitive and knowledge-level models. However, while there is no generally accepted cognitive model describing the way designers and design teams think, the concept of the knowledge-level has enabled a more pragmatic approach to be taken to the development of models of problem-solving activity.

Different researchers have developed knowledge-level models for the design process, particularly as part of the *Common*KADS methodology (one of the principal knowledge-based system development methodologies currently in use). These design models have significantly extended design thinking in this area. However, the models do not explicitly support the concurrent design process.

I have developed top-down knowledge-level models of the concurrent design process by analysis of published research and discussions with academics. However some researchers have criticised models for design that are not based on analysis of 'real–life' design. Hence I wished to validate my top-down models by analysing how concurrent design actually occurs in a real–life industrial setting.

Analysing concurrent design activity is a complex process and there are no definitive methodological guidelines as to the 'right way' to do it. Therefore I have developed and utilised a novel method of knowledge elicitation and analysis to develop 'bottom-up' models for concurrent design. This is based on a number of different approaches and was done in collaboration with a number of different design teams and organisations who are engaged in the concurrent design of mechanically based products.

My resulting knowledge-level models are an original contribution to knowledge. They suggest that the concurrent design process consists of a number of discrete sub-tasks of propose, critique and negotiate. These models have been instantiated as generic model templates, using the modelling formalisms specified by *Common*KADS. These models have been implemented on a software tool, the *Common*KADS workbench, in order to provide support for developers of computer-based systems for concurrent design.

Table of contents

List Of Figures

List Of Tables

Declaration

I declare that while registered as a candidate for the University's research Degree, I have not been a registered candidate or enrolled student for another award of the University or other Academic or Professional organisation. I further declare that no material contained in this thesis has been used in any other submission for an academic award.

1 Introduction to the research

1.1 Introduction

Design is a multi-faceted activity spanning many domains, including the different engineering disciplines, software and architectural design. Computer-based tools have been used successfully to support the design process for a number of years. In some cases, tools such as CAD systems have become indispensable aids to the designer.

The ambitious aim of many in the artificial intelligence and design communities is to develop computer-based support for designers that either supplants the designer in some way or aids the designer in the process of design itself. Given the expertise and design knowledge currently utilised by human designers, this is clearly a very ambitious and potentially difficult undertaking.

Concurrent engineering is an approach that is widely used in industry to reduce product lead times, improve product quality and reduce costs. The design process associated with a concurrent engineering approach is often termed 'concurrent design' (Finger et al [1992], Hernandez et al [1991]). A key aspect of concurrent design is that knowledge of downstream life-cycle perspectives is brought to bear on the design process.

This thesis is concerned with developing our understanding of the concurrent design process based upon an analysis of concurrent design behaviour. In the remainder of this introductory chapter I outline the background to the research and identify the objectives in the work undertaken and highlight the original contributions that I have made.

I start by giving a general background to the research, my interest in computer support for concurrent design and the modelling of the concurrent design process.

1.2 Research programme origins

The background to this research stems from work done on a separate project to develop a computer aided learning (CAL) tool to familiarise engineering students with the concepts of concurrent engineering and in particular, the design process implied by a concurrent engineering approach, that is, concurrent design. The CAL tool, [*]Design Builder, features a prescribed, staged model for the design process. This is outlined in Figure 1.1.

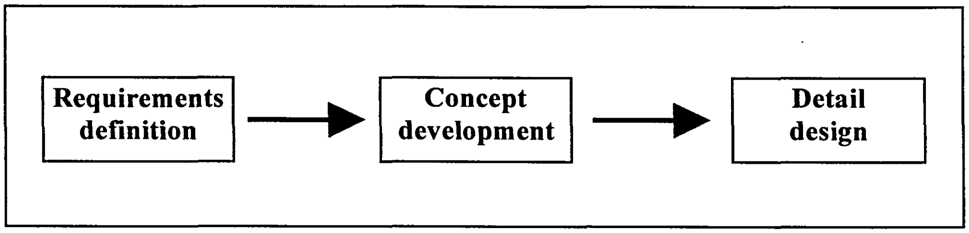


Figure 1.1: The design process inherent in the 'Design builder' computer aided learning tool.

This utilises a three-stage design process of requirements definition, followed by the development of concepts and the subsequent detail design of a selected concept.

---

[*] Details of the CAL tool, named 'Design Builder', and it's development are outlined in more detail in Parkinson and Short [1994], Barker et al [1995], Barker et al [1996a] and Barker et al [1996b]. The development of the tool was funded as part of the UK governments' TLTP (Teaching and Learning Technology Programme) initiative. The rationale behind this project was to encourage the development and take up by UK Universities of CAL technologies in order to improve the student learning experience and reduce the reliance on traditional lecture and tutorial based teaching. The TLTP project has been partially successful in this aim.

The 'Design Builder' software places the student or group of students in the role of a designer working for a company producing mechanically based products, which must be manufactured and assembled in some way. The software leads the student through a structured design process, from the development of a suitable specification through to the detailed design of the product. The design process incorporated in the software was influenced by established engineering design texts such as those by Pugh [1991], Pahl and Beitz [1984] and Hubka [1982]. These texts advocate an essentially prescriptive approach to the design process whereby a designer or design team produce a design, via a number of prescribed steps or tasks. 'Design Builder' is now being used extensively in UK higher education establishments.

However during the development of the software and the resulting discussions with design teachers and designers working in industry, it became clear that there is no consensus about a widely established, prescriptive model of design that designers use. This realisation, coupled with an interest in general knowledge-based computer support for the design process (and in particular the concurrent design process implied by a concurrent engineering approach) led me to an exploration as to how concurrent design occurs and the methods and techniques used by designers and design teams operating in a concurrent manner.

The aim of the exploration was to analyse and model the process of concurrent design, both when practiced by single designers and by design teams. The research is situated within the context of engineering design. As I have already noted, design as a process spans a wider range of domains. As a result, while I focus on the domain of engineering when discussing and analysing existing literature, I also reference existing literature from a wider selection of domains where relevant.

1.3 Modelling design

Before considering models for design, it is interesting to document some perceptions about what design is and outline some of the definitions that currently exist to describe the design process.

1.3.1 Some definitions for design

- Chern [1991] suggests *"Design is a process in which the designer builds artefacts to satisfy given specifications"*.
- Ertas and Jones [1993] state *"Engineering design is the process of devising a system, component or process to meet desired needs"*.
- Chandrasekaran [1990] suggests *"the solution to the design problem consists of a complete specification of a set of components and their relations that together describe an artefact that delivers the functions and satisfies the constraints"*.

- Brown and Birmingham [1997] *"we loosely interpret design to be an information processing activity that creates a description of an engineered artefact (for example a building or a software module)".*
- *'The most essential design activity, therefore, is the production of the final description of the artefact* ' - Cross (1989).
- Top and Akkermans [1994] suggest *"Design means conceiving a set of structural properties such that the required behavioural properties are realised".*
- Kruger and Wielinga [1993] suggest a design problem is specified in terms of functional requirements, non-functional requirements and constraints. *"A solution is a description of a set of components and their interrelations that satisfy the requirements and constraints".*

It must be noted that these definitions are drawn mainly from the fields of engineering and artificial intelligence (AI.). As Lawson [1990] states *"In different contexts the word design can represent such varied situations ... an engineer may be said to design a new gearbox for a car while a fashion designer may be said to design a new dress"*. However, as my research is based in the fields of engineering and AI, these definitions have acted to form my views and the research is based on design in this context

It must also be noted that the output from the design process is not the finished artefact. It is a description in some terms of the final artefact or component. For example in the engineering field this could consist of a set of engineering drawings with additional material, such as a specification of a set of components and their assembly plan. Different domains will have their own terminologies or ontologies to describe a design.

Design can also be seen as an iterative two-stage process of requirements definition followed by solution generation. Bernaras and van de Welde [1994] characterise these two stages as analysis and synthesis. The synthesis stage, where design solutions are generated, forms the basis of this research.

1.3.2 Concurrent design

The given definitions of design focus on the generation of a description of an artefact (or a sub-component comprising a complete artefact) as being at the core of the design process. As we will see in the next chapter, concurrent design can be seen as a particular specialisation or form of the design process and a number of different definitions for the process of concurrent design are evident in the literature. The term concurrent engineering or concurrent design started to come to prominence in the available literature from the early 1990's onwards. However, it can be convincingly argued that pracrtitioners of design have been utilising this form of design prior to this time, see Jo et al [1991].

My views as to what constitutes concurrent design (and what differentiate it from more 'conventional' design) were greatly influenced by a number of texts, including Pugh [1984], Desa and Schmitz [1991], Evans [1991]. In addition, discussions with academics and engineers working in industry helped to crystallise these views. Essentially, I take concurrent design to be a form of design where the consideration of downstream perspectives (such as manufacture, assembly, maintenance and reliability) are considered, and thus have a considerable influence, at the design stage of a products' life-cycle. Where I feel concurrent design differs from more traditional design is in the process model that is utilised to reach the description of some artefact.

In order to help form my views on how the process of concurrent design can occur, it was necessary to consider a number of different models for the design process and attempt to reconcile these with my views on how concurrent design might occur.

1.3.3 Paradigms to describe design

Cognitive models for design attempt to model the actual cognitive processes utilised by human designers when they work. However a number of researchers have outlined some of the fundamental difficulties in trying to analyse cognitive processes in humans. While extensive research is underway (see Lloyd and Scott [1994], Kolodner [1996] and Visser [1996]), Smithers [1996] believes there is currently no generally accepted cognitive model for the design process.

A number of procedural models for the engineering design process have been advocated by different researchers. These include models from Hubka [1982], Pahl and Beitz [1984] and Pugh[1991] and are expanded on in the next chapter. Typically, these models prescribe a series of stages via which the design process proceeds, from the specification of design requirements through to the detail design of the finished artefact. As such, they present a high-level, prescriptive model for the design process. Such prescriptive models, while being generally accepted methods in the wider engineering community, have their critics in the more general field of design. Cross [1989] outlines why these models can be too problem oriented, in that they focus on the requirements of the given problem definition at the expense of the solution.

A paradigmatic difference observed in the researched models for design is that between these so-called prescriptive models and those based on a 'reflection-in-action' model, usually attributed to Schon [1991]. Schon argues that at a lower level of detail, the design process (that is, the actions of personnel involved in the design process) proceed via a process of reflection-in-action. Joseph [1996] presents a comprehensive critique of the contrasting 'prescriptive' and reflection-in-action' views of the design process.

The field of artificial intelligence also presents a number of models for how both the design and concurrent design process can be modelled. In particular, the work of Chandrasekaran [1990] has been very influential in this area. Chandrasekaran argues that the design process proceeds via a number of lower level processes, or tasks, of propose, critique and modify. Hence, while previously we have considered design as being the process whereby design solutions are proposed or generated, Chandrasekaran makes explicit two further processes or tasks, critique and modify, which are of importance. From a concurrent design perspective, the process of critiquing was seen as being where the views of different downstream life-cycle perspectives could be brought to bear on the overall design process.

1.3.4 Formal process models for design

A number of the prescriptive models for design have been semi-formalised using different modelling methodologies. In Europe, the development of knowledge-level models of expertise has been formalised in a knowledge-based system development methodology called *Common*KADS (Breuker and Van de Welde [1994], Schreiber et al [1994a]).

Modelling at the knowledge-level can be seen as a more pragmatic approach to modelling expertise for subsequent implementation in a computer-based system. The idea of a 'knowledge-level' is believed to have first been put forward by Newell [1982]. Smithers [1996] contrasts Newell's view of the knowledge-level with that subsequently used in knowledge engineering methods, such as *Common*KADS. Smithers goes on to state that knowledge-level models *'make no cognitive claims, that is, they do not attempt to say what the human problem-solving behaviour must be'.* I will discuss the concept of the knowledge-level in more detail in Chapter 3. See also Newell [1982], Steels [1990]). The way in which modelling at the knowledge-level is incorporated in the *Common*KADS methodology is described by Van de Welde [1993]. I was greatly influenced by work that has been done as part of the *Common*KADS methodology.

*Common*KADS is designed to assist developers of knowledge-based systems in the specification of knowledge-based systems. In order to achieve this aim, *Common*KADS incorporates a library of different problem-solving methods, which can be applied to different tasks. The intention is that these models act as templates in a support-library for developers of knowledge-based systems in a variety of different domains and for a number of different problem types. Design is one of the problem types which is documented and supported in the library. These models have been derived from work by earlier researchers, including Chandrasekaran [1990], and have been clearly influenced by the prescriptive design methods I have discussed.

However, analysis of the existing models in the *Common*KADS library suggest that while they provide a considerable level of support for what I term 'traditional' design, they do not specifically provide support for the process of concurrent design. Smithers [1990] argues, convincingly in my view, that knowledge-based systems to support processes (such as concurrent design) should be based on accepted, knowledge-level models of the process.

It also became apparent that a considerable number of computer-based tools, in the form of research-based systems and more commercially driven products have been developed to support the concurrent design processes. However few, if any, are explicitly based on a commonly accepted model for the concurrent design process. In fact, as will be shown in Chapter 3, the development of such tools can be seen as being driven more by technological issues than more fundamental methodological considerations.

Hence I began to see that a formally described, knowledge-level model for the process of concurrent design would be a major asset to developers of computer (particularly knowledge-based) systems to support the process of concurrent design. However, a criticism of existing models developed to support the design (and other processes) is that they have resulted from researchers thinking and reading about the processes, as indeed my colleagues and I had been, rather than experimentally observing and analysing pracrtitioners in the given domain.

1.3.5 Experimentally recording and analysing expert behaviour

One of the traditional bottlenecks in the development of knowledge-based systems has been how to actually elicit and acquire knowledge from human expert sources. The field of knowledge elicitation is concerned with the analysis and acquisition of knowledge for subsequent incorporation in computer-based systems and is a huge field in its own right. Wielinga et al [1990] discuss some important issues in this field. Because of it's mutli-disciplined nature, concurrent design presents a number of challenging issues to the knowledge engineer wishing to analyse expertise in this area.

An analysis of the literature revealed a large number of different techniques and methods that are available to the knowledge engineer to attempt to model this behaviour. However, there are, as yet, no definitively agreed guidelines as to how this is best done and what techniques are most suited to different tasks (*Common*KADS also expands on the methods available in the literature and prescribes a number of different methods for deriving models of human expertise).

Two distinct approaches to knowledge-level model development are top-down and bottom-up modelling. Via top-down modelling, a model of expertise is first hypothesised and then compared with gathered data to refine and verify the hypothesised model. Bottom-up modelling involves deriving models from the gathered data without any preconceived ideas of the form of the eventual model. As will be shown, both approaches have their own strengths and weaknesses.

Based on this introductory discussion, the overall aims and objectives of the research can now be outlined.

1.4 Aims and objectives of the research programme

The overall aim of this research is:

'The development of a knowledge-level model for concurrent design'

This model will aid developers of knowledge-based systems intended to support the concurrent design process. In order to achieve this overall objective, a series of intermediate objectives is implied:

- To develop a top-down, theoretical model for concurrent design based on the available literature.

- Develop a bottom-up model for concurrent design based on an evaluation of a number of different case studies, involving organisations and individuals involved in the concurrent design process.

- Relate and validate the two models.

This series of objectives implied other objectives that must also be fulfilled:

- Determine the formalisms to be used for the modelling.
- Determine the method by which the bottom-up model is to be constructed.
- Identify case studies and contexts.

1.5 Achievement of the objectives

The attainment of the aims and series of objectives outlined in the previous section is now discussed.

1.5.1 The modelling formalisms used

In this chapter I have discussed a number of different paradigms for modelling processes or tasks, in particular the design process. In order to model an process in some way implies that some formalism is available or must be developed to actually structure the form of developed models. *Common*KADS provides a number of modelling formalisms for this purpose. Essentially these comprise graphical structures which are used to model knowledge and knowledge processes. Because *Common*KADS is the principal knowledge-based system development methodology currently in use, it was a logical choice to make use of the formalisms provided by the methodology. Chapter 3 discusses the existing *Common*KADS models for design and also outlines the modelling formalisms used by the methodology.

1.5.2 Determine the methodology for model construction

As I have discussed, a number of techniques and approaches have been used to analyse and model the design process. However, there is no concensus as to the 'correct' approach to use. The conclusion I came to, having researched and analysed these different techniques and methods, was that a novel method for observing and analysing expert behaviour in the context of concurrent design was required. Chapter 5 outlines the thinking behind the development of this novel method, which is partly based on a number of the techniques prescribed by *Common*KADS.

1.5.3 A top-down, knowledge-level model for the concurrent design process

I have already outlined how research in a number of different areas had acted to form and refine my views on how the process of concurrent design occurs. Based on the outlined methodology, the next step was to use these findings and the results of my informal discussions with academics and practicing engineers as input to the development of a top-down model of how concurrent design occurs. The term top-down is used to describe model development as it happens from deriving a general overview of the problem area. A more bottom-up form of model development would proceed by using empirical, experimentally derived data to drive model development. These issues are expanded on in more detail in Chapters 5 to 7.

This initial knowledge-level model for concurrent design uses the graphical notation advocated by the *Common*KADS methodology. The work of Chandrasekaran and his view of design as a process of propose-critique-modify was particularly influential in the development of this initial model. The process of propose-critique tallied with my view of how concurrent design could occur. However, I did not believe the modify process fully captured the essence of how contrasting critiques of a design would influence the process.

When different downstream perspectives critique a given design solution, these critiques will often conflict. Hence there is clearly some intermediate process where these possibly different critiques are reconciled. I began to see how important this process, which I term negotiation, could be to the overall concurrent design process. Hence my initial model for concurrent design consisted of a series of sub-tasks of propose-critique-negotiate. Based on the available literature, various plausible models for how the propose and critique tasks could be modelled were considered. However, no detailed model for the negotiation process was available.

Hence by this stage, after extensive research of the literature and discussions with academics and practicing engineers, a knowledge-level model for the concurrent design process had been developed in a top-down manner. This was extensively modified and refined before arriving at the final model. The details of this exploratory process are outlined in Chapter 7. Having developed this model in relative isolation, I decided to publish the findings of the research to a wider audience. This resulted in a number of papers describing this initial model, see Barker et al [1996a] and Barker et al [1996b]. This culminated in a more comprehensive paper describing the model development, Barker and Meehan [1999].

1.5.4 Case studies

A bottom-up modelling approach depends on analysing experts 'in the field'. I began to look for a number of case studies which could be used to derive experimental data based on the methodology I had developed. Using contacts in the field of engineering, a number of different designers and companies agreed to me analysing their design methods in a work place setting. Details of the case studies are outlined in Chapter 6.

1.5.5 Bottom-up modelling with the different case studies

Having developed a top-down model for concurrent design, I then used the developed model as a basis for discussing and experimentally observing designers and personnel involved in the concurrent design task in an industrial setting. The resulting transcripts from interviewing and observing these personnel were then used to drive the development of more bottom-up models of the tasks associated with concurrent design. These bottom-up models expanded on the original top-down model as well as giving more detail as to how certain tasks were undertaken.

The results from this lengthy period of analysis and bottom-up modelling were a series of task models, based on *Common*KADS formalisms, showing how the process of concurrent design occurred in the different case studies. Chapters 8 and 9 outline this process in more detail and also outline some of the problems inherent in attempting to analyse expert behaviour in an industrial setting.

1.5.6 Generality of the models

The expertise task models that resulted from the case studies exhibited a large number of features in common. However, certain characteristics were specific to a particular case study. What I wished to do was to derive more general, or generic models from the bottom-up models of the case studies and also account for the differences between models. Chapter 10 outlines how generic models of expert problem-solving behaviour (and also the organisational context in which the problem-solving behaviour took place) were derived.

1.5.7 Instantiating my generic models on the *Common*KADS workbench

The *Common*KADS workbench is a software tool, which supports the development of the different models comprising the *Common*KADS model set (see Kingston et al [1995]). To this end, the workbench provides considerable support for the development of the different models comprising the full *Common*KADS model set. Up to this point, the main focus of the research had been on the expertise modelling elements of the *Common*KADS model set. This chapter goes on to investigate issues such as modelling the organisational context in which the models could be implemented.

This tool was used to instantiate the model templates supplied as part of *Common*KADS with details of the generic models derived from the case studies. Using the workbench in this way ensured the models comply with the formalisms of the methodology and also provide a series of templates, in a useable format. The intention is that these are available to inform developers of systems to support concurrent design.

The instantiation of the generic models using the *Common*KADS templates is described in Chapter 11.

1.5.8 Validating the model

In order to validate and refine these generic model templates and also to illustrate how the models could be used, I have instantiated the generic models with details of two contrasting scenarios. The first of these scenarios relates to a situation encountered in one of my case studies while the other scenario involved re-engineering an existing expert system that has been extensively described in the literature. The findings from these instantiations helped to refine and extend the model templates. This is described further in Chapter 12.

1.6 Chapter Summary

In this chapter, I have outlined both the background to the research and the context in which the research is set and introduced a discussion of models of design. The concepts of cognitive and knowledge-level modelling are outlined as possible means of modelling concurrent design activity. *Common*KADS, which uses a knowledge-level modelling approach, has been introduced as providing suitable modelling formalisms for the development of models of human problem-solving expertise.

The following chapters will now give a detailed account of the development of a knowledge-level model for the concurrent design process and how a novel knowledge elicitation method has been used to elicit the knowledge, which has driven the model development.

2 Models for design

2.1 Introduction

In this chapter I begin by discussing the main characteristics of concurrent design and in particular, what differentiates this type of design from what can be seen as more 'traditional' design. I also discuss how concurrent design has been implemented in different organisations.

Design is a wide ranging activity applied in many different domains, both in engineering (mechanical, electrical, civil etc.) and other areas including software, architectural and product design. However at an abstract level, design in these different domains can be seen as having many characteristics in common. Hence design can be seen as being generic in nature and largely independent of the domain in which it is practiced.

The process of design can be broken down into a number of more discrete processes or sub-tasks. In particular, design can be seen as two distinct process of requirements formulation and solution generation. These are sometimes termed the analysis and synthesis stages of design. In this research I focus mainly on the synthesis stage of the design process.

A number of paradigms exist for describing the design process. These include the so-called prescriptive models and the rational pracrtitioner approach. In addition, cognitive-level and knowledge-level modelling are presented as two approaches to modelling design processes. This chapter goes on to present existing models for design (focussing on the design synthesis task) and the various sub-tasks associated with this process. These have been derived from research in a number of diverse domains although the main focus is on those models developed within the engineering fields.

2.2 The importance of the design process

The importance of design to the overall life-cycle of a product can sometimes be underestimated. Victor et al [1993] state that in general, 70 - 85 % of product cost is determined at the earliest stages of product design. Cha and Guo [1993] look at the effect of design decisions on the overall life-cycle cost of a product and suggest that 70% of the total life-cycle cost of a product will be determined at the design stage, a view previously voiced by Andreasan et al [1983] *'upwards of 70% of a product's manufacturing cost is dictated by design decisions'.* Hence any approach which can pinpoint problems at this stage will result in products being brought to market *"at lower price and in significantly less time"* - Jo et al [1991].

In addition, the degree of commonality shown by the process of design across a wide variety of domains has been commented on by a number of researchers including Lloyd and Scott [1994] who state *"...research suggests that we can identify common behaviour in designers regardless of discipline".* Brown and Birmingham [1997] indirectly assume this when they see design as a process that *"creates a description of an engineered artefact, for example a building or a software module".* Candy and Edmonds [1997] also support this view when they state: *"In both product design and software design, characteristics in common have been identified".*

2.3 The product life-cycle

The life-cycle of a product can be defined as the complete life span of a product from its' initial inception through design, manufacture, assembly, use and ultimately the disposal of the product.

In the craft type industries that abounded during the era before the industrial revolution, a single person would typically be responsible for the complete life-cycle of a product. A craftsman would liase with the customer or analyse the market place in order to determine the requirements of the product. The same craftsman would then be responsible for the design, manufacture, assembly and packaging of the product before delivery to the customer. Even after delivery, the craftsman could be responsible for the maintenance, repair and disposal of the product. Hence the craftsman would be responsible for a considerable part of the life-cycle of the product and through this involvement would have considerable knowledge of these different life-cycle aspects.

However, with the advent of mass production, this approach to manufacturing became untenable. Mass production meant that individuals would only be responsible for a very small part of the life-cycle of a product. As a result, the person or personnel responsible for the design of a product would not have in depth knowledge or responsibility for other life-cycle aspects of the product. This approach has been termed the 'serial' or 'over the wall' approach to product design. Via this approach, a product could be designed, then manufactured and then assembled with relatively little interaction between the different disciplines. The literature abounds with cases whereby products have been designed which are impossible or at the least very difficult to manufacture, assemble, service and maintain. For example, Bull [1993] describes how the Ministry of Defence estimated that unscheduled maintenance costs of defence equipment exceeded 1 billion pounds in 1990.

Because of the inherent problems associated with the serial approach to product design, products could have long lead times, be of indifferent quality and expensive. In the earlier part of this century, when consumer products were in relatively short supply, a seller's market dictated that such problems did not seriously harm a product's marketability. The preoccupation was in using mass production techniques to satisfy an expanding market. However increasingly sophisticated mass production techniques meant that the shortage of consumer goods did not last. As the marketplace became saturated, a buyer's market began to dictate that products should be of high quality, low cost and modern in design.

The limitations of this serial approach led to the development of the philosophy of simultaneous engineering. Simultaneous engineering grew out of a recognition that the traditional serial (or sequential) approach to the design and manufacturing process has serious drawbacks when applied to the modern day product market place. To quote Hedberg [1994] *"The days are past when design engineers brewed up a design entirely on their own and then 'threw it over the wall' to manufacturing."*

Initial definitions of simultaneous engineering concentrated on the simultaneous design of the product and it's associated manufacturing processes. Sohlenius [1992] characterises it as *"a way of work where the various engineering activities in the product and production development process are integrated and performed as much as possible in parallel rather than in sequence".* However, Bayliss et al [1994] state that *"the philosophy is the simultaneous involvement of suppliers and customers at an early stage in the design process".*

As Pugh [1991], Finger at al [1992], Young and O'Grady [1994] and Evans [1991] point out, there are other considerations, apart from manufacturing and the customers' and suppliers' views, which it is important to consider at the design stage. These include assembly, maintenance, cost, marketing and safety issues, indeed any issue that affects the complete product life-cycle. While simultaneous engineering implies developing a number of processes, particularly manufacturing processes, in parallel with design, concurrent engineering can be seen as a more all-encompassing philosophy which allows the consideration of a number of downstream life-cycle aspects at the product design stage.

2.4 Concurrent engineering (CE)

Concurrent engineering (collaborative and parallel engineering are analogous approaches) is a philosophy that is being used increasingly in industry to reduce costs and improve the quality of manufactured products. *'Only by becoming more flexible and thereby more responsive to market demand, while at the same time maintaining product quality, will companies be able to remain competitive. Concurrent engineering is a major step towards achieving this goal'* – Stevenson [1994]. This chapter gives a concise review of the main principles and issues involved in the concurrent engineering approach. Jo et al [1991] and Prasad [1996] give a comprehensive review on the principles of CE while Ranky [1994] outlines which organisations are adopting this approach.

No single definition for concurrent engineering has been found, rather there are a number of alternative definitions and resulting implementation issues to consider. I will now outline some of these definitions in order to determine the key characteristics of the approach.

'Concurrent engineering is the philosophy which realises the importance of quality, communication and the parallel design of the product and the processes that affect it throughout it's life-cycle' – Bayliss et al [1994].

'Concurrent engineering has recently been recognised as a more integrated approach to developing high-quality products and bringing them to a highly competitive global market at lower price and in significantly less time'. Jo et al [1991].

'... a number of different aspects of a product are engineered simultaneously or concurrently'. Kott et al [1990]. Carter [1994] also characterises this approach and states that *"downstream manufacturing and support processes are identified early in the product development cycle and addressed, along with product design".*

The concurrent engineering approach has evolved *'... from the recognition that design decisions made early in the product development life-cycle have significant impacts on manufacturability, quality, cost, time-to-market and thus ultimate market-place success of the product'.* Jo et al [1992].

'Concurrent engineering calls for site-specific, simultaneous evaluation of manufacturing, cost and other performance measures early in the design process'. -Thurston [1993].

From these definitions it is clear that the design process and different product life-cycle perspectives are key factors within concurrent engineering. Concurrent design can be defined as the design process practiced in a concurrent engineering context. In order to define what concurrent design is, it is necessary to consider how different life-cycle perspectives affect the design process within a concurrent approach.

2.5 Concurrent design and life-cycle aspects

Pugh [1991] promotes the Quality Function Deployment method, which emphasises the importance of 'the voice of the customer' in the design process. Sonnenwald [1996] states that concurrent design includes *"participants from different domains who must explore and integrate their specialised knowledge"*. Shiva Kumar et al [1994] suggest *"in the field of collaborative engineering ... several life-cycle issues of the product have to be incorporated at the time of design"*. Darr and Birmingham [1994] state *that "such parallelism helps to identify design conflicts early, avoiding iterations that could arise in the serial approach"*. Rajeev et al [1993] echo similar findings from the civil engineering industry - *"the repercussions of not communicating information between agents often takes the form of escalated costs and costly rework"*. It is important to note that the relative importance of different life-cycle perspectives will differ from product to product.

The concurrent design approach can be compared to other types of design such as collaborative and co-operative design found in the literature, for example Sonnenwald [1996]. However while co-operative and collaborative design (by their very definitions!) assume some form of co-operation and common goals amongst the different participants, this is not necessarily the case with concurrent design. In an idealised concurrent design environment, the overall goal of the different participants would be the development of the overall design. However, this is not always the case as will be later illustrated. It is this defining factor that I believe distinguishes concurrent design from what will be termed these more co-operative modes of design.

2.6 Concurrent design and the use of 'multi–disciplinary' knowledge sources

Desa and Schmitz [1991] outline a concurrent engineering method and suggest *"concurrent design implies the use of multi disciplinary design teams who interact during early design"*. They advocate a method which they term 'Virtual Concurrent Engineering'. Via this approach, a designers decisions are subject to evaluation by different downstream life-cycle perspectives (they concentrate on the manufacture and assembly aspects). They go on to describe a software tool, the 'Producibility Evaluation Package' (for analysing designs from a manufacture viewpoint) which has been used to evaluate the Virtual

Concurrent Engineering' approach. They conclude that the design resulting from such an approach will be globally optimal when performance, producibility, assembleability, reliability, serviceability etc are considered. However, they do not discuss how conflicts between different life–cycle aspects may be handled. The designer has control over the process and decides which evaluative advice to heed.

Schmidt and Schmidt [1995] support this emphasis on teamwork by stating *"Concurrent engineering is the engineering process which results from solving an engineering problem with the help of a team"*. Wheeler [1991] also looks at the use of 'multi disciplinary teams' in concurrent engineering while Klein and Lu [1989] investigate problems of conflict and conflict resolution strategies that can arise from the use of such teams.

However whilst concurrent design is frequently implemented in practice through the use of teams, the process of concurrent design does not necessarily imply that more than one person be involved in the design process. It is possible, in principle, for a single designer to be able to bring knowledge of a number of different life-cycle perspectives to bear on a given design problem.

Hence a suitable definition would define concurrent design as 'the consideration of all downstream activities which are likely to affect the product's life-cycle at the product design stage'. In a typical engineering based context, these would include issues such as manufacture, assembly, costs, materials and marketing as well as considerations such as links with suppliers. This definition allows for other perspectives, which will affect the product life-cycle and assumes multi-disciplinary teamwork will usually (but not always!) play an important part in the process.

In an idealised concurrent design scenario, a designer or design team (i.e. personnel whose main expertise is in the area of design) will work on the day to day design of a product and will specify most of the design. They are able to bring in 'experts' in different areas when required, who are able to give their own perspective on an evolving design. These 'experts' bring in 'design for X' (or DFX - see Dewhurst and Abbatiello [1996]) expertise in a number of different areas such as design for manufacture, design for assembly etc.

The differences between the serial and concurrent approaches to design and the associated input of knowledge are summarised in Figures 2.2 and 2.3.

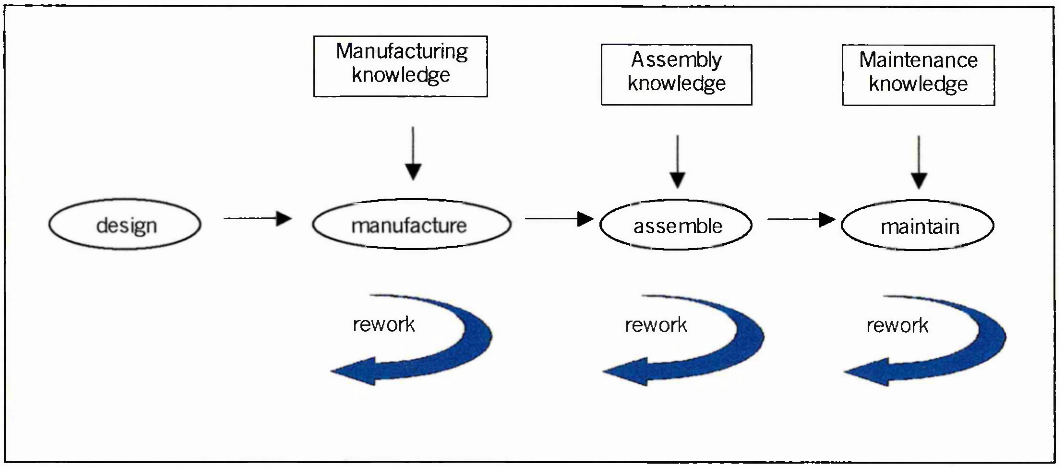


Figure 2.1: The serial approach to design and knowledge input.

This shows how potentially costly and time-consuming rework can result when life-cycle knowledge is brought to bear downstream from the design process.

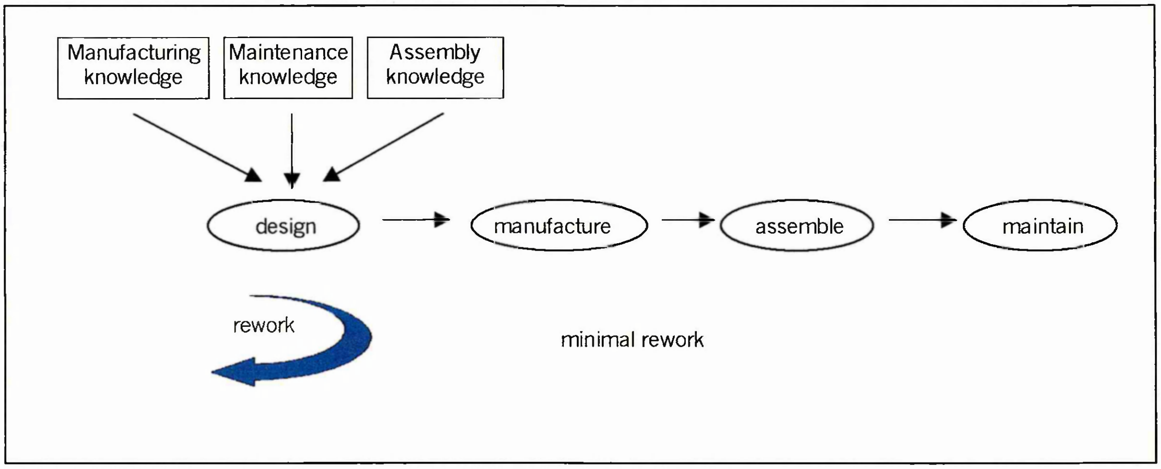


Figure 2.2: The concurrent approach to design and knowledge input.

This shows how rework can be restricted to the design process, with the input of knowledge about downstream life-cycle perspectives at this stage. This results in these downstream processes then proceeding with minimal rework.

The reliance on the use of teams to support concurrent design has resulted in major strategic changes in organisational structures. The business process re-engineering philosophy is increasingly being used to implement such changes in organisations. This approach involves identifying key processes that are vital to an organisations' success and then reorganising the company structure to group the necessary resources (especially personnel) around these key processes. This typically involves changing the traditional hierarchical, functionally-oriented organisation structure to a much 'flatter' process-oriented structure. See Spurr et al [1994], Hammer and Champy [1993] for further details. The goal of the 'Enterprise project' (Fraser [1995]) is to develop computer-based tools to support enterprise modelling and the improvement of an enterprise model.

Having analysed and discussed some of the essential characteristics of the concurrent approach, I will now outline some existing models for design and discuss the extent to which these models are relevant and applicable to the process of concurrent design.

2.7 Models of design

A number of models of design exist, generally falling into one of two paradigms. These are the *rational design approach* (Pahl and Beitz [1984], Simon [1969]) or the *reflective practitioner approach* (Schon [1991]). Joseph [1996] compares and contrasts these two views of design. It is possible to formulate models for concurrent design within either of the paradigms above. However, as far as I have been able to establish, the formalised models that have currently been implemented for design are firmly situated within the former paradigm, as will be discussed later in this chapter.

2.7.1 Rational prescriptive models for design

These rational models can be seen as models of design which prescribe a series of steps or tasks that a designer should perform during the design process, hence the term prescriptive. A number of prescriptive models of the design process are presented in the literature. Acknowledged models in the engineering field include those by Pahl and Beitz [1984], Hubka [1982] and Pugh [1984] (Cross [1989] outlines additional prescriptive models for design).

Pahl and Beitz [1984] suggest that design is either variant, adaptive or original in nature depending on how new or original the design problem is perceived to be. The actual design process is seen as progressing through requirements definition, concept generation, embodiment design (where layout and form are determined) and detail design. They suggest functional decomposition as a suitable method to use to help solve design problems.

Hubka [1982] also advocates a procedural model for design based on going from a problem assignment to a design specification from which a function structure is generated. Concepts are then generated followed by preliminary layouts, dimensional layouts and finally detail and assembly drawings. This rational or prescriptive approach has been formalised by the VDI [1987] (The German processional engineers association) into a design methodology. The VDI propose that designing should proceed in a sequence of stages. The overall problem should be decomposed into sub-problems and then solutions found and then combined into the overall solution.

Pugh's 'Total design' process (Pugh [1991]) is more analogous to the concurrent approach in that the development of the design is analysed from a number of different perspectives. However, Pugh still advocates an essentially prescriptive model for the design process in that design proceeds via the development of a design specification through conceptual and detail design.

Cross [1989] also presents some commonly used design models and argues that such systematic or prescriptive models are needed to cope with the increasing complexity of modern design problems. However, Cross also suggests such prescriptive models for design can be too problem oriented and encourages the designer to apply strategic thinking to the design process.

March's view of design (see March [1984]) is seen as being more solution oriented. March suggests that designers must generate some form of solution in order to help them think about the overall design problem. Cross [1989] extends this view by suggesting that *"exploration and identification of the complex network of sub-problems is often pursued in* *practice by considering possible sub solutions"*. However, Purcell and Gero [1996] outline how a fixation on a particular solution can hinder the evaluation of alternative solutions.

Lawson [1990] also criticises some of the more prescriptive 'maps' of design as being *"derived more by thinking about design than by experimentally observing it"*. Lawson suggests that designers in different domains adopt different approaches to the design process. Scientists and engineers typically problem solve by analysis in that they try to understand the problem as clearly as possible before formulating a solution. This is seen as a problem-focused approach to design.

However, Lawson believes designers' solve by synthesis. The formulation of possible solutions is used to improve understanding of the problem so this is a more solution-focused approach. Cross [1989] also suggests that the problem and solution development proceed in parallel - *"exploration and identification of the complex network of sub-problems is often pursued by considering possible sub solutions ...creative designing seems to proceed by oscillating between sub-solution and sub-problem areas"*.

2.7.2 The 'reflective' practitioner

Joseph [1997] and Dorst and Dijkhuis [1995] assess the limitations of the prescriptive approaches and contrast these with the approach of Schon [1991]. In particular, Joseph [1997] emphasises the lack of any real strategic planning inherent in the prescriptive approaches. The reflective practitioner approach, outlined by Schon suggests that designers (and other professional practitioners) work through a process of reflection-in-action. Design as a process is characterised as *"a reflective conversation with a unique and uncertain situation"* and occurs by a series of actions and then reflection on the consequences of these actions. Via this exploration process the designer becomes more committed to the design solution being developed. Schon characterises the designer as being *"in graphical conversation with the design"*.

Suchman [1987] suggests that when performing some task, experts do not necessarily follow an explicit, pre-defined plan. What they do is to respond to the changing environment based on tacit skills. This view is supported by Oxman [1995] who sees conceptual design as a series of transitions between states, the actual transitions or moves being governed by rules.

However, while a great deal of work has been done to formalise prescriptive models of design as design methodologies or tasks, as far as can be ascertained, no implemented model currently exists for the reflective practitioner approach to design.

Research which seeks to achieve this is ongoing and is acknowledged to be at an early stage, both in terms of determining appropriate methodologies (Davies [1995], Dorst [1995], Lloyd et al [1995]) and developing models which span design (Lloyd and Scott [1994], Oxman [1996], Kolodner [1996], Visser [1996], Sonnenwald [1996]).

2.8 Cognitive and Knowledge-level models of design

In the previous section, we have seen how the prescriptive models for design give a general, prescribed overview of how the design process should proceed. The reflection-in-action hypothesis outlines at a more detailed level the processes designers are believed to actually use. Models of how experts in a domain carry out some complex task (such as design) can be broadly split between models at the cognitive-level and models at the knowledge-level.

Cognitive models attempt to model the thought processes designers actually use when designing which can encompass the broad spectrum of cognitive activity including emotions and personal goals. Lloyd and Scott [1994] review a number of different cognitive models presented for the design process and suggest that designers utilise three main modes of thought. These are generative (where a proposal is made), deductive (where the proposal is clarified) and evaluative (where the proposal is assessed) in nature.

Condoor et al [1992] argue the case for design models based on cognitive issues. However, a number of researchers have argued that cognitive-level models for complex cognitive processes such as design currently do not exist. Smithers [1996] states '*...without a much more complete theory of human cognition, any attempt at developing and testing a theory* *of design as cognition is going to have a very difficult time ...'*. Churchland [1989] gives further arguments as to why general theories of human cognition may never be realised.

Knowledge-level models do not specifically attempt to replicate the thought processes of human designers. The idea of the 'knowledge-level hypothesis' was advocated by Newell [1982] and the importance of modelling at the knowledge level has been shown by Clancey [1983].

The concept of the 'knowledge-level' was influenced by early work on the development of expert systems and other knowledge-based computer systems. A common approach to the implementation of such systems was to encode human problem-solving expertise in the form of 'rules'. However, rules are effectively a computational structure. It became evident during the development of such systems that rules do not necessarily reflect the way in which a human expert structures their problem-solving knowledge. This drawback became particularly apparent during the knowledge acquisition phase when experts would be asked to describe their expertise, which would not necessarily be structured in a procedural-type format, in terms of rules.

In addition, re-use and maintenance of such systems often proved to be virtually impossible because of the way in which the flow of control inherent in such systems was embodied in the different rules. '*... the hope that an intelligent system could be achieved by lumping representations of knowledge fragments together in a knowledge-base and then letting a general inference mechanism (e.g. a forward chaining rule interpreter) sort out when to use those elements has failed ... programming tricks corrupt the interpretation of rules or frames as pieces of knowledge ... second generation expert systems explicitly reflect in their implementation aspects of the knowledge and their control structure.*' - Van de Welde [1994].

The knowledge-level is seen as an intermediate, implementation-independent means of modelling an expert's problem-solving knowledge. Subsequent researchers including Bylander and Chandrasekaran [1987] and Steels [1990] have expanded and refined these original ideas. The term 'knowledge-level' is used to describe some entity as if it possesses knowledge without making commitments about how such knowledge be represented or implemented in a computer system. The symbol level is taken as being the level of abstraction where representations of objects at the knowledge-level are defined in terms of symbols that can be manipulated by computer programmes.

Typically, knowledge-level models have been developed using task analysis and the resulting models are task-oriented as a result. Bakyan [1996] suggests *"task analysis produces a description of the constraints on behaviour that must be satisfied to solve the problem at a given level of intelligence where intelligence is defined as the ability to use knowledge to solve problems"*.

However, not all models or views of design are task-oriented in nature. Candy and Edmonds [1997] outline a criteria-based model for design. They also suggest that *"... we cannot assume that the user performs tasks in closely ordered, sequential and predictable ways"*. The implications of this are that any task-oriented view of design should be flexible enough to accommodate this flexibility in design behaviour.

Work on the knowledge-level extends to the field of artificial intelligence (AI). This field has also resulted in the development of a number of different models for the design process. Before discussing these models, I present a brief introduction to work in this area.

2.9 Artificial intelligence

In order for a computer-based system to simulate knowledge of processes and expertise, specialist advice must be captured and represented in some way in the system. The field of artificial intelligence (AI) is concerned with the modelling of human expertise and problem-solving and using computers as 'intelligent' problem solvers. The field of AI grew from attempts to develop autonomous 'thinking machines' capable of independent thought and problem-solving. Winston [1984] and Charniak and McDermott [1985] give a further introduction to AI and work in the field.

Newell [1990]) suggest that different tasks (such as design, planning, diagnosing etc.) are examples of a more general form of human problem-solving. A common assumption is that at a usable level of abstraction, different tasks exhibit similar characteristics when performed in different areas of applications (or domains). Hence, for example, the process whereby a piece of software is designed is believed to have many characteristics in common with engineering design.

Artificial intelligence models objects and activities in the 'real world' and when implemented in some form of computer-based system, they are typically termed knowledge-based systems.

2.10 Artificial intelligence and models for design

Two broad issues that emerge in the use of AI in design are the use of AI techniques to model artefacts and spatial relationships between the objects encountered in design and the use of AI to model the processes within design. This is articulated by Takeda et al [1990] who suggest that the representation of design knowledge is a two-part process - *"the representation of design objects and the representation of design processes".*

2.10.1 Product models for design

There are a number of issues related to the computational representation of objects associated with the design process. Balachandran and Gero [1988] look at a frame based architecture for representing knowledge relating to engineering components. Gero [1990] suggests 'design prototypes' as a suitable representation format for a design object. Taleb-Bendiab et al [1992] and Sanderson et al [1990] look at AI techniques to model the assembly of such components. Xue and Dong [1993] look at object oriented techniques to represent the product in a concurrent design environment while Ball et al [1996] describe a novel adaptation of the object-oriented approach to product modelling. However, it is the processes within design that this research is primarily concerned with.

2.10.2 Process models for design

A number of researchers within the field of AI have attempted to model design processes using techniques from artificial intelligence. As noted earlier in this chapter, I focus on the methods and techniques used for design synthesis as opposed to design analysis.

Chandrasekaran [1990] states that the design problem is *"formally a search problem in a large space for objects that satisfy multiple constraints".* However since the problem space of design for most non-trivial problems is vast, clearly designers employ methods and heuristics to limit search in this space.

Chandrasekaran [1990] goes on to describe the design process as a series of sub-tasks of propose-critique-modify and outlines a number of different computational methods utilised for the 'propose' task of design. These include case-based reasoning, decomposition-solution-recomposition and constraint satisfaction. This view is endorsed and expanded on by Maher [1990] who outlines additional methods that designers are believed to use for the design 'propose' task including the use of grammars. However, Maher [1990] goes on to outline how *"the major barrier to the application of grammars to engineering design is the lack of a formal basis for representing function".* Chandrasekaran and Maher also criticise some of the earlier prescriptive methods because of the constraints these place on designers and stress that the use of the methods they outline is done in a flexible way – *"a useful architecture is one that can invoke different methods for different sub-tasks in a flexible way"* (Chandrasekaran [1990]).

A number of the models for design analysed do consider the evaluation of a design. However this is typically after a complete design has been proposed or generated and consists of ensuring that the design meets the functional requirements of the specification. For example, Maher [1990] considers design evaluation, but this is concerned with checking how a partial or fully specified design complies with the expected performance (i.e. the function) of the design.

Chandrasekaran sees the critiquing task as being the stage where a design is evaluated. 'Critiquing' is defined as the *"subtask in which the causes of a design's failure are analysed".* I.e. it is implied that this approach is taken in order to highlight functional drawbacks in a completed design or partial design. Goel [1989] also takes a functional analysis approach to critiquing. Therefore these forms of critiquing are involved with ensuring that the completed design complies with the functional specifications required, rather than that the evolving design complies with different life-cycle constraints, which is one of the main issues considered in my research.

Chandrasekaran goes on to suggest that this approach can be used at any level of abstraction for the design process. I believe that for a true concurrent design process, this approach would be used at a very low level. For instance if a designer was to propose a complete design and then present it for critiquing, this essentially represents an iterative, serial type approach. However if this approach is adopted at a much lower level of granularity, this can represent a more concurrent approach to design.

The work of Takeda et al [1990] is also important, as they illustrate how a cognitive model of the design process can be mapped to a computational implementation, 'The Design Simulator'. Their general design theory, GDT, suggests the designer proceeds via a process of abduction, deduction and circumscription. However, this model does not give much insight into how concurrent life-cycle perspectives impinge on the design process.

Lloyd and Scott [1994] analyse the cognitive processes they believe designers use. Interestingly, they conclude that designers' cognitive processes are generative, deductive or evaluative in nature. The generative and evaluative processes can be seen as analogous to the propose and critique tasks respectively that more task-oriented research has outlined.

What is interesting to note from this analysis of influential research in the area of design is the way in which the overall design synthesis process is generally broken down into sub-processes or sub-tasks. I will now discuss the form these sub-tasks may take.

2.11 Models of the design proposal task

Chandrasekaran looks at the propose stage of the design cycle in more detail and suggests three methods by which designers propose a solution or partial solution to a problem. These are decomposition-solution-recomposition, case retrieval (a form of case-based reasoning) and constraint satisfaction. A designer will choose the most suitable method based on the design problem being considered.

Decomposition-solution-recomposition involves breaking down the design problem into a series of smaller sub-problems. This may be done at the function or component level. Solutions to the sub-problems are then generated and combined to form a solution to the original design problem. Kruger and Wielinga [1993] suggest that this is the way in which solutions to most industrial design tasks are generated. However, this method is not specifically a design proposal method. Application of the method effectively sets up a number of sub-problems (in design).

Maher [1990] adds 'transformation' to the 'propose' methods outlined in Chandrasekaran. This transformation method implies that transformational rules (similar to a grammar) operate to transform the formal specification into a detailed design. However this form of design assumes that accepted transformational grammars actually exist in the given domain. Maher points out that *"currently the models are ill defined and many issues need to be resolved before they can become domain independent formalisms"*. Gagdas [1996] also outlines a design approach based on shape grammars.

Chern [1991] defines parametric design as *"the process of assigning values to attributes which are called the parametric design variables. It should be noted that the values to be assigned are not always numeric, but could also be a type or class designation (e.g. a material type)"*. Motta and Zdrahal [1996] also present a generic model of parametric design and conclude that what is needed are flexible models of problem-solving in order to support the design process.

Hence we can see that a number of different methods may be used to generate design solutions. However, in order for different life-cycle perspectives to be able to influence the evolving design solution, these solutions must be open to some means of criticism, where potential problems are outlined. Research from the field of critiquing can give an insight into how this criticism may occur.

2.12 Models of critiquing

A number of computer-based critiquing tools, some of them from the field of engineering design, are discussed in Appendix B. Typically such systems involve a computer-based critic analysing a human user's actions. As a result, there is considerable discussion in the analysed research regarding computational aspects that must be considered when implementing such tools and also the way in which critiqung models are represented computationally. What is more of interest in this research are the underlying models of critiquing employed by such systems.

2.12.1 Definitions of critiquing

It is useful to consider exactly what a critique implies and what the essential features of critiquing are.

- 'critic' (n) one who passes judgment.
- 'critique' (n) carefully judged criticism - Concise English Dictionary.

Fischer and a number of colleagues at the University of Colorado have done extensive work on critiquing and computer-based critics in cooperative problem-solving environments and learning environments. *"Critics analyse a product and provide suggestions as to how the user can improve that product"* - Fischer and Mastaglio [1991]. Fischer et al [1993b] suggest *"critiquing is a dialogue in which the interjection of a reasoned opinion about a product or action triggers further reflection on or changes to the artefact being designed".*

"Critiquing is the process of evaluating a solution to a task and providing an appraisal of it to contribute to possible improvements" - Rankin [1993]. Rajeev et al [1993] *"Criticism involves evaluating a design in terms of it's effectiveness in satisfying design objectives*

and constraints"... "Criticism is performed by subjecting the design solution to a number of tests that determine it's degree of acceptance".

The critiquing approaches described by Chandrasekaran [1990] and Goel [1989], which have been discussed in a previous section, are subtly different from critiquing to pinpoint additional life-cycle constraints, which I believe characterises the concurrent approach.

2.12.2 Different types of critic

Silverman [1992] outlines different types of critic that have been implemented in computer-based tools. 'Influencers' work before a specific subtask to influence the user and give positive feedback. 'Debiasers' work after a subtask and give negative feedback. 'Directors' assist users with application of a cue.

Fischer et al [1993b] outline how critics can be either passive or active in nature, depending on whether advice from a critic is requested by a user or whether a critic is free to intervene whenever the critic feels a need. This is supported by Silverman and Mezher [1992] who also suggest that critiquing may be done in batch mode (after an entire solution has been generated) or incremental mode (where a user is interrupted during his or her task).

Silverman [1992] suggests that *"experiments on during versus after task critiquing have so far proved inconclusive".* However Silverman and Mezher [1992] go on to suggest that the batch mode of critiquing (utilised in 'classic' critiquing systems such as those described in Miller [1984] and Langlotz and Shortliffe [1983]) can allow a solution developer to embark on an erroneous design without critic intervention where an incremental critic would be more beneficial.

Fischer et al [1993b] characterise critics as being 'generic', 'specific' and 'interpretive'. Interpretive critics are used to view a design from different viewpoints.

However, while these different modes and type of critic give a valuable insight into the field of critiquing, they are more to do with when a critic should intervene rather than how the critic actually performs the critiquing task. They do not explicitly state or suggest how the critiquing process might occur in a task (or any other) type of manner.

2.12.3 Critiquing models and strategies

Langlotz and Shortliffe [1983] describe a critique *as "an explanation of the significant differences between the plan that would have been proposed by expert system and the plan proposed by the user"*. This implies that their model of critiquing has some proposal stage followed by a compare and contrast stage.

Miller [1984] considers critiquing by local or global criteria in the analysis of a medical treatment plan based on patient symptoms. Critiquing by local criteria implies only a small sub-set of locally related parameters need to be considered. However, global plan critiquing implies that a much more comprehensive overview of the plan needs to be considered.

Fruchter et al [1993] *"Critiquing entails analysis and evaluation of the design. In the analysis stage, the performance of the design is predicted. In the evaluation step, the derived performance is compared against the requirements".*

Fischer et al [1991b] describe the possible sub-processes of critiquing as goal acquisition, product analysis, critiquing strategies, adaptation capability, explanation and argumentation and advisory capability. They outline two possible models for critiquing, the differential and the analytical approach. In the differential approach, the critic generates it's own solution and then compares it with the given solution, pointing out differences. This technique works best where there is a single optimum solution. However where a number of radically different but equally valid solutions can exist, this technique has limitations.

Analytical critiquing involves finding sub-optimal features in a given solution. In this critiquing mode, a critic does not need a complete understanding of all aspects of the given solution in order to critique from a particular perspective.

Rajeev et al [1993] view the design process as a series of synthesis-critique-modify steps (analogous to the propose-critique-modify approach of Chandrasekaran). They suggest criticism is performed by subjecting the design solution to a series of tests that determine its degree of acceptance. This issue of a quantitative or qualitative assessment as to the degree of acceptance of a solution to a critic has also been noted by other researchers. Rajeev et al [1993] *"From the computational point of view it is desirable that a rating be generated for each critiqued aspect. This rating represents a qualitative estimate of the quality of the solution".* However, I believe that care must be taken when using computational considerations as the basis for developing models of problem-solving processes.

Research has also indicated that some record of the design process is necessary to facilitate the process of critiquing. Bañares-Alcántara [1995] suggests *"It soon became apparent that expecting a computer program to criticise and propose improvements for a given chemical plant would be tantamount to asking a person to explain the plot of a film by analysing a single frame from it. For such a knowledge-based system to operate in a directed and useful way, it would be necessary to give access to the history of the design process".* This suggests that the overall context of a design is important when considering a critique to make.

Hence, there are clearly a number of different ways in which the critiquing process might be decomposed in a task–oriented manner. In a concurrent design environment, I believe that critics may behave in an incremental or passive mode, depending on the organisational structure within which the design process occurs, although in a purely concurrent environment, critics would be able to intervene whenever they perceive sub-optimal design decisions from their own perspective.

2.12.4 Other issues in modelling and implementing critiquing

Clearly different personnel involved in the concurrent design process have different vocabularies to describe a product and also have different ways of visualising the product. This can be expressed as different 'views' of a product.

Mastaglio [1989] *"they must be able to explain recommendations in terms the user can understand"*. In addition, the justification for a critique is also very important (in that the critic must explain to the user why a critique has been generated). The importance of the explanation or rationale behind a critique is also outlined in Fischer and Mastaglio [1991]. A critic should also be able to match its critique to different users. Silverman [1992] *"critics with no user adaptivity run significant risk of saying the wrong thing to their users"*.

In describing their ICM (Interdisciplinary Communication Medium) system, Fruchter et al [1993] suggest that *"The architect and structural engineer have different views of the model of the design"*. In the 'Design Fusion' system Finger et al [1992] also allow local representations of the shared representation to be created for reasoning and analysis. Ramscar et al [1996] look at the problems of dialogue limitations when moving from a closed domain to a more general design domain. Oxman [1995] suggests that *"each knowledge structure may also be associated with it's own representational medium"*. This requirement for different views of a design has influenced the development of systems that can interpret different views of a design – see Balachandran and Gero [1988], Dwivedi et al [1993].

2.12.5 Some conclusions regarding critiquing

According to Krishnamoorthy et al [1991*] "a study of the nature of the critiquing process shows that the knowledge representation formalism and the methodology of evaluation are independent of the domain that a critic addresses ... "* Shiva Kumar et al [1994] go on state *"it is worthwhile to capture the generic aspects of a critic into a generic tool and then reuse it... "*. This implies that lessons learned from critiquing in other fields will have direct relevance to a model of critiquing in the field of concurrent design.

A number of critiquing systems in the field of design have been analysed and discussed. One important issue that is interesting to is the degree to which the referenced literature focuses on the computational implementation of such systems. The systems outlined typically imply a model of how critiquing occurs but do not discuss how or where the underlying model of critiquing is derived. Hence while the researched literature is a useful starting point as to how critiquing occurs, it does not offer a fundamental, experimentally-derived model of critiquing in a concurrent design context.

A number of the critiquing systems and their underlying model of critiquing imply that a propose-critique type approach is applied to the process. Where only one critic is evaluating a plan or design, I believe this task model is valid. However, because of the inherently multi disciplined nature of concurrent design, conflict will inevitably arise between different 'agents' involved in the concurrent design process. Harrington et al [1995] - *"conflict between agents is therefore inherent and the resolution of this conflict is a major problem in distributed environments".*

Because of this, there must be some means of resolving conflict between the different participants in the concurrent design process. This process whereby these differences are reconciled in some manner I will term negotiation. However, references in the literature to 'argumentation' (see Clark [1990]), and 'contested collaboration' (see Sonnenwald [1996]) appear to be processes with similar attributes.

2.13 Models for negotiation

Negotiation has been analysed as a key process in a wide-ranging field of disciplines or domains from social psychology (for example Druckman [1973]) through distributed AI (Bond and Gasser [1988]). Because of the large amount of diverse work that has been done in this field, I will focus the discussion of negotiation on work that has been done in the design-related engineering fields.

When different 'experts' cooperate during the concurrent design of a product, there will inevitably be some 'discussion' where differences in viewpoint between the experts are outlined and reconciled. Lander [1997] sees one of the key issues as conflict resolution between different agents engaged in concurrent design and sees negotiating strategies as vital in resolving such conflict.

Negotiation provides a means by which conflicts, derived from multi-perspective critiquing of an evolving design, can be resolved. Bucciarelli [1988] summarises this as *'decisions made across disciplines are best seen as negotiations amongst parties who, whilst sharing a common goal at some level, hold different interests'.*

Klein and Lu [1989] look at negotiation strategies suitable for computational implementation and suggest that conflict resolution knowledge can be viewed as a form of problem-solving expertise. This implies (as has been suggested for critiquing) that negotiation strategies may be generic in nature. This work is expanded on by Klein [1991] and Klein [1992].

Klein and Lu [1989] go on to look at deadlocks that can occur when different experts involved in the concurrent process offer conflicting critiques of a design. They suggest negotiation can be either competitive or co-operative. The conflicting form is likely to be dictated by personality and issues relating to personal, not overall goals. Hence they ignore these 'psychological' human factors and concentrate solely on more cooperative modes of negotiation. Protocol analysis of designers involved in architectural design is used to outline different methods of co-operative conflict resolution strategies relevant to machine based agents.

The work of Klein and Lu [1989] outlines a number of different strategies utilised in architectural design to resolve conflict in a co-operative mode of design. They consider a rationale for a design decision as being a crucial element in any resulting negotiation process. However, while concurrent design can be co-operative in nature, this is not necessarily the case. The views and goals of different perspectives involved in the concurrent design process can be very polarised in nature resulting in a more conflicting mode of negotiation.

Werkman [1991] sees negotiation as an aid to resolving conflicts in a distributed agent-based system called the 'Designer fabricator interpreter'. The model of negotiation employed allows for both cooperative and conflicting negotiating stances to be taken by the computational agents comprising the system. Negotiation is managed within the system using a novel representation formalism termed 'shareable perspectives'. When deadlock situations occur, a central arbitrator software agent makes a decision on the solution to take based on how important each agent flags it's view of the particular issue to be. In the field of human conflict resolution in a conflicting negotiating scenario, this is believed to be a possible way in which conflicts are settled. However, whether it is the relative importance that each agent attaches to the issue that is the defining factor is open to debate.

Because of the inherently conflicting nature of concurrent design as opposed to more co-operative or collaborative modes of design, conflict resolution is a very complex area. It may be that for a particular area of conflict, there will be no means of satisfying all parties and some more arbitrary means of conflict resolution may be employed to counter deadlock situations. Hence while some researchers (E.g. Baker [1993]) see negotiation as a means of achieving some accord, in a conflicting concurrent design scenario this may not necessarily be the case.

Baker [1993] defines specific styles of negotiation based on the goals and attitudes of agents before, during and after the negotiation, the types of thing that are negotiated and strategies and communicative acts for achieving goals. In common with other research on negotiation, Baker makes a distinction between conflicting and more cooperative modes of interaction and goes on to conclude that *"more detailed modelling of the (negotiation) process is required"*.

Harrington and Soltain [1995] suggest concession making as an effective negotiating strategy where differences between conflicting parties are small but not so effective where the different parties are inflexible over the issues. They also suggest that this implies a co-operative model of conflict, not a conflicting one.

When implemented computationally, conflict can generally be classified as design time (i.e. case-specific techniques can be unearthed at development time through knowledge acquisition techniques with ‘experts’) or run time (i.e. more generic resolution strategies are used as the teamwork process progresses). Development time conflict resolution is seen as being a computational means of implementing conflict resolution but I believe that it is unlikely that human experts have pre-determined conflict resolution strategies for use in every possible different conflict scenario. Harrington et al [1995] point out how the possibly thousands of interdependencies between design decisions mitigate against development time strategies. Lander [1997] also outlines the advantages of dynamic conflict resolution strategies "... *we cannot engineer an agent at design or implementation time to be in agreement with all other potential future agents (which may not even be imagined yet). Therefore dynamic conflict management is an inherent requirement of MADS (Multi Agent Design Systems)*". Hence, the conclusion is that the more flexible and generic run time strategies are more indicative of models of how human negotiation processes occur.

Bahler et al [1994] recognise that in the concurrent design process, conflicts among sharply diverging viewpoints may occur. They describe a negotiation protocol, based on economic utility. They look at the application of this protocol in using constraints to detect conflict and support negotiation in a design advice system.

Clark [1990] outlines argumentation as a process whereby experts can outline inconsistencies in a cooperative problem-solving task. They go on to argue that in such situations *"it is not easy or even possible to identify the 'right' answer ..."*.

The TEAM system (see Lander and Lesser [1993]) is a framework for integrating agents into a multi-agent system. In particular, Lander and Lesser focus on conflict that can occur between the agents and negotiating strategies for resolving this conflict. They suggest that techniques used in this area include bargaining, restructuring, constraint relaxation, mediation and arbitration. Interestingly, Lander and Lesser argue that sharing information about constraints and priorities at an early stage in the design process is a powerful conflict management technique.

Wong [1994] describes a qualitative problem-solving system based upon a formal model of social choice theory and computational methods to manage the expression of preferences by different agents and to support negotiation between them. This system provides more than support for communication between users but nonetheless negotiation is driven by humans and is not automatic. Sycara's PERSUADER (Sycara [1989]) provides a sophisticated approach to support for negotiation. It uses a mediated negotiation model and multi-attribute utility theory.

Zlotkin and Rosenschein [1996] have also described different negotiation domains. In order of increasing complexity and subtlety these are task-oriented, state-oriented and worth oriented domains. Analysis of the nature of the domain within which negotiating agents are to be implemented informs the choice of computational negotiation mechanism. In the field of distributed AI, negotiation is also a rich topic of research, see Bond and Gasser [1988].

Hence, negotiation can be seen as a widely used process in fields other than concurrent design. It is interesting to note that while computational considerations are a focus of a number of the systems and research discussed, the analysed literature does make reference to underlying models of negotiation and the experimental basis for these models. In addition, critical factors that need to be considered during the negotiation process are how cooperative the process is and what the goals of the different agents involved in the process are.

A study by Olson et al [1992] analyses a number of participants in software design meetings at an early stage of the software life-cycle (from an initial incomplete specification to the development of a conceptual design). This study of the early software design process shows some similarities with the mechanically–oriented concurrent design process. However, an interesting aspect was the relative lack of conflict and negotiation evident in the Olsen study. A particularly telling point was that the team of software engineers are developing 'designs to be built by others'. This suggests that the participants in the design meetings had less of a personal stake on the potential downstream effect of the different constraints resulting in a more co-operative mode of designing.

2.14 Chapter Summary

There are many different and sometimes conflicting definitions for concurrent engineering and the associated design process, which I term concurrent design. Concurrent engineering techniques are currently being used in a number of diverse industries. The techniques and methods associated with the complete concurrent engineering process have been extensively researched and a huge body of literature exists on the subject. I have not attempted to comprehensively reference this body but have outlined the main issues and characteristics of concurrent engineering.

Design can be characterised as consisting of an analysis phase, where a design's requirements are defined and a design synthesis stage, where design solutions are generated. This research focuses on the design synthesis stage.

I define the main features and characteristics of the concurrent design approach as involving the use of multi-functional expertise or knowledge to pinpoint possible downstream life-cycle constraints on the design of a product. This can involve the use of teamwork, although concurrent design does not implicitly assume a team–based approach to the design process. However there are a number of practical considerations that limit the successful use of such teams. These include communication, availability and the geographical distribution of team members. I believe concurrent design is subtly different from other, more co-operative, modes of group design. What is surprising is the relative absence of literature regarding the concurrent design task implied by a concurrent engineering approach and the development of models to represent this task.

Models for design can be broadly classified as based on either the rational or the reflective pracrtitioner approach. A number of the rational methods have been formalised as prescriptive guidelines for the design process. These so-called prescriptive methods outline a prescribed process which a designer should use to tackle a given design problem. However, the models do not explicitly support the process of concurrent design as I have defined it, in that knowledge appertaining to different life-cycle perspectives can act as a very important informing constraint on the design process

The work of Chandrasekaran [1990] and others suggests design proceeds via a series of steps or sub-tasks of propose-critique-modify. I present a number of models which attempt to describe these propose and critique tasks. However, I believe that the process whereby differing critiques of a design are accommodated in the concurrent design process is more complicated than merely modifying the original design solution. The concept of negotiation is introduced as a means whereby conflict resolution between different participants in the concurrent design process is achieved.

Smithers [1996] argues that what is required are knowledge-level theories of the design process. These should be developed with reference to lessons learned in the field of knowledge engineering. As will be shown in the next chapter, models of the design process have been formalised as expertise task structures and problem-solving methods, specifically in the KADS and latterly the *Common*KADS methodologies. These are effectively structures showing the stages of design and types of knowledge that designers use during the design task.

3 Formalised models for design

3.1 Introduction

The previous chapter has outlined and discussed a number of different models for both the complete design process and also models for sub-processes (or tasks) comprising design. These models were generally presented in a descriptive, natural language format.

A number of researchers and methodologists have attempted to formalise these and other models for the design process using appropriate methodologies. In particular, the *Common*KADS methodology contains a number of formalised models for both the overall design process and some of the sub-tasks comprising the design process.

I begin by outlining the fundamental points of different software development methodologies, with an emphasis on *Common*KADS. This chapter then gives a critical review of the *Common*KADS models for design with an emphasis on the degree of support they provide for concurrent design. The analysis of both the *Common*KADS and other models discussed in this chapter is discussed in an implementation-independent manner. However, an implicit assumption of a number of the models I discuss is that they will ultimately form the basis of some computer system. Where relevant, any derived computational models and implementations are also discussed.

3.2 Design model representation in formal and semi-formal systems

In order to analyse the contribution made to the development of knowledge-level models for design as a result of the *Common*KADS methodology, it is necessary to outline the fundamentals of both *Common*KADS and also other knowledge-based systems' development methodologies.

A number of different methodologies exist for the development of software systems. These include SSADM and Yourdon (see Sommerville [1992]). Because of their requirements, the development of knowledge-based software systems places particularly stringent demands on the requirements of a development methodology. Most importantly, these include issues such as how to elicit and model knowledge for use in such systems.

Early work by Newell and Simon [1982], which developed ideas around the 'knowledge-level' have acted as the driving force for different knowledge-based system development methodologies and the way knowledge is acquired and modelled in such systems. Van de Welde [1994] outlines how research since the early knowledge-based systems has changed the way in which such systems are now viewed:

"the idea that knowledge is there to be extracted from the human expert and translated into usable knowledge elements is misleading. Instead knowledge is now viewed as one way of modelling rational (or rationalisable) behaviour as it is being seen by a particular observer, for a particular expert, on a particular problem in a particular situation" –

A number of structured methodologies for developing KBS have been developed, or are under development and different tools and research projects attempt to support the various stages of the development of KBS. This can include any of the stages from the initial knowledge acquisition phase with experts through to computational implementation of actual systems. A number of the different tools show a number of similarities although clearly some projects are aimed at different stages of KBS development.

VITAL (see Dominique et al [1993]) is both a methodology and a set of software tools, which support the structured development of knowledge-based systems. This methodology supports the top-down refinement of models of expertise at various levels of abstraction.

The ACACIA project (see Dieng et al [1994]) aims to help knowledge engineers and experts during the knowledge acquisition phase by developing a knowledge acquisition tool and a methodology. In particular, the project emphasises the need to allow for knowledge acquisition from multiple experts.

However, in Europe at least, and increasingly in the USA, the *Common*KADS methodology has come to be regarded as the principal methodology for developing knowledge-based systems. The methodology has evolved from extensive work done by a number of researchers, led by the University of Amsterdam, on the original development of the earlier KADS methodology (see Tansley and Hayball [1993]). It has been greatly influenced by work done on knowledge modelling such as Steel's 'Components of Expertise' (see Steels [1990]). For a more detailed description of the methodology and its evolution, see Breuker and Wielinga [1985], Schreiber et al [1993] and Breuker and Van de Welde [1994]. The key features of the methodology are now described.

3.3 KADS and *Common*KADS

The main focus of the original KADS methodology was as a means of knowledge acquisition for subsequent incorporation in a KBS and the modelling techniques advocated by the methodology were geared towards this goal. However *Common*KADS is intended to be a more encompassing methodology to support the entire KBS development life-cycle from inception to implementation in an enterprise or organisation. KADS (and latterly *Common*KADS) take a very task–oriented view of human problem-solving behaviour.

Knowledge-level modelling in *Common*KADS is driven by a 'competence' based approach. A knowledge-level model for some task requiring human expertise is assumed to be sufficient if the model can be used for problem-solving in the domain. Hence this approach does not require a complete cognitive understanding of human problem-solving expertise in order to implement computational support for some task. As a result, the knowledge-level modelling incorporated in *Common*KADS does not attempt to accurately reflect or predict the cognitive processes utilised by humans. Rather, it is intended to allow the development of implementation-independent models of problem-solving for subsequent incorporation into computer systems.

Originally, KADS suggested that many tasks, at a certain level of abstraction from any one domain, such as classification, diagnosis and indeed design are generic in nature (i.e. they follow the same pattern irrespective of the area under consideration). As discussed in the previous chapter, this is supported by the fact that many types of design, such as mechanical, software and architectural design show a number of similarities. In order to fully model both the knowledge required by a knowledge-based system and also the organisation within which such a system would be implemented, *Common*KADS advocates the building of a number of different models which together comprise the *CommonKADS* model set.

3.4 The *Common*KADS model set

*Common*KADS assumes a number of different models are combined when developing a complete knowledge-based system. These are the:

- Expertise model
- Task model
- Organisation model
- Communication model
- Agent model
- Design model

These different models and the dependencies between the models comprising the *Common*KADS model set are expanded on in Chapter 11 and discussed in more detail in de Hoog et al [1993(a)]. The models give different and complementary views on how processes occur within the context of a particular organisation. *Common*KADS provides templates for each of these models. It is assumed that during a *Common*KADS project, these templates are instantiated to the required degree. The models relating to an expert's problem-solving expertise are contained in the expertise model. The expertise model is where human problem-solving expertise and knowledge are modelled in *Common*KADS.

3.5 The *Common*KADS expertise model

The top-level components in a *Common*KADS expertise model are application and strategic knowledge. The application knowledge consists of three distinct epistemological categories or layers. These categories are summarised as follows:

- Task knowledge consists of a task definition defining what the goals of the task are and the task body, where the activities comprising the task are described. This is modelled as a task structure. Task decompositions show the different sub-tasks that are used to accomplish the given task. Where sub-tasks cannot be further decomposed, they are assumed to be basic inferences.

- Inference knowledge specifies basic inferences that can be made in the domain knowledge and can be linked in inference structures. *"An inference specified at the inference level is assumed to be primitive in the sense that it is defined through it's name, an input / output specification and a reference to the domain knowledge that it uses"* - Aben [1994]. In this sense, inferences can be seen as a 'black-box' in that inputs and outputs are defined but the way a human expert derives outputs from the inputs is not modelled. Complex computational techniques may be required to produce the required output.

- Domain knowledge summaries the way in which an application 'sees the world' in terms of a domain ontology and a domain model which uses the domain ontology to capture groups of statements about the domain. This can be seen as a more 'static' structure than the task and inference layers.

Knowledge roles control the links between the different categories and allow general concepts defined in the different layers to be specified for a particular application.

It should be noted that the task models discussed, which represent problem-solving strategies within the expertise model, are distinct from the *Common*KADS task model itself, which is a higher level description of the tasks a knowledge-based system is to support. To prevent any misunderstanding, the task models within the expertise model will hereinafter be referred to as expertise task models.

At a certain level of abstraction, an expertise task model can be seen as analogous to a cognitive model of an expert's problem-solving behaviour. However, an expertise task model is ultimately decomposed to inferences and the computational means to achieve the inferencial capability will not necessarily model any cognitive behaviour. As a result – *"a task model is an engineering artefact, designed by the knowledge engineer which does not* *necessarily correspond to a cognitive model of the domain expert's problem-solving behaviour"* – Rademakers [1991]. However, an expertise task model does show the task decomposition that a domain expert could use to solve a particular task.

The original KADS projects identified a hierarchy of expertise task models, see Tansley and Hayball [1993] and Appendix A. This classification of tasks was influenced by a number of earlier works including that by Clancey [1985] on heuristic classification. Tasks are split up into three broad areas: analysis, modification and synthetic type tasks. Design is considered to be a synthetic task. The original KADS hierarchy also gives an indication of the amount of work done in different areas to analyse problem-solving behaviour. More work has been done in the analysis area than in the synthesis and modification areas. The expertise task models presented for design are not definitive and Tansley and Hayball [1992] suggest that more work needs to be done to verify and extend these models.

Different researchers have developed expertise task models for a number of different tasks. The task models developed for design are discussed later in this chapter. It must be stressed that these expertise task models are not merely abstract academic ideas. They have been developed from detailed study of how people accomplish certain expert tasks. As a result they act as a very important foundation stone for developing any knowledge-based system. The link between task definitions in the task layer of application knowledge and problem-solving methods are now discussed.

3.6 Tasks and Problems solving methods in *Common*KADS

The terms 'method' and 'task' in the literature are not always consistent (especially with respect to the 'granularity' at which the method is applied) and the problem-solving methods and tasks outlined above do not necessarily correspond with problem-solving methods in the *Common*KADS sense. This is echoed by Chandrasekaran et al [1992] - *"the word task has been used in somewhat differing senses in the field, contributing to much confusion".*

CommonKADS explicitly makes a distinction between problem-solving methods and tasks. A task definition is effectively a statement, defining what inputs are required and what outputs are generated from application of the task. However, the task definition is implementation-independent in that the task does not dictate how the task might be achieved. It is the problem-solving method that is applied to the task that effectively dictates 'how' the specified task is decomposed into more discrete steps or subtasks. Duursma et al [1994] - *"a set of coherent activities that are performed to achieve a goal in a given domain".* When referencing tasks and problem-solving methods hereinafter, these definitions are assumed.

Valente et al [1994] characterise problem-solving methods in *Common*KADS as specifying:

- How a certain task can be decomposed into sub-tasks at a lower level of detail
- How the execution of these sub-tasks is controlled
- Which requirements are imposed on the representation of the domain knowledge in order for the method to work

This is summarised by Wielinga et al [1994a] as *"A problem-solving method is applied to a task definition and after a mapping of (generic) terms used in the method description onto* *the task specific terms, the body of the task can be instantiated from the method description".*

At this point it is also necessary to note that the *Common*KADS library includes a number of different PSM's (problem solving methods) which can be applied to different tasks. As outlined in Breuker and Boer [1998], it is a current issue of debate as to whether a problem-solving method can be considered independent of the task to which it is applied – i.e. are problem-solving methods generic?

The *Common*KADS library of problem types and problem-solving methods is not intended to be a static library. It is intended as a dynamic library with models continually being updated and refined. For example, in the field of planning, Benjamins et al [1996] are working to expand and refine the problem-solving methods applicable to the planning task. The intention of the developers of the *Common*KADS suite of problem types and expertise task models is that they act as suitable templates that can then be refined by knowledge engineers when implementing KBS.

In an earlier section I outlined how the KADS methodology outlined a hierarchy of different tasks, which had particular characteristics in common. However, as we can now see, the application of a particular problem-solving method can also be used to characterise problem-solving expertise. Hence current *Common*KADS thinking suggests that it is not the task that is generic but that there are in fact a number of different problem types with their own distinctive and generic nature. In addition, any given task may require the solution of a number of different problem types.

3.7 The *Common*KADS suite of problem types

In their groundbreaking work in AI, Newell and Simon [1972] characterise problems as *"some conflict between a current state and a goal state"*. This 'goal state' was assumed to exist in the problem space of possible solutions. Breuker [1994] goes on to suggest that a given problem type (e.g. design, assessment etc) is defined by the type of solution implied by the problem type – *"problem types are defined by their generic solution or conclusions"*.

Breuker suggests the design problem type is characterised by a *"structure of elements".* This corresponds with Chandrasekaran [1990]'s view of the design solution as an assembly of sub-components and some of the other definitions for design discussed in Chapter 1. Breuker goes on to outline how this emphasis on different problem types has allowed the suite of problem types to be classified – *"by distinguishing the problem types from the tasks, common and rational dependencies between problem types become visible".*

I will now outline the important points that *Common*KADS assume differentiate problem types from tasks, this is expanded on in greater detail in Breuker [1994].

Breuker suggests that there are three steps involved in going from a given problem type to a task. These are identifying the problem where a discrepancy between a current state and a norm state are identified. This is followed by defining the problem, whereby potential solutions aim at a goal. The third step is to turn the well-defined problem definition into a task by constructing a problem-solving method.

Different problem types do not exist in isolation, rather there are important dependencies between the different problem types. As an example (Breuker [1994]), I will use the design problem type to illustrate these dependencies.

On first inspection, the design problem can imply the generation of a solution in the form a structure of elements. However, the driving force for this design problem is in the form of a set of requirements. These design requirements are not simply given, they are taken to be the result of a modelling process (where modelling is another problem type). Therefore the design task contains both modelling and design problem types. This dependency is illustrated in figure 3.1

modelling → *design*

Figure 3.1: Dependencies between modelling and design (adapted from Breuker [1994]).

Current *Common*KADS thinking suggests that different problem types are dependent on one another. The generic solutions from one problem type act as input roles to another. For example, the design problem type is dependent on a prior modelling task being performed to determine the requirements of a design, typically in the form of a design specification.

However, Breuker goes on to argue that the assignment of the structure of elements generated as part of the design problem to a physical implementation is a further necessary step. If this assignment problem results in any unforeseen problems, this may result in a drastic reconfiguration of a system and the possibility of redesign. Hence this further dependency between problem types is now shown in figure 3.2.

modelling → *design* → *assignment* → *prediction* → *monitor*
→ *diagnosis* → *design*

Figure 3.2: Dependencies for the complete design cycle. (adapted from Breuker [1994]).

As well as the design problem type being dependent on a prior modelling task, there are dependencies exhibited between problem types that occur after the design task. For instance, the generic output of the design problem type is a structure of components. The assignment problem types takes this structure as input and distributes additional elements over the structure, typically in a configurational manner.

The dependencies shown on Figure 3.2 also imply some iteration may be necessary between the different problem types during accomplishment of the design task. Detailed analysis of other problem-types show similar dependencies. Hence *Common*KADS suggests that the suite of different problem types and their dependencies is now as given in figure 3.3

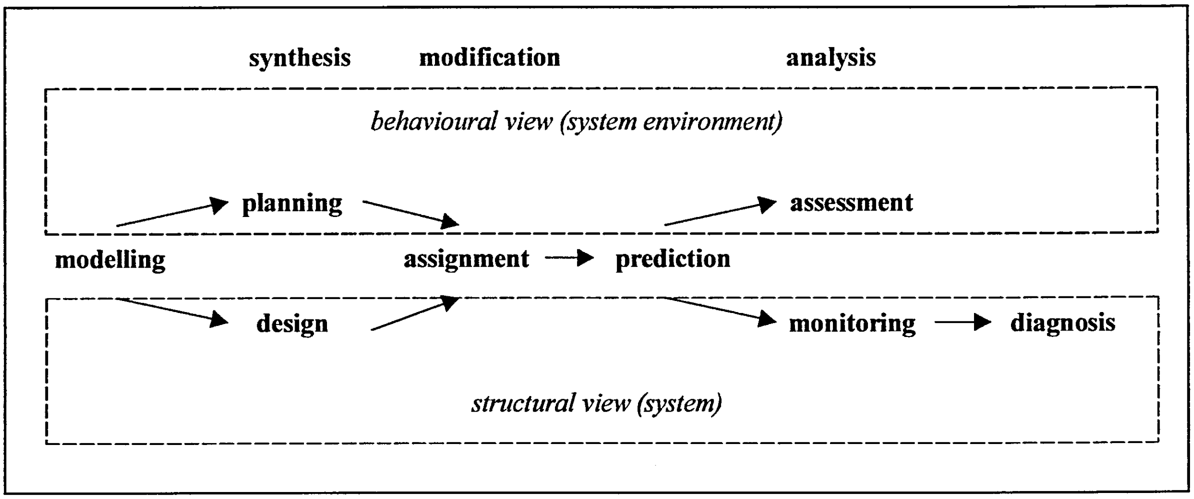


Figure 3.3: The *Common*KADS suite of problem types (adapted from Breuker [1994]).

The complete *Common*KADS suite of problem types exhibits the dependencies between the main *Common*KADS problem types. These types are characterised as being either synthetic, modificational or analytical in nature. In addition, Breuker differentiates between problem types concerned with structure (design, assignment etc) and those concerned with behaviour (planning, assessment).

The steps involved in going from a problem type to a task involve the construction of a task decomposition. The task decomposition for a task, in the form of an expertise task model, is dependent on the problem-solving method(s) that are applied to the task in order to solve it. Work on both KADS and *Common*KADS has resulted in the development of a number of different expertise task models for the design process.

3.8 The KADS expertise task models for design

The original KADS methodology resulted in the development of a number of models for the design task. The complete set of expertise task models for design, developed as part of KADS and *Common*KADS, are detailed in Appendix A.

In order to help illustrate the work done and what the structure of the different models imply, some of the models relating to design will be discussed and analysed. Figure 3.4 shows the KADS generic task structure for the design task. This model shows how knowledge roles (in the rectangular boxes) act as input to tasks or inferences (shown in ellipses). The output from each sub-task are further knowledge roles, which can then act as inputs to further sub-tasks.

Each sub-task may be expanded in a similar manner to give a task decomposition diagram. Different problem-solving methods may be used to accomplish the different sub-tasks, each giving it's own task decomposition. Where it is assumed a task or sub-task cannot be expanded any further, the sub-task is assumed to be a primitive inference.

The concept of a 'knowledge role' is used so that any knowledge an expert may use in problem-solving may be structured in different ways depending on the task or sub-task it is applied to. For instance, a sub-assembly of components may be linked to the knowledge role 'Solution' output from some design task or it may appear as the role 'Specification' input to some assignment task. For further details as to the structure of KADS and *Common*KADS expertise task models and the concepts of sub-task, inference and roles, see Breuker and Van de Welde [1994].

At the top level of abstraction, the original KADS model for design represents a serial approach to the design process. This task structure for design is broadly in agreement with the prescriptive models for the design process presented earlier in Pugh [1990] and Pahl and Beitz [1984]. However Pahl and Beitz suggest that an embodied design would be an intermediate stage between a conceptual design and a fully detailed design.

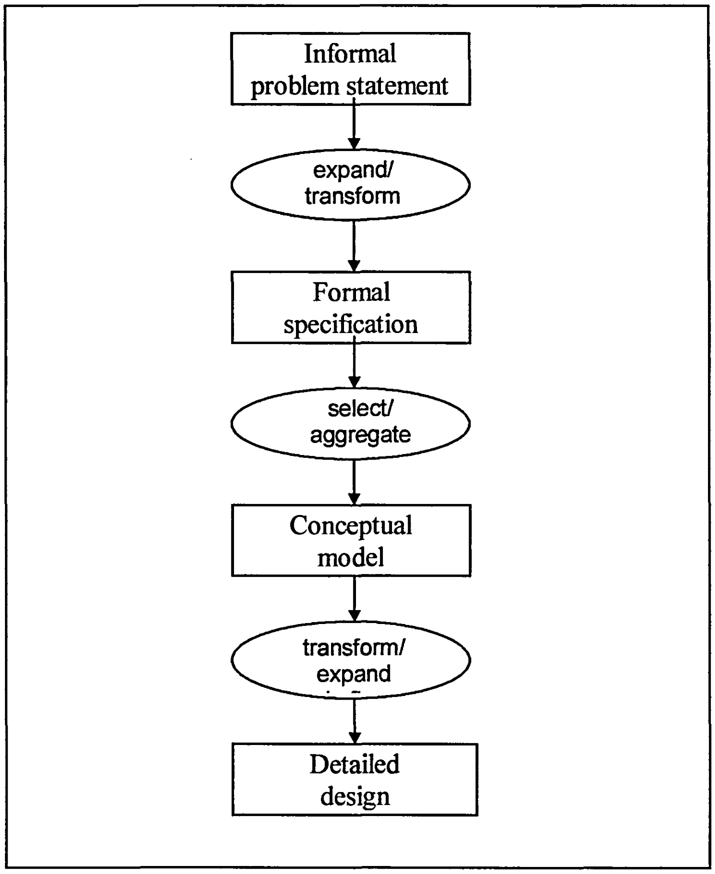


Figure 3.4: The original KADS task structure for the generic design task (adapted from Tansley and Hayball [1993]).

This expertise task model shows how the design task was originally modelled by KADS as being composed of three separate sub-tasks: expand/transform, select/aggregate and transform/expand. Via these sub-tasks, a detailed design is derived from an informal problem statement.

KADS also included two refinements of the design task, hierarchical and incremental design. Incremental design implies that the developed conceptual model is decomposed into a number of functions that the model must accomplish. Each function is then matched to a suitable component, which will fulfill the stated function, and the components are then combined to arrive at a detailed design, using an existing model as the basis for combining the different components. As discussed earlier in this chapter, the method of functional decomposition is extensively used in the design process (see Pahl and Beitz [1984], Schmidt and Schmidt [1996] etc.). Chandraserkaran [1994] gives a historical review of the field of functional representation and it's use in fields such as design. However, while the KADS model uses this method to generate a detailed design, other texts analysed use this 'method' to derive conceptual designs from a specification, not for this latter part of the design process. For an example see Sharpe and Bracewell [1993].

Hierarchical design involves deriving a skeletal model from a formal specification and a set of models. This model is then decomposed to component level. Each component is then 'designed' in the same way in a recursive manner until the component is specified at a sufficient level of detail. The components are aggregated to form the detailed design and there is a 'compare' inference to ensure the detailed design complies with the original formal specification from a functional perspective.

Both these lower level models of design imply that the product being designed is somehow decomposed into it's constituent functions / components. This is analogous to the 'chunking' approach discussed by Ulrich and Eppinger [1995]. They show how individual functions can be combined into one component or 'chunked' across different components and discuss the advantages and disadvantages of the two different approaches.

I would consider the overall KADS model for design model to be very coarse-grained in that it gives a high-level overview of how the design process occurs. However, it does not give much indication, at a more fine-grained level, of how the design process occurs. The models for hierarchical and incremental design give more information about how the design process may be further decomposed into subtasks. However, neither of these approaches makes any reference to concurrent consideration of life-cycle perspectives.

3.9 The *Common*KADS expertise task models for design

The *Common*KADS models for design (see Bernaras and Van de Welde [1994]) expand on the models developed as part of KADS and attempt to synthesise the work of different researchers, including Steels [1990] and Kruger and Wielinga [1993].

Kruger and Wielinga used protocol analysis to analyse a number of single designers working on the same given design task. Based on their study, they suggest that designers

use the decomposition–solution-recomposition method for the design problem. They suggest that no evidence can be found for designers working in the propose–critique–modify mode. However, they do suggest that during the identification of requirements or constraints, the problem is approached from different viewpoints including ergonomics and construction.
Because of the nature of the study (a single designer developing a design) it is perhaps not surprising that there is a lack of evidence of concurrency. I believe that the case studies analysed by Kruger and Wielinga do not adequately reflect the reality of how concurrent design occurs in an industrial setting.

The *Common*KADS models for design presented by Bernaras and van de Welde [1994] view design as two distinct phases of analysis and synthesis, representing the requirements definition and solution generation stages. These two phases are examples of the *Common*KADS problem types modelling and design respectively. In terms of the different problem types discussed in an earlier section, the analysis stage is effectively the modelling problem type while the synthesis stage is the problem type I would generally perceive as a design problem type.

Analysis is seen as translating the 'needs and desires' of a customer into a requirements description. These requirements are then further formalised, in the form of constraints, as problem statements. Hence Bernaras and Van de Welde propose a generic task structure for design as shown in Figure 3.5.

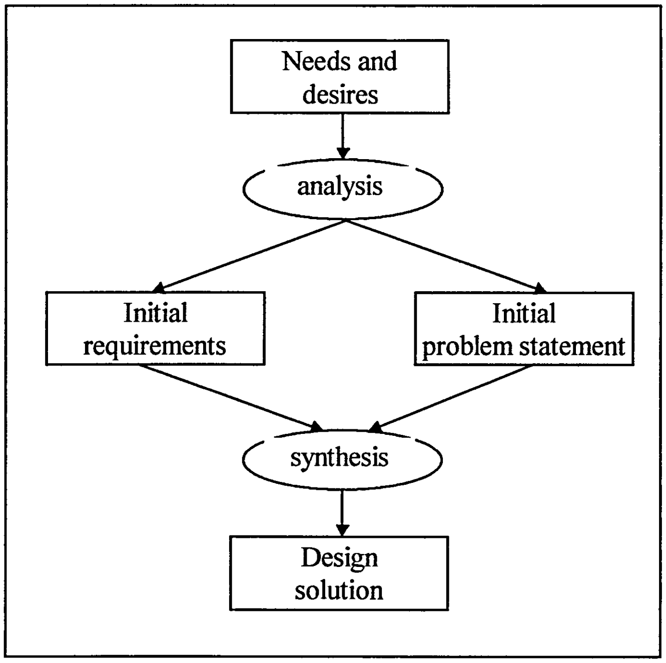


Figure 3.5: The *Common*KADS task model for the analysis and synthesis stages of the overall design process.

Analysis and synthesis are examples of the *Common*KADS problem types modelling and design respectively. Hence, it the synthesis task which would generally be considered as 'design' although the analysis task is clearly an essential precursor to the synthesis task. Requirements are the criteria for evaluating a design solution while problem statements (constraints) are these requirements expressed in a formal and concise manner.

Effectively the analysis process is one of interpreting abstract customer ‘needs and desires’ into a more concrete, formally expressed representation. The analysis models presented by Bernaras and van de Welde depend on how abstract the initial customer request is. This could be anything from a very naive customer request expressed in vague and abstract terms to a fully specified requirements document. They view the analysis phase as understanding the knowledge a customer has, hence analysis can effectively be seen as performing a knowledge elicitation exercise with the customer. This analysis phase extends into the area of requirements engineering, a rapidly expanding field. Hoffman [1993] gives a general review of work in this area while Siddiqi and Shekaran [1996] outline some trends in this field.

Barneras and Van de Welde [1994] view design synthesis as a more complex task and develop design models viewed from three different viewpoints. These are the construction dimension (relating to the object or artefact), the requirements dimension (relating to the intended user) and viewed from the type of static design knowledge available. From the requirements dimension, they present models for routine, innovative and original design, based on how well defined the given specification for the design is. These correspond to the types of design outlined earlier by Pahl and Beitz [1984] and others. From the types of static design knowledge available they outline models for case-based design, transformational design, decomposition based design and generic model design.

The construction dimension is concerned with the elements and their attributes involved in the design problem. They view allocation, configuration design and parametric design as being types of design in this class and present several models for parametric design. These models outline in more detail the task structure that results from a designer producing a detailed design from a formal specification.

From the requirements dimension, original design has the task structure shown in Figure 3.6. This task model clearly involves the exploration of a product's requirements. However, the sub-task 'construct design' does not give any detail as to how this task might be accomplished.

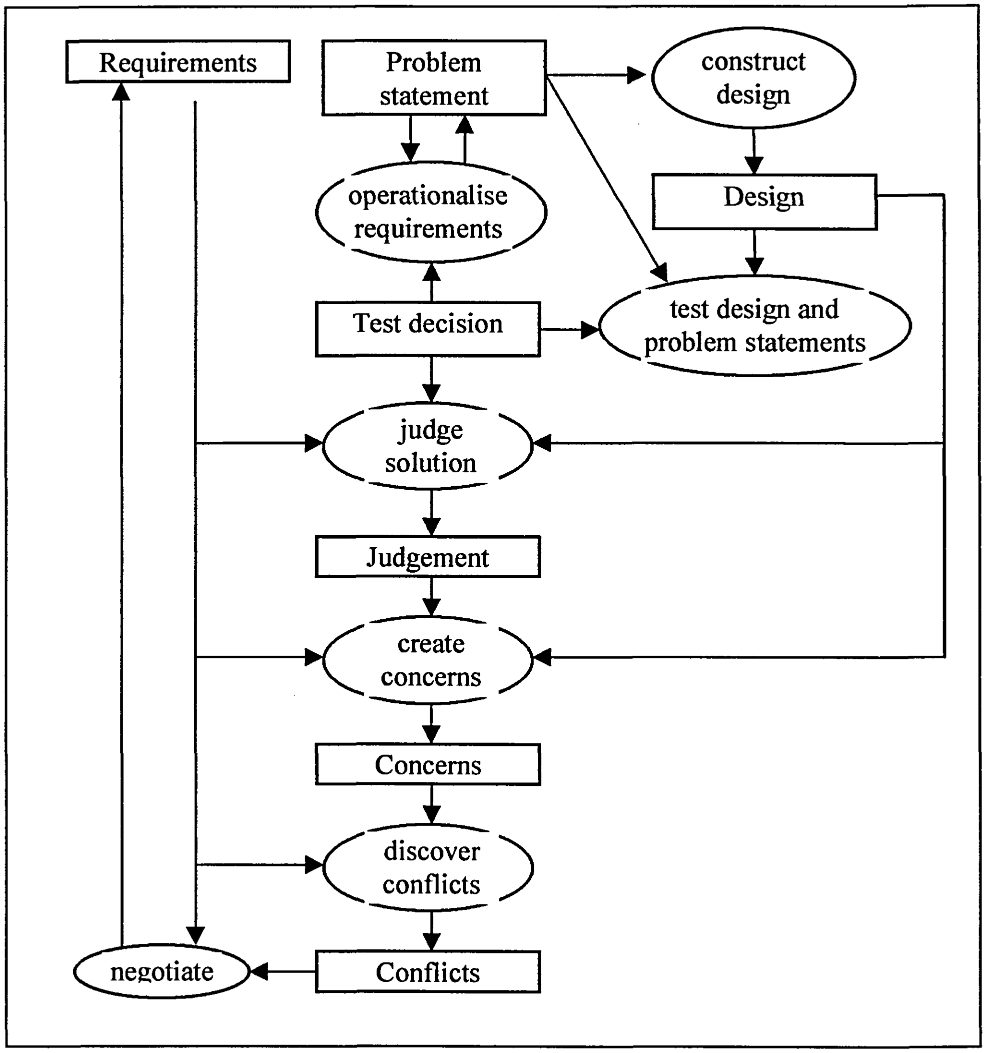


Figure 3.6: The *Common*KADS model for original design.

In this type of design the requirements evolve in parallel with the design process. However, the model does not give any indication as to how the sub-tasks of 'discover conflicts' and 'negotiate' occur.

Kingston [1994] suggests that a further specialisation of the generic design task is exploratory design (see Appendix A). This is essentially a rapid prototyping form of design whereby a design is generated and presented to the user. The user then identifies further constraints, which were not specified, or immediately apparent in the original problem definition and this modified problem definition then acts as the driving force for a new round of design proposal. This is analogous to the *Common*KADS model for original design (i.e. the requirements for the design evolve in parallel with the design synthesis process).

The model for exploratory design proposed by Kingston [1994] does address the issue of design constraints being evaluated. However the implications of Kingston's research are that this is again done after a complete design has been generated. Kingston also considers constraints as being the major components of what is effectively a design specification and does not fully expand on what I believe is an important distinction between constraints and functions in a design specification.

From the construction dimension, Bernaras and Van de Welde [1994] present a *Common*KADS model for case-based reasoning having the expertise task structure outlined in Figure 3.7.

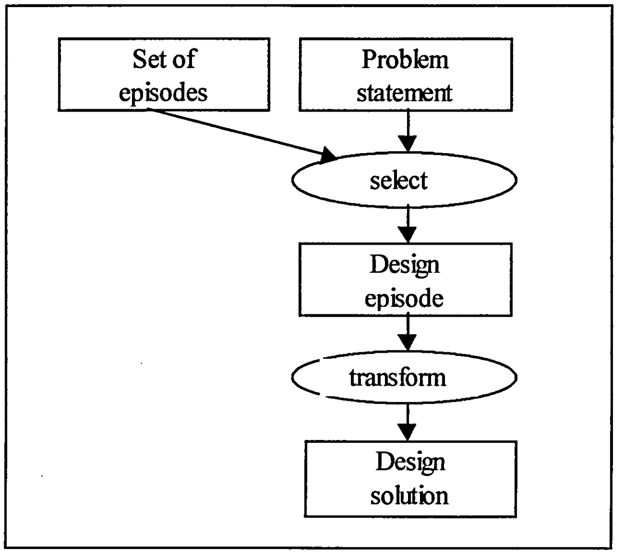


Figure 3.7: The design task being solved by application of a case-based problem-solving method.

Case-based reasoning (see Kolodner [1993]) involves adapting a past case to fit a new set of requirements. Aamodt and Plaza [1994] give an introduction to work in this field. One of the critical issues is in how to select a suitable case to match the requirements of a given scenario.

Via this method, a suitable episode is selected from a library of previous cases. This is then modified or transformed to meet the requirements of the problem statement.

3.10 A critical analysis of *Common*KADS design models and concurrency

As a general model for the design process, the splitting up of the design process into analysis and synthesis reflects the attitudes of other design researchers and also the actual practice of design.

From the requirements dimension, Bernaras and Van de Welde suggest that the analysis and synthesis stages may proceed in parallel to a degree in that the design specification evolves in parallel with the design solution. Some form of design specification must be in place before any synthetic task can begin, however the extent to which the design is specified before the synthesis task first begins dictates how 'original' the design process is. I believes this reflects actual design practice in that it is very unusual for a specification for a design to be fully developed before any work is done on the synthesis stage. This is a theme explored in more detail by Kingston [1994]. Kingston's model for 'exploratory' design implies that the design specification gradually evolves with each iteration of the design synthesis process. Additional constraints pertinent to the design are unearthed with each cycle of design synthesis and these effectively reformulate the design specification.

However, within the design synthesis stage I believe that the approach of Bernaras and Van de Welde does not adequately support the consideration of multiple, downstream life-cycle perspectives. As a result, I believe there is a potential for further work to expand and refine the *Common*KADS suite of problem types, previously discussed in Chapter 2. In particular, I believe that there are further steps between the design and assignment processes described in Breuker [1994] (where the design is assigned to actual physical elements). I believe that this is where the implications of downstream constraints will impinge on the design process, see Figure 3.8.

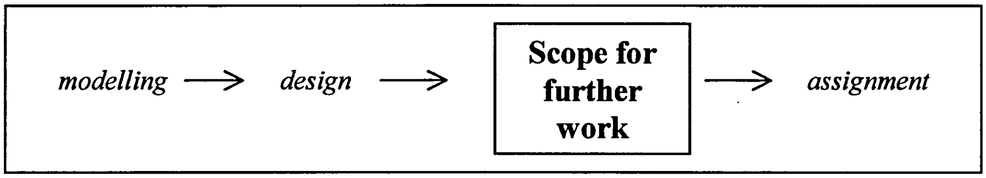


Figure 3.8: Dependencies between *Common*KADS problem types and the scope for further work (adapted from Breuker [1994]).

This shows the area where I believe my work can complement and expand on the existing *design* problem types.

As discussed in the previous section, Bernaras and Van de Welde present different models for design synthesis along both the construction dimension and from the types of design knowledge available, e.g. case-based reasoning. These models for design proposal effectively mirror the findings of other researchers previously described (Chandrasekaran [1990], Maher [1990] etc.).

I would tend to view the design synthesis process in a slightly different way and believe that the models for the different types of design are most effectively viewed as being different problem-solving methods which may be applied to the design synthesis (or proposal) task.

I also believe that the models presented by Bernaras and Van de Welde do not adequately reflect the way in which the use of different methods may be utilised within the concurrent design process. They do not explicitly make clear the way in which a number of different proposal methods may be used within a particular design episode. For instance, the way in which a large number of design problems are solved is via the decomposition-recomposition method. This implies that different methods may then be used for the design sub-problems generated, which imposes a dynamic, and complex control structure on the application of different sub-tasks.

Critically, in terms of the concurrent design process, they do not consider the way in which different life-cycle perspectives can impinge on the design process. I believe that the knowledge relevant to these life–cycle constraints should act as an important knowledge role input to the concurrent design process. It has previously been outlined how this consideration of downstream life-cycle perspectives is key to the concurrent approach to design.

The *Common*KADS library also includes a number of models for the modelling task (Top and Akkermans [1994]). They see modelling as being a form of design and their comments are very pertinent to the development of models for design. They believe the specify–construct-assess PSM is used for the modelling task. The specify sub-task is where the requirements for the model are outlined (similar to the analysis stage of design). The construct and assess sub-tasks can be seen as analogous to the propose – critique tasks outlined in Chandrasekaran [1990]. Top and Akkermans go on to describe the construct task as being where design alternatives are generated and also where additional constraints are identified.

Top and Akkermans also make some relevant points regarding the control structure governing the task. Ideally, they suggest the overall modelling task consists of a single specification step followed by an iterative process of construct–specify (or generate and test). In practice, they suggest that the specification step will be repeated a number of times to make implicit assumptions explicit. However, they also consider the case where multiple viewpoints must be considered during the process. They suggest that the construct – assess process must be repeated for each viewpoint. This corresponds to the situation I have outlined for concurrent design where knowledge from a number of different viewpoints must be considered during the design process. However, Top and Akkermans suggest that ideally this process considers a single viewpoint at a time and do not consider the case where multiple viewpoints have input to the process at the same time.

3.11 Constraints within models for design

Bernaras and Van de Welde characterise a problem statement as abstract customer ‘wants’ (needs and desires) which are refined to criteria (requirements descriptions) and then further refined to constraints (problem statements). These problem statements then act as constraints in the search for design solutions. However, in the engineering field, constraints are not merely formally expressed criteria, they are believed to be subtly different entities.

Hence I would view a problem statement as a design specification which contains functions (criteria which dictate what the design solution must comply with) and constraints (which dictate to an extent how the design solution is achieved). A simple example would be a design specification for a car where a function might be expressed as 'the car's maximum speed must exceed 150 KPH' while a constraint might be 'the design solution will be presented within six months'. As Chandrasekaran [1990] states *"The distinction between functions and constraints is hard to formally pin down, however a distinction is believed to exist"*. Chandrasekaran [1994] gives a review of work on functional representation. Wielinga and Schreiber [1997] state *"constraints differ from requirements in that requirements must be satisfied while constraints must not be violated"*. Brown and Birmingham [1997] succinctly capture the way in which I would view constraints as *"They describe what must not be violated"*.

Constraints may also be of two kinds. The example given above would be specified by a customer or market needs and will be termed an external constraint. However there will also be constraints on the design that will be dictated by perspectives downstream from the design process. A typical example might be that only a limited number of manufacturing processes are available to produce the resulting design. This type of constraint is unlikely to be of interest to the customer, hence it can be termed an internal constraint. These 'internal constraints' are key to the concurrent approach in that they represent how downstream perspectives can affect the design. Internal and external constraints in this context are to my own definition and are taken as being different from those outlined by Lawson [1990], where internal constraints are defined as being under the designer's control whereas external constraints are fixed externally (e.g. standards, customer dictated constraints etc.). Constraints are also sometimes viewed as being 'hard' or 'soft' depending on the degree to which they may be relaxed or ignored (see Rajeev et al [1993]).

3.12 Knowledge roles input to the design process

Another criticism of the models for design is that they are deficient in the knowledge roles that play a part in the design process. These are in addition to the roles representing life-cycle constraint knowledge, which have already been outlined. Clearly during the 'construct design' stage (and other stages), a complete design is not instantaneously generated. A design will evolve in stages. Hence clearly an important role is played by the current state of the design. This could possibly be a conceptual model or a fully detailed design, as the models analysed imply. However it could also be at some intermediate stage where some parts comprising a concept are detailed etc. This model could be physically manifested as sketches, CAD drawings etc. or some combination of these. Hence, this knowledge role, which will be termed the 'current design model', must play a continual role in the design process.

It may also be necessary to provide a number of different 'views' on this model so that personnel with differing expertise are able to analyse this model from their own perspective. E.g. a designer might be interested in the form of the product, hence a 3-D model would be required, while an assembly engineer would be interested in interactions between different components comprising the evolving design model.

The potential importance of the critiquing and negotiation tasks within concurrent design has already been noted. A number of different researchers have outlined models of how critiquing and negotiation may occur. It is now necessary to analyse where these models have been formalised as knowledge-level models and if not, the form such models may take.

3.13 Formalised models for critiquing

*Common*KADS does not contain a generic model for the critiquing process, however a number of researchers have suggested that critiquing as a task may have a number of generic features (Krishnamoorthy et al [1991]). *Common*KADS does include a number of models for the assessment task, see Valente and Locknehof [1994]. However critiquing is believed to be more encompassing than assessment as a task. The result of an assessment is basically an allocation to a decision class or some other factor chosen from a predefined set of results, such as a statement of difference. While a critique may include such an assessment, it is also likely to include a rationale for the critique and even some counter-proposal to the artefact (in this case a design) being critiqued.

The models for critiquing presented in the previous chapter have mainly been derived from work to develop computationally based critiquing agents and a number of different possible forms for the critiquing task were presented. However, a number of issues are pertinent to how human 'agents' involved in the concurrent design process may critique a design from different perspectives and it is informative to discuss at this stage how such issues could be represented in a task type structure.

Based on the task-type representation formalism of *Common*KADS, it is possible to outline some issues relevant to a model for critiquing based on the analysed literature. For instance, a critic may generate their own proposal for the portion of the design of interest before comparing with the originally proposed design. However, another approach is to outline sub-optimal aspects of the given design without necessarily generating a complete alternative solution.

A number of different inputs and outputs to the critiquing process were also evident. Clearly, the most important inputs are the evolving design that the critic is to critique and the specific domain knowledge of the life-cycle aspect that the critic represents. However, some knowledge of the capabilities of the original proposer of a design are also seen as being important. One possible motivation behind a critique might be to educate the original proposer of a design.

There can also be a number of different outputs from a critique. These include some assessment (either quantitative or qualitative) of the proposed design, an alternative proposal, an indication of areas where the original design is sub-optimal, some form of explanation or rationale for the critique and where different constraints have either been violated or even constraints the original design has failed to address

3.14 Formalised models for negotiation

In the context of design and engineering, there appears to be little in the way of formalised, knowledge-level models of the negotiation process. Klein and Lu [1989] outline five different computational models, as distinct from knowledge-level models, of conflict resolution. These are development-time conflict resolution, backtracking based failure handling, numerically weighted constraint relaxation, specific conflict resolution advice and general conflict resolution expertise. However, they acknowledge that compiling development-time conflict-resolution strategies for all eventualities may be prohibitively time consuming. Shaw and Gaines [1989] present a formal approach to resolving conflict at development time.

As for critiquing, a number of issues relating to a possible knowledge-level model for negotiation can be discussed. These include the nature of the negotiation. Conflicting forms of negotiation will likely result in different models for the negotiation task when compared to more cooperative forms of negotiation. Strategies such as concession making are also important while knowledge of constraints and other forms of knowledge are critical.

3.15 Chapter Summary

Techniques from artificial intelligence have been used to model the design process. These have resulted in both cognitive models and knowledge-level models. Models at the knowledge-level have been formalised as *Common*KADS models for design. *Common*KADS is seen, in Europe at least, as being the standard methodology for developing knowledge-based systems. This chapter gives a concise summary of the relevant aspects of this methodology.

*Common*KADS suggests that there are a number of different problem-solving types, design being one of these types. There are a number of dependencies between these types and the solution of a given problem, such as design, may involve a number of these different problem types. *Common*KADS also outlines a number of possible expertise task models, which describe how the design task may be accomplished by application of a suitable problem-solving method. These different models are discussed.

However, the existing KADS and *Common*KADS models for design do not implicitly support the process of concurrent design, where I have defined the principal distinguishing characteristic of concurrent design as allowing the input of knowledge from different life-cycle perspectives at the design stage. Critiquing and negotiation represent possible ways in which such life-cycle issues may be incorporated in the concurrent design process, however, no formalised, knowledge-level models for theses tasks are believed to exist.

In the next chapter, I go on to look at existing tools to support both the design and concurrent design processes. In particular, I analyse the underlying design models that these systems are based on.

4 Computer support for design

4.1 Introduction

In the two previous chapters I have described different models for design and how these have been formalised as expertise task models as part of the *Common*KADS methodology. A fundamental implication of *Common*KADS thinking is that such models will ultimately be used as the theoretical underpinnings of some computer-based tool, which either performs or supports the task in question.

In this chapter, I begin by discussing two fundamental ways in which computers can be used within the context of the concurrent design process. I then go on to look at how computers have generally been used within the context of both design and more specifically, concurrent design. In particular, I focus on the theoretical models, if any, underpinning such systems.

I conclude this chapter by summarising the issues that have been discussed so far in the areas of models for design and how they can support the concurrent design process. I then go on to look at how these models can be refined and expanded on in order to provide more comprehensive support for the concurrent design process.

4.2 Systems to do or support design?

An implicit assumption of some of the early work on AI in design (and also on other problem-solving tasks) has been that the complex cognitive activities associated with performing the task could ultimately be encapsulated within self–contained computer programs.

This essentially forms the basis of the research. That knowledge-level models for a particular process or task, in this case concurrent design, can ultimately be used as the basis for some computer-based tool to support the process or task. Liddament [1999] outlines the general assumption encountered in the literature that computers may be used to support the design process (and indeed should be used for such a purpose!):

"there is implicit in much of the writing about computationalism an a priori belief that human cognitive activity must be encodable in some specifiable set of explicit, unambiguous instructions of the type that could be produced in the form of a program to be run on a serial computer".

Liddament then goes on to question the entire philosophical basis upon which existing (and possible future) computer support for design is based. Liddament suggests that the computational paradigm is deficient (and possibly inappropriate) for supporting the design process. Liddament suggests that encapsulating the cognitive activities encompassing design is currently untenable. To a certain extent I believe this stance to be valid and this is why the goal of this research is the development of knowledge-level models for concurrent design as opposed to cognitive-level models.

However, I believe it would be inappropriate to conclude that computers cannot be used at all to support the design process in some way. Computers are clearly useful tools and are currently successfully used by designers to support them in different aspects of their work. I believe it would be counter-productive to ignore their possible use in supporting the design process. I believe that concurrent design, because of it's inherently multi–disciplined nature, requiring knowledge input from a wide and diverse range of sources, can greatly benefit from computer-based support.

The reflective practitioner approach to design also questions whether knowledge that is applicable to a particular design situation can be generalised to suit other design situations. Hence, based on the presented evidence, I believe that the idea of a generic 'design machine', whereby a computer programme automatically generates designs across a wise range of domains, from an input set of specifications, using the same cognitive mechanisms utilised by human designers, is an unrealistic goal at this time.

A number of tools, such as the ACDS system outlined in appendix B (see Darr and Birmingham [1994]) have had considerable success in generating designs, in a particular domain, using a fully automated system. The ACDS system exhaustively generates the space of all possible designs. However, I believe this is not a method commonly employed by human designers and scaling or extending such a method would be difficult.

An alternative is to support the design process with systems that act in a subordinate, supporting manner to designers and design teams rather than attempting to supplant human designers. Smithers et al [1990]:

"..,. which are simply able to create designs by some means or other. The other approach is to try to build systems, again usually computer programmes, which provide intelligent support to people doing design".

Lander [1997] also asks where strategic control for an application will lie. I.e. is it distributed in the different software agents or does it rest with the human users? My conclusions are distinctly biased towards the latter given my current understanding of strategic control of the design process. This is supported by Chandrasekaran [1990] who suggests that computer support for the design process is best supported:

"not by architectures that impose a monolithic task structure on the designer but rather more flexible architectures that allow designers to pick and choose different problem-solving methods for different parts (or sub-tasks) associated with the design process".

4.3 Computer support for concurrent engineering, design and concurrent design

Because of the uptake of concurrent design principles and techniques by industry, it is perhaps inevitable that a large number of research tools have been developed or are being developed to support the concurrent design process. Molina et al [1995] present a review of work to develop computer aids to the simultaneous engineering process. Culley et al [1996] indicate how computer-based tools and multi-disciplinary teams are currently being employed in UK industry.

Appendix B describes a number of additional computer-based tools that have been developed to support design and concurrent design. It must be emphasised that this is not meant to be a definitive survey of all such tools. Rather, I have presented a representative sample of such tools in order to outline typical characteristics shared by the tools.

Support for concurrent design can be classified as either enabling communication between personnel involved in product design or simulating the expertise necessary to support concurrent design using knowledge-based systems. However the two approaches are not mutually exclusive and systems and research described may contain elements of both approaches.

Much research has been done in the area of facilitating communication between team members at the level of systems, human factor aspects and computer-mediated co-operation. Many organisations that have implemented a concurrent design philosophy support distributed teams with computer networks etc. Fischer [1990] looks at some of the requirements of such problem-solving systems. The SHARE project (see Toye et al [1994]) consists of a number of sub-projects, the main aims of which are to support design teams by enabling shared understandings of designs through the use of information technology. Next Link (see Petrie et al [1994], Petrie et al [1995]) is one such sub-project. Next Link is a framework, allowing existing software tools (which support design) to communicate over the Internet. Such research extends to the emerging area of computer supported co-operative work (CSCW). Mccarthy [1994] presents a review of work in this area. Stevenson and Chappell [1994] review CSCW and PIM (Product information management) tools to support concurrent engineering.

However, while facilitating communication is clearly of vital importance in supporting the design process, this is not the main focus of this research. Rather, it is in the use of computational techniques to model the expertise of participants in the concurrent design process.

4.4 A discussion of some existing tools to support design

A number of the systems outlined in Appendix B and also other computer-based support for concurrent design have achieved success in supporting designers and design teams in the design process. However a common criticism aimed at some of these tools is that that they are not based on an underlying model of design which reflects what happens when real-life designers actually design (Landauer [1995]). They suggest that system development has been driven more by technical considerations regarding what computers can be made to do rather than in analysing how designers and design teams can best be supported.

Smithers [1996] argues that, in general, knowledge-based support for design has been implemented in a fairly ad hoc manner without the assistance of lessons from the field of knowledge engineering: *"so far all this engineering activity has been carried out in the absence of any useable theory or theories of design process" ... "designing gets characterised by what we can get computer programs to do, rather than what really goes on when professional designers design".*

To a certain extent, this is a valid criticism, especially in respect of explaining why there are many more failed systems than successful ones. A large amount of work I have analysed and reviewed concentrates on the computational aspects of the systems. This can include considerations such as whether an object-oriented approach should be used and which programming language a possible tool should be implemented in. One possible reason for this preoccupation with computational aspects is that a number of the tools outlined have been developed by technologists whose main areas of expertise lie in the computing arena and not design methodology and theory.

Generally the systems reviewed incorporate a fixed and prescriptive model for the design process. For example, the CACID system (Schmidt and Schmidt [1996]) assumes a functional decomposition and recomposition type model for the design process. This is a commonly encountered model prescribed by other systems. Ligman [1990] also advocates the use of functional attributes as a means of generating novel designs. By contrast, the SPARK system (Young et al [1994) uses a constraint satisfaction approach. However, as the preceding chapters have show, there are many other plausible models for the design process and a single design problem may incorporate the use of a number of different process models in its solution.

Smithers et al [1990] suggest that in artificial intelligence there are two distinct ways of studying design. One approach is to build systems, which attempt to replicate human design behaviour, the other is build systems which provide intelligent support to people doing design. I believe that the critiquing paradigm offers a more suitable method of supporting designers and design teams as opposed to a more traditional 'expert system' based approach (where the computer-based system effectively supplants an expert in some problem-solving task, such as design, rather than supporting them). Fischer et al [1993b] see an effective design support environment as a *"co-operative problem-solving system"*.

I would subscribe towards this view of computer-based systems to support design, which is summarised by Finger et al [1992] when they state *"an intelligent design system should aid the designer in understanding the interactions and trade-offs among different, even conflicting, requirements"*.

An increasing number of researchers are calling for more basic research into how designers actually work and think and the development of more complete models of the design process. Smithers [1996] in particular, has called for the development of knowledge-level models of the design process in order to facilitate the development of more effective computer support for design. Computer-based systems to support the design process should mesh smoothly with the social and organisational settings in which the system will be used. Landauer [1995] suggests that many existing IT systems have not been successful because these factors have been ignored. However these issues should not be taken as a blanket criticism of existing work that has been done. Clearly, a large number of very valuable and credible systems have been developed which go a long way to improving computer-based support for design.

4.5 A recap on design models, methodologies and tools

In this and the previous three chapters, I have outlined a number of different models, which model the process of design and discussed to what extent these support the concurrent design process, which has it's own distinctive characteristics.

Two distinct paradigms which describe design, the prescriptive and the rational pracrtitioner approach have been described and discussed. The prescriptive models for design give general, high-level guidelines as to how the design process should proceed but give little indication at a finer level of detail as to what actually happens when designers and design teams work. The rational pracrtitioner approach attempts to describe at a lower level of detail how elements of strategy are incorporated in the design process.

In addition, I have outlined how these models can be considered as cognitive or knowledge-level models. Research from the field of artificial intelligence indicates that the design process may be considered as a series of sub-tasks of propose-critique-modify. While I believe that the sub-tasks of propose-critique reflect how concurrent design can occur, the modify task is not felt to adequately reflect the sometimes complex process whereby conflicting critiques of a design are incorporated into a revised design solution or proposal. The concept of negotiation has been introduced as the process whereby such conflict resolution can occur.

A number of the prescriptive models for design have been formalised as knowledge-level models as part of the *Common*KADS methodology. However, none of the models developed as part of *Common*KADS explicitly supports the process of concurrent design. I have discussed where these models fall short in their potential support for the concurrent design process. Hence, I believe I have outlined where useful research can be done to develop knowledge-level models for the process of concurrent design. This would act to improve support for developers of knowledge-based systems in the field of concurrent design.

4.6 The experimental basis for existing knowledge-level models for design

In the previous chapter I have described a number of existing knowledge-level models for design developed as part of the KADS and *Common*KADS methodologies (attributed to Bernaras and Van de Welde [1994], Kruger and Wielinga [1993] and described in Tansley and Hayball [1993]). I have also briefly discussed the experimental methodology used by Kruger and Wielinga [1993] and noted how the study they used to inform their model for design was based on a protocol analysis study of a single designer.

However, perhaps surprisingly, Tansley and Hayball [1993] and Bernaras and Van de Welde [1994] make no reference to any case studies from which the presented models for design have been derived. In fact, Breuker and Boer [1998] suggest that a large number of the expertise task models (which are effectively based on different problem-solving methods as discussed in Chapter 2) in the *Common*KADS library have been derived from studying the available literature - *"most PSM's were collected from descriptions in the literature"*.

It is a central tenet of this research that if knowledge-level models are to adequately reflect the working practices of actual designers, the models must necessarily be informed by the problem-solving expertise these designers utilise in their work. This implies that, ideally, the models should be developed from observation and analysis of expert designers and concurrent design teams engaged in real-life design scenarios. This necessarily raises issues of how practicing designers and design teams can be observed and how knowledge elicitation can most effectively be performed in such situations.

4.7 Chapter Summary

A number of different computer-based tools to support the concurrent design process have been outlined. These systems either act in a communication-enabling role or play a more active part in the design process. The knowledge-based systems described range in scope from simple expert systems to multi-discipline design support systems. The researched literature suggests that knowledge-based systems should be used to support the designer or design team during the design process rather than attempt to replace them.

In addition, it has been discussed how most concurrent design support tools have not been developed based on an underlying theory or model for concurrent design. Tools to support the design process have been influenced more by computational considerations than by any model of how designers work (or indeed would like to work). I believe that a knowledge-level model for the concurrent design process would usefully inform the development of such systems. In order to do this, it is necessary to consider how best to experimentally develop such models.

The experimental basis on which existing *Common*KADS models have been developed has also been questioned. As far as can be determined from the available literature, the models have been developed more by thinking about the processes or tasks in question, rather than by experimentally observing and analysing personnel involved in the activities.

In order to evaluate possible ways in which concurrent design may be experimentally analysed, in the next chapter I will go on to discuss different ways of experimentally analysing expert behaviour and in particular, methods that have been used to analyse the design and concurrent design process.

5 Methodology

5.1 Introduction

Techniques and approaches to the analysis of human problem-solving continue to develop and is an ongoing field of enquiry. As a result, there is no absolute, guiding methodology for the analysis and modelling of human problem-solving behaviour.

A number of different techniques have been used to analyse human activity while performing some complex task (this has often been done with a view to implementing this behaviour in a knowledge-based system). This coaxing of knowledge out of an expert or experts is usually termed elicitation. I begin by analysing how different researchers have attempted to analyse design activity. An important aspect involved in analysing design (or any other problem-solving activity) is the way in which the observer affects the outcome of the expert's problem-solving. A key summary of this evaluation of techniques that have been used to analyse design is that while extensive research has been performed and indeed is still ongoing as to how best to analyse design behaviour, there are no definitive or concrete guidelines for this process.

As a result, this chapter then goes on to present a brief but focussed review, which discusses the most widely used elicitation techniques in domains other than design and outlines their strengths and weaknesses. Firlej and Hellens [1991], Wielinga et al [1990] and Jackson [1990] act as a good starting point for further work in this area.

Broadly speaking, methods for developing knowledge-level models can be split up into top–down and bottom–up methods. This chapter assesses the strengths and weaknesses of the two different approaches and their suitability for eliciting and modelling concurrent design behaviour. A number of the top-down methods, in particular, are advocated by *Common*KADS as suitable means of knowledge elicitation and expertise model development.

I finish by describing my own methodology for analysing concurrent design behaviour which utilises elements of both the top–down and bottom–up approaches. I discuss the thinking behind the evolution of the method and how this was used to drive the development of an initial knowledge-level model for concurrent design. Throughout the discussion I use the term 'knowledge engineer' to describe the role played by myself in the knowledge elicitation process.

5.1 Methods that have been used to analyse design behaviour

Knowledge elicitation can be defined as the process whereby a knowledge engineer attempts to glean (or coax out) and then verify information and data from an expert, regarding their expertise in some specialist task. Knowledge acquisition is a more all encompassing term and implies the acquisition of knowledge from all sources including books, technical manuals etc. as well as from human subjects and it's subsequent representation in a computable form. Gruber and Russel [1991] use the term 'design knowledge capture' to describe this eliciting, recording and subsequent modelling of design knowledge.

The analysis of a complex problem-solving task, such as design, is potentially a very difficult undertaking. I will begin surveying some contemporary studies of the design process, which principally make use of the technique of protocol analysis.

Protocol analysis is one of the most widely used techniques for attempting to determine cognitive behaviour while performing some complex task. This technique involves the subject 'voice their thoughts' while performing the task. Clearly, this gives more insight into the task than might be gained by merely observing the subject perform the task in silence. Typically, the subject will be recorded while the task is being performed (usually audio but sometimes video recording). The recordings are then usually transcribed to give a transcript.

The technique of protocol analysis emerged from psychological research during the early part of the century. However it was not until the wide availability of tape recorders that studies of protocol analysis became a practical possibility. Some of the earliest studies include de Groot [1965] on chess playing and Newell and Simon [1972] on logical problem-solving.

The main limitation with this approach is the assumption that people can accurately articulate their cognitive processes as they perform some task. The actual process of talking can also affect the experts' problem-solving behaviour - Bainbridge [1979] *"Talking alters the experts thinking process"*. There can also be practical problems in some situations due to factors such as not being able to hear the 'expert'. Other researchers, including Goguen and Linde [1993] have criticised protocol analysis because it fails to take account of the social context within which the expert's 'voicing of their thoughts' occurs. They term protocol analysis an 'unnatural discourse form'.

However, despite these drawbacks, protocol analysis has been used in a number of different domains, although it is only relatively recently that design activity has been studied using this technique. Relevant research will now be discussed.

The complexity of studying real-life design episodes has been noted in a study by Valkenberg and Dorst [1998]. In their study, Valkenberg and Dorst analyse the reflective design practice of teams of student designers by dividing design protocols into 'episodes', where each episode is believed to be pertinent to a particular mode of the design process. They also give some other useful insights into this process. However, I believe that student designers do not necessarily design in the same way as experienced, professional designers, a view that is endorsed by Schon [1991]

In the study presented in Kruger and Wielinga [1993], a designer is observed performing a given design task. Again, protocol analysis is used to analyse their performance. However, Kruger suggests there is little evidence of the type of concurrency I have discussed displayed by the designer. A contemporary study by Gero and McNeill [1998] attempts to extend the protocol analysis technique through the use of a domain-dependent coding scheme to give a richer coding structure.

Protocol analysis has also been extended to analyse teamwork activity, where instead of having subjects 'voice their thoughts' it is the verbal interchanges between team members that acts as the basis for analysis. By looking at how different researchers have used this technique to analyse design activity, the strengths and weaknesses of the technique for divulging design behaviour can be assessed.

Cross et al [1996] give a very interesting account of a study comprising twenty different research centres involved in analysing design activity. The different researchers or research teams were all given the same tape recorded and videod protocols of both an individual designer and different design teams working singly or collaboratively on the design of a carrying rack for a bicycle. The resulting analysis of the protocols show a number of different views of the design process. This study has generally become known as the Delft workshops and its influence on subsequent protocol analysis studies of design activity has been extremely significant.

Interestingly, most of the studies neglected to make use of the videod protocols and concentrated on the tape-recorded protocols. This could be due to the difficulty of analysing video footage of designers working.

The study analysed in the so-called 'Delft workshops' has a number of designers working on a design simultaneously. I feel that such a scenario, while a realistic reflection of how design can occur, diverges from a true concurrent environment as all three designers had similar aims and objectives (i.e. the successful completion of a design). Their areas of expertise were also similar and this is felt to be unlike a typical concurrent design environment where the inputs from different personnel are likely to be conflicting in nature. However the experiment resulted in a number of original insights into the design process.

Some of the 'Delft workshop' studies also use a coding technique, whereby different parts of a transcript are 'labelled' in accordance with a pre-defined coding scheme, in order to analyse transcripts obtained from protocol analysis of design sessions. However, the choice of coding initially used will inevitably affect the results. This coding technique has also been used to analyse video recordings.

Lloyd and Scott [1994] analyse five different engineers performing a similar design task. They also analyse the protocol results using statistical analysis but then conclude that qualitative methods are probably more suitable for analysing the designer's behaviour.

In addition, there are a number of other drawbacks inherent in the protocol analysis techniques. Typically, one hours tape recording can take five hours or longer to transcribe, although tools such as data dictionaries can also be used to aid in this analysis of transcripts.

The problems in having designers 'think aloud' as they accomplish a design task are discussed in Galle [1996]. Galle goes on to suggest 'replication protocol analysis' as a means of attempting to deduce design behaviour. By this technique, a third party is asked to comment, in a retrospective manner on what they perceive to be the steps implied by a completed design. Again, protocol analysis is used to analyse the results.

Stauffer and Ullman [1988] suggest that *"Direct observation is a non intrusive method"*, however Davies [1995] argues that analysing design activity by having designers 'voice their thoughts' while designing can fundamentally alter the design process and has limitations in revealing design cognition. Clearly these observations have implications as to a suitable knowledge elicitation technique to use in order to analyse an expert's steps in performing some task.

Gruber and Russell [1991] outline how design rationales can be used to 'explain' or 'justify' how a particular design is generated. In particular, they note the problem of analysing expert behaviour in a group who can see no obvious benefit to themselves from taking part in the analysis. Their method also requires that extensive design histories are already available for the given design in order to stimulate discussions regarding design rationale.

As a result, it is pertinent to summarise that a variety of methods and techniques, particularly protocol analysis, have been used to analyse design activity. However, there is no common consensus as to the correct methodology to use, although notable attempts to develop such methodologies are ongoing – see Davies [1995], Dorst [1995] and Lloyd et al [1995]).

A number of researchers have commented on the lack of basic research in this area. Cross et al [1996] suggest that *"the industrial design domain has been studied relatively little"*, while Stauffer and Ullman state *'relatively little research has been based on empirical evidence, especially in mechanical design'*. Blessing [1994] gives a comprehensive listing of existing studies in engineering design and concludes that *'the number of detailed studies in industry is low'*.

The need for research in this area is further outlined by Gero and McNeill [1998] *'there has been remarkably little research on capturing, presenting and analysing the activity of designing as carried out by human designers as a set of phenomena to be modelled'*.
Because of its' inherently multi-disciplinary nature and the resulting 'conflict' of perspectives that can occur, concurrent design as a domain presents a number of interesting and novel problems for the knowledge engineer attempting to elicit and model knowledge. Klein and Lu [1989] in their study of co-operative architectural design state that *"knowledge acquisition in co-operative design presents special challenges and requires additional techniques compared to traditional knowledge acquisition"*.

Clearly, in order to develop knowledge-level models for concurrent design, some means of analysing this activity is needed. I will now go on to look at additional techniques that have been used to analyse expert behaviour in other domains and what application these may have in analysing design behaviour.

5.2 A brief review of knowledge acquisition and elicitation techniques

Musen [1993] presents an overview of knowledge acquisition. In particular, Musen outlines a number of perceived pitfalls and difficulties that may befall the knowledge engineer when attempting to elicit and model expert problem-solving knowledge:

- Tacit knowledge. Experts are not always able to introspect accurately and are not always able to explain their expertise.

- Communication problems can occur between expert and knowledge engineer, as the two do not always share a common ontology or vocabulary. This can be a particular problem when the knowledge engineer first encounters the domain of interest.

- Knowledge representation methods used by the knowledge engineer may not be applicable or suitable for representing knowledge in the domain.

- Failure to create a 'deep' model of the knowledge within a knowledge-based system can lead to the system becoming 'brittle' when faced with unusual or unexpected problems.

Clearly, these issues are of fundamental importance to developers of models of expertise and knowledge-based systems.

The elicitation phase of knowledge-based system development has traditionally been the bottleneck which has hampered the development of such systems, therefore extensive research has been conducted in this area and a number of techniques and tools are available to the knowledge engineer to aid in this process. For example, the ACACIA project is an attempt to develop tools and a methodology for knowledge acquisition (see Dieng et al [1994] for further details) while KADS was originally developed as a knowledge acquisition methodology.

The field of ethnography outlines issues that must be considered when studying human behaviour (see Bucciarelli [1988]). These more 'human' issues expand on the technical issues previously discussed from Musen [1995] and include the possibility of ambiguities arising between participants in the process in question and the context in which the process occurs. Other possibilities to consider include:

- The expert may be uncooperative and not wish to 'share' their problem-solving expertise. This may especially be the case if the expert perceives any resulting knowledge-based system as a threat.

- The expert may be constrained by time-scales and not see any immediate benefit to them from a potentially obtrusive and time-consuming knowledge elicitation exercise.

- The personality of an expert can affect the elicitation process. An introverted expert may be reluctant to discuss issues with the knowledge engineer while a more extrovert expert may attempt to provide too much information or digress excessively from the focus the knowledge engineer wishes to pursue.

- As has been discussed for protocol analysis, analysing an expert performing some task may fundamentally affect the way in which they perform the task.

Major and Reichelt [1990] split elicitation techniques up into standard or contrived techniques, standard techniques being for general purpose knowledge acquisition and contrived techniques being used for more focused and specific knowledge elicitation. In addition, elicitation and modelling techniques can be characterised as either bottom-up or top-down in nature.

The elicitation and analysis techniques previously presented and discussed have indicated how data is collected from observations of expert problem-solving. This data can then used to generate models or hypotheses representing this problem-solving behaviour. This represents what I shall term a bottom-up approach to the development of models of problem-solving behaviour. This is analogous to the grounded theory approach (attributed to Glaser and Strauss [1967]) where analysis of data is used to derive hypotheses and models of expertise in a 'bottom-up' fashion.

However, models can also be developed in a top-down manner. By this approach, models are generated or hypothesised and experimental data is then used to refine and validate these models. I will begin by analysing methods for knowledge elicitation which are essentially bottom-up in nature.

5.3 Bottom-up techniques for knowledge elicitation

Because of the data-driven nature of the bottom-up methods of knowledge elicitation, the gathering and analysis of data is essential to these techniques.

5.3.1 Standard techniques for knowledge elicitation

These include interviews (structured and unstructured), thinking aloud techniques, direct questioning and ethnographic studies (usually involving an 'expert' or team of 'experts' being videod performing a task),

5.3.1.1 Direct questioning

Direct questioning techniques can be useful for eliciting specific, concise information from an expert. However this technique assumes the knowledge engineer is in the position to ask the 'correct' questions, Musen [1993] - *"the use of direct questions to elicit knowledge however assumes the knowledge engineer asks the right questions and the expert gives the answer, not what the expert perceives is a plausible answer"*. The comments regarding direct questioning also apply to a large degree to the use of questionnaires. Goguen and Linde [1993] analyse elicitation techniques for use in requirements engineering. However, their findings are very pertinent to this research. They discuss the use of questionnaires and outline their limitations in that they are effectively a closed form of interaction with a pre-determined set of possible answers.

5.3.1.2 Interviews

This is probably the most widely used technique for eliciting knowledge from an expert. Firlej and Hellens [1991] outline the importance of the interview in the elicitation process *"we stress the interview as the central, practical tool for successful elicitation"*. However Firlej and Hellens [1991] also *suggest "Experts do not always know what they know"*, hence they go on to suggest that the interviewing process can become more focused as the knowledge engineer becomes familiar with the domain being analysed. The knowledge engineer may start with an overview interview in order to become familiar with the domain followed by increasingly more structured interviews. Following each interview, the knowledge engineer should analyse the results of the interview (notes, transcripts etc.) and review the findings before following up with more interviews.

Goguen and Linde [1993] see open ended interviews as having more potential for being successful but stress that problems may occur because of the different category systems (I.e. words mean different things to different people in different contexts). They see more general discussion as being a technique to elicit knowledge that interviews may fail to unearth.

The interview is a tried and trusted technique in the field of knowledge elicitation. However in order to elicit information beyond general domain knowledge, interviews must become increasingly structured from one interview to the next, as the knowledge engineer becomes more familiar with the expert's domain.

5.3.1.3 Approaches based in ethnomethodology

Goguen and Linde [1993] also discuss techniques from ethnomethodology, which can be used to analyse conversations. For example, discourse analysis can be used to analyse structures larger than sentences while conversational analysis attempts to describe the underlying social context that makes conversation intelligible.

An interesting (and valuable) aspect of conversation analysis is the way in which, unlike interviews, conversations are not pre-planned or ordered in any way: *'the order of interaction is negotiated in real-time as the conversation proceeds'* (Goguen and Linde [1993]. This means that issues such as differing terminologies can be examined and discussed as appropriate as the conversation between the knowledge engineer and expert progresses.

To ensure each participant contributes to the conversation, informal techniques such as 'turn taking' can be utilised. Narratives from experts can also be important constructs for divulging complex expertise and illustrating how particular problems may have been solved historically.

5.3.1.4 Recording techniques

Clearly, for interviews, thinking aloud analysis, questioning and also the contrived techniques for knowledge elicitation, some means of recording an experts utterances (and possibly actions) is necessary in order to be able to analyse the expert's behaviour. Firlej and Hellens [1991] suggest that tape recording should be used wherever possible although issues regarding security and secrecy on the part of the expert may preclude this.

Ethnographic methods may involve the use of video to analyse personnel operating in a real-life work setting. Videoing an expert or experts performing some task clearly gives insight into the manual steps such a task involves, however it will not necessarily reveal much of the thought processes utilised by a designer.

However, there are also practical constraints implied by videoing practicing designers in industry. These include issues such as security of information, regular access to the experts, analysis of huge amounts of video data etc. This view is echoed by Musen [1993] who states *" the considerable inconvenience and expense of ethnographic field work, however, is a major barrier to widespread adoption of these techniques".*

Firlej and Hellens [1991] also suggest that while the knowledge engineer taking notes or relying on memory have been used in the past, these techniques are not to be recommended as reliable recording processes!

5.3.2 Contrived techniques

The so called 'contrived' techniques are usually used to determine more specific information once a knowledge engineer has become relatively familiar with the 'experts' domain. These include sorts, repertory grids and laddering techniques.

5.3.2.1 Sorts

Concept sorting is based on the theory that people make extensive use of categories schemes to order their knowledge and are able to describe their own categorisation schemes with reasonable accuracy. Concept sorting involves an expert being given cards (or some other object) depicting or representing a number of different concepts (typically derived from an interviewing process). The expert will then be asked to sort the cards into groups. In this way, relationships between different concepts can be revealed. There are a number of variations on this technique. These include 'Q sorts', 'hierarchical sorts' and 'all in one sorts'. Rugg and McGeorge [1997] outline these sorting techniques in more detail and Canter et al [1985] outline a multiple sorting procedure.

However, both sets of researchers also outline a number of disadvantages of these sorting techniques. One of these is the way in which the knowledge engineers choice of elements for use in a sorting procedure has a decisive effect on the procedure's success in divulging an expert's categorisation scheme. Rugg and McGeorge [1997] also state that *"...sorts only address static, flat, explicit knowledge. They cannot conveniently access knowledge about sequencing procedures, trade offs .."*. This places limitations on the use of sorts to elicit the process-type knowledge inherent in performing some task such as design.

5.3.2.2 Laddering

This technique is typically used for eliciting hierarchical type structures inherent in experts' thinking. A typical application of this technique would be in determining a hierarchy of concepts. By taking one concept the knowledge engineer perceives as belonging to a particular group, the expert can then be quizzed as to whether subsequent concepts are sub-concepts, super-concepts or unrelated to the initial concept.

5.3.2.3 Repertory grids

This technique involves asking an expert to 'rank' particular objects on a scale where the end points of each scale will be defined. A simple example might feature a concept such as 'expense', the scale being defined as running from 'cheap' to 'expensive'. An expert could then be asked to place an object (say a manufacturing process) somewhere on this scale. This technique is generally believed to derive from 'personal construct theory' - Kelly [1955]. Statistical analysis is one of the most common methods of analysing data obtained from knowledge elicitation sessions using contrived techniques. However a number of researchers have suggested that the major attraction of this (and other) contrived techniques lies not in their effectiveness in revealing an 'experts' domain knowledge but in the ease with which results can be mathematically analysed. This is supported by Canter et al [1985] who suggest that *"the attraction of repertory grid techniques have been governed more by the simplicity by which results from this technique can be analysed more than any ability of the technique to elicit an experts thinking".*

5.3.3 Computer aided techniques

Computer aided techniques can be broadly split into two categories. CASE tools can be used to support both the standard and contrived elicitation techniques that have been outlined, while automated knowledge elicitation systems can be used to effectively take the knowledge engineer out of the elicitation loop and allow an expert to directly input 'knowledge' into a system.

A number of tools exist to support existing knowledge elicitation techniques. These range from augmented text editors for analysing protocols from interviews etc. to more ambitious integrated systems. ALTO is a tool to support laddering (see Major and Reichgelt [1990]), Shelley (Anjewierden et al [1990]) supports the analysis and development stages of a knowledge-based system and assumes elicitation is performed using protocol analysis. Shelley does not actively support other elicitation techniques. Gaines and Linster [1990] and Linster [1993] attempt to tie together a number of different tools to aid in the elicitation and design phases so that as knowledge changes, so does the design of the system. The *Common*KADS workbench (see Kingston et al [1995]) provides CASE tools to support the elicitation process and provides direct support for the *Common*KADS methodology.

However, as well as using CASE tools to aid or support the aforementioned elicitation techniques, a number of researchers have attempted to automate the elicitation process itself. This is clearly a more ambitious step than attempting to support existing manual techniques.

Hart [1985] outlines machine-based rule induction as an automated elicitation technique. OPAL (Musen et al [1987]) uses a domain model to automate knowledge elicitation from an expert. Jackson [1990] outlines programs that work by machine learning. These are guided by the learning strategies employed by the systems.

However, a major disadvantage of these tools is that they generally assume some structure on knowledge within a domain before an 'expert' can interact with the system. This generally limits the use of such tools to augmenting the knowledge already contained in a fairly complete expertise model. Also, the considerable time and effort involved in developing such automated elicitation system precludes their use in anything but very large studies or research applications.

A perceived weakness in the bottom-up approach to model development is in the time and resources required to derive models from a purely data-driven approach - *"It should be stressed that data-driven modelling is costly in terms of resource (knowledge engineers, experts etc) and time"* - Wells [1994]. In addition, the lack of a common ontology makes communication between the knowledge engineer and expert difficult. However, this bottom-up type approach is not the only way in which models may be developed.

5.4 Top-down approaches to expertise model development

Wells [1994] suggests that *"task modelling is likely to be carried out top-down i.e. by task decomposition. However, it may be necessary in a complicated domain, to identify tasks being carried out in transcripts"..."data-driven modelling should be used when the application is unresearched. If the library can provide sufficient support for a select-modify approach, for instance a generic task is available as a starting point, this should be used in preference to data-driven modelling".*

The main advantage of the top-down approach is that an ontology for the domain can be constructed as the top-down models are being developed. The technique of first defining an ontology for a domain before attempting any form of knowledge elicitation has been utilised by other researchers, in particular, by Gruber and Russell [1991].

The original KADS methodology advocated that expertise task model development should be carried out in a top–down manner (see Tansley and Hayball [1993]). However these top-down approaches have been expanded on by *Common*KADS and Akkermans et al [1994] identify three different top-down methods via which *Common*KADS expertise models may be developed.

5.4.1 Select and modify approach

The task and inference structures are assumed to be predefined as far as possible and are selected from a library of existing *Common*KADS models. One of the key goals of *Common*KADS is to provide libraries of different expertise models and problem-solving methods to aid knowledge engineers developing systems in different domains. Knowledge elicitation with the expert is then mainly concerned with instantiation of the domain layer and subtle refinements to the task and inference layers. Musen [1993] suggests that choosing the best conceptual model for a task at a suitable level of abstraction can greatly simplify the knowledge acquisition task. This is essentially a top–down method of model development.

However, Orsvarn et al [1994] comment on the select modify approach: *"there are hardly any examples of KADS 1 expertise models that have been developed on the basis of an interpretation model without modifying it"*. An important reason is that there can be a great deal of variation between tasks of the same kind.

5.4.2 Compositional modelling from library elements

An expertise model is constructed in an incremental manner from existing generic components (typically provided by the *Common*KADS library). This is done at a lower grain size than the select and modify approach and is more flexible in that a complete task and inference layer is not assumed. However this can make the modelling task more difficult. This technique combines elements of both the top-down and bottom-up approaches.

5.4.3 Refinement approaches

Model construction is driven by the introduction of subsequent refinements. Generally this involves starting with a very high level function structure for the expertise model and slowly refining various parts to reveal the 'true' model (E.g. tasks initially assumed to be primitives can be successively decomposed into their constituent subtasks etc.) Library components can be used although the resulting expertise model may not be structurally similar. This approach is often used for new and novel model development. This technique can also be classified as essentially a top-down type approach.

The differences between the refinement approach and the select-modify approach are not completely clear cut. However the select and modify approach implies that a model similar to the intended model already exists in the *Common*KADS library, while the refinement approach implies that a general high level functional model is developed by the knowledge engineer and then successively refined down to the level of individual inferences.

The scope for combining different approaches can clearly be seen in that using a refinement approach, a modelling element from the *Common*KADS library could be introduced during the refinement process along the lines of the compositional modelling approach. This is supported by Wells [1994] who shows how a number of different approaches are combined during the development of a particular expertise model.

Further details of the strengths and weaknesses and the stages and activities involved in model development using these different approaches can be found in Wells [1994], Orstvarn et al [1994].

5.5 A summary on techniques and approaches to knowledge modelling

In this chapter I have discussed a number of techniques used to aid the knowledge elicitation and modelling processes. This analysis has indicated that a number of different techniques (including the contrived and standard techniques, protocol analysis etc) and approaches (top-down and bottom-up) are available to the knowledge engineer. In particular, a number of these techniques and approaches have been used to analyse design activity. *Common*KADS also provides a number of guidelines as to how the knowledge elicitation and modelling process should proceed. I will now go on to analyse the implications of these findings and discuss how these influenced my choice of method for modelling and analysing concurrent design behaviour.

While a number of different techniques and approaches have been used with varying degrees of success in different domains, there seems to be no definitive guidelines as to which techniques and approaches would be most effective in analysing the problem-solving domain of concurrent design. This is summarised very succinctly by Rugg and McGeorge [1997]: *'Although it is clearly essential to choose the correct technique for a task, and to use it correctly, there is surprisingly little guidance on this in the literature'.*

The technique of protocol analysis has been used to analyse the design process. However, while this technique has been relatively successful in this area, a number of flaws can limit its effectiveness. In particular, researchers have questioned whether experts are able to verbalise their cognitive abilities and also the effect that attempting to verbalise their thoughts actually has on their ability to perform the given task.

A number of techniques for recording experts performing some task, typically involving discussions and interviews, have also been outlined. The resulting transcripts can give insight into the expert behaviour involved. A number of researchers have used coding schemes to attempt to analyse such transcripts. However, the choice of coding chosen by the knowledge engineer can have a fundamental affect on this process. A more qualitative analysis of transcripts can be seen as an effective means of transcript analysis.

A number of studies discussed have also made use of student 'experts' or actual experts in an artificial or laboratory setting. Different researchers have outlined how student designers do not necessarily solve problems in the same way as experts. In addition, the problems inherent in experts performing a task in an unusual setting or context have also been discussed.

Bucciarelli [1988] sees design as a social process and concludes that different participants can have quite different internal representations of a design and suggests that knowledge-based systems to support design must accommodate these differences in perspective. Because of the lack of a common ontology to allow communication between the knowledge engineer and design experts and the constraints this imposes on the elicitation process, I decided that a purely data-driven or bottom-up approach would not be an appropriate course to take. Some general model of problem-solving in the domain of concurrent design would be required in order to drive data-driven knowledge elicitation in the domain.

However, a criticism that has been levelled at a number of prescriptive models for design (see Lawson [1990]) is that they have been developed more by thinking about how design occurs rather than observing actual designers in the field. As a result, they do not accurately reflect the expertise of practicing designers. The dangers of relying on introspection as the sole means of knowledge elicitation are also outlined by Goguen and Linde [1993]. The implication is that any models developed using such methods need to be extensively validated using empirical data.

These issues helped to form my opinions of how knowledge elicitation in the domain of concurrent design would most effectively allow the development of credible models of expert problem-solving behaviour. My analysis of the domain suggested that elements from each of the different techniques and approaches could be useful for the development of knowledge models.

5.6 Characteristics required of an elicitation and modelling technique

At this stage it was necessary to determine what characteristics and features I considered important when attempting to elicit and model the concurrent design process. This was based on my previous discussions of techniques that have already been used to analyse the design process, other available techniques and their relative strengths and weaknesses. A key intention of the research was to analyse and model design tasks at a more detailed level than the existing *Common*KADS models, with a view to then abstracting developed models to be more generic. As a result, desirable features of the elicitation and modelling method were:

- To study design in a realistic a setting as possible with 'real-life' designers working on real projects. A considerable amount of published research on analysing design activity involves either case studies involving student designers (Valkenberg and Dorst [1998]) or laboratory based experiments involving either single designers or teams of designers (Cross et al [1996], Kruger and Wielinga [1993]). However, while these studies have made significant contributions to my current understanding of design, because of the nature of concurrent design I wished to study the process in as realistic a scenario as possible. Ideally I wished to analyse designers and teams of designers in their everyday work setting on real design problems and solutions.

- Make use of teams or groups of people with sufficiently polarised aims and objectives in order to achieve a realistic 'concurrent' scenario.

- Achieve a reasonable level of articulacy in the domain of concurrent design before coming into contact with concurrent designers and design teams in order to ensure we were 'talking the same language'.

- Use consistent and formalised knowledge representation techniques when deriving models of problem-solving behaviour in the domain.

- Make as much use as possible of the more 'qualitative' knowledge elicitation methods (such as interviews, discussions and narratives) when analysing designers.

Based on these observations, the scope for developing a novel elicitation and modelling technique began to emerge.

5.7 A novel method for knowledge elicitation and modelling

Compositional modelling implies that a general model for the domain exists which must be instantiated, e.g. from the *Common*KADS library of elements. The *Common*KADS library already contain a number of models for design (Bernaras and Van de Welde [1994]) however none of these directly address concurrency in design. Because a key aim of the research is to analyse these concurrency issues, I decided that these were not suitable starting models to analyse concurrent design behaviour.

Due to this lack of any general models for concurrent design problem-solving, my chosen approach was to derive an initial top-down model for concurrent design. This was done using the available literature as the driving force for initial model development. In the role of knowledge engineer I researched established design texts and current papers on concurrent design to generate initial models of problem-solving in the domain. These were represented as expertise task models, using the *Common*KADS modelling formalisms. As a result, the initial model used the abstract terminology of respected design texts to represent key tasks and knowledge roles comprising the overall concurrent design process. The different models were effectively refinements or formalisms of models already existing in the literature, albeit not in the representation formalism specified by *Common*KADS.

As a result of this approach the initial model development was accomplished more quickly than would have been possible through a purely data-driven approach using a technique such as protocol analysis. Hence I would consider this an essentially top-down approach to initial model development. However, it must be noted that even using this approach, model development was an extensive, time consuming process and it took several months of research and iteration to develop initial top-down models.

These initial models, developed from analysis of the literature, were then discussed and analysed with academics (who are experts in the domain of concurrent design). This refined the developed models and also prompted further research of the literature. This further iterative process involved considerable effort on the part of the knowledge engineer, over a period of several months, resulting in the generation of initial models for concurrent design. This process is discussed in more detail in Chapter 6 and is also described in Barker et al [1996a], Barker and Meehan [1999].

However, as I have previously discussed, a number of researchers have criticised model development which is driven by 'thinking about' problem-solving in a domain. Hence I wished to validate and refine these initial models with more empirically based research.

5.8 Validation and verification of my initial models

By developing initial models for concurrent design in a top-down manner, these models could then be validated and verified by assessing if they accurately reflect how concurrent design really occurs in an industrial setting. *"Validation concerns the compliance of a system with the user needs and requirements". In software engineering, verification is usually defined as "the demonstration of consistency, completeness and correctness of a system in each stage of it's development"* (Wielinga et al [1994b]).

Compared to traditional software implementations, knowledge-based systems (and by implication the underlying knowledge-level models) provide special problems during the validation and verification stages. Traditional software systems are generally based on definite algorithms, which dictate system performance and correctness. However output from a KBS is generally governed by heuristic methods which may only be correct under certain circumstances and assumptions. This mirrors the human world where two experts may give conflicting 'solutions' to a problem. This does not imply that one expert is 'wrong', more that in the field of knowledge-based systems, there is usually no definitive 'right;' solution to a problem. This makes formal validation of KBS extremely difficult.

Traditionally, validation of KBS has been performed by comparing a number of test case results from the completed KBS, with solutions obtained from 'experts'. Clearly, this is a late stage to be checking the performance, as a number of potentially unchanging implementation decisions will have been taken by this stage. I am more concerned with the verification of the developing expertise task models at each stage.

Wielinga et al [1994b] give a number of approaches to validation and verification, however these are at a very formal, abstract level. I.e. they do not give any concrete practical, guidelines that knowledge engineers should follow during the validation and verification of knowledge-based systems.

Clearly, it would be unrealistic to develop design tools based on the developed models to test the validity of the models. Rather, what is needed is some means of analysing the processes and methods designers working in industry currently use when designing in a concurrent manner. By developing initial top-down models for concurrent design, these represented a suitable starting point to guide the knowledge engineer in this analysis. In particular, they provided the knowledge engineer with a common ontology for discussing processes and entities in the domain of concurrent design with design personnel.

*Common*KADS suggests that a possible model validation technique to use is to develop graphical models for an expert's expertise and then use these models as a basis for further knowledge elicitation with an expert as to the validity of the models – *"the models are then a common ground for expert and knowledge engineer to communicate about some problem-solving activity"* – Van de Welde [1994]. Rademakers and Vanwelkenhuysen [1993] suggest that *"the expert should play a central and active role in the model construction process"*.

In this way, the developed models can be validated and refined by more focussed knowledge elicitation sessions with the expert. However, I have not found any significant references to the use of this technique, particularly in relation to analysing design activity.

The use of such graphical languages for knowledge representation has been noted by other researchers. For example, Oxman [1995] suggests that the use of such Visual languages should be encouraged *"... a body of tools for the observation and recording of design behaviours should be identified and further developed. This work would include, for example, the development of visual languages... "*.

5.9 Techniques to use in the refinement of the initial models

In general, the contrived techniques discussed earlier are used for focused knowledge elicitation when the knowledge engineer already has a fairly clear understanding of the domain being analysed and for clarifying certain ambiguous areas of an expert's problem-solving processes. Contrived techniques can also be used to instantiate domain models and refine task and inference structures. This implies that a knowledge engineer already has a fairly complete knowledge model for an expert's problem-solving before the different contrived techniques are used to further refine these models. Hence these techniques would be most useful at the later stages of a knowledge elicitation exercise to refine an already developed model.

I decided that a number of the different standard techniques, particularly open-ended interviews, discussions and conversations would be particularly appropriate techniques to use. In addition, tape and video-recording offered a practical way to record such interactions. I saw transcript analysis as an effective way to refine the initial top-down models and drive more bottom-up models of problem-solving.

The need for more studies of designers actually working in an industrial setting is supported by Akin [1995] *"It is important that more effort is focused on practice outside of the laboratory context"*. In fact, Stauuffer and Ullman [1988] outline how *"While there is a significant body of research in the study of design, relatively little research has been based on empirical evidence, especially in mechanical design."*. Hence I decided to try and study designers working in a work-like setting, if possible, rather than having designers' work on set problems in a laboratory type setting.

By attempting to use different knowledge elicitation techniques in this context and my initially developed models as a starting point, the effectiveness and applicability of each technique could also be assessed.

The *Common*KADS workbench (see Toussaint et al [1994], Kingston et al [1995]) provides support for expertise modelling within the *Common*KADS methodology and also provides limited support for a variety of different standard elicitation techniques. Because

*Common*KADS is a widely accepted and respected modelling methodology, I decided that this tool would be used to support both the knowledge elicitation and modelling processes where possible.

I believe this to be a novel approach to the development of knowledge-level models in the domain of design. However I believe the unique characteristics of concurrent design necessitate the use of this approach. The approach utilises a number of different standard elicitation techniques, which have been used to analyse problem-solving in a variety of different domains, including design.

5.10 From dialogue to model

My dialogues with expert in concurrent design used my initial top-down models of concurrent design behaviour to stimulate discussions and analysis of experts as to their problem-solving behaviour. These discussions and analysis resulted in the generation of transcripts from recording concurrent designers and design teams in action. These transcripts were then used to develop further graphical models of expertise, which illustrate the inferences and knowledge roles (in a *Common*KADS sense) utlilised by the design personnel. The iterative process implied by my novel approach resulted in the initial models more closely resembling the design experts' models with each iterative cycle. This transition is graphically illustrated in figure 5.1. This shows the necessary feedback loops from the generated graphical models to the experts, which is achieved in practice by the design personnel 'critiquing' the graphical models developed.

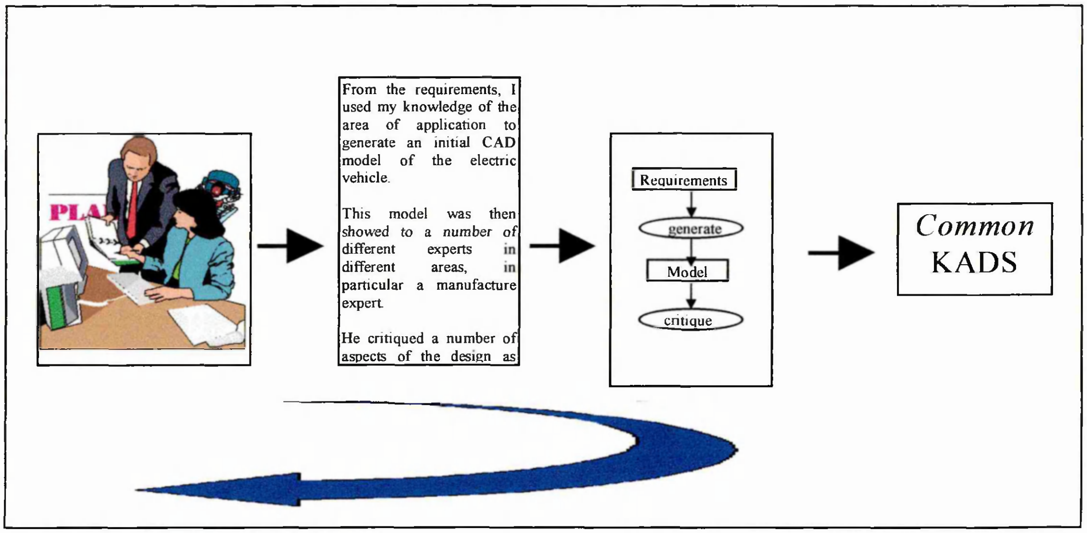

PLA
From the requirements, I used my knowledge of the area of application to generate an initial CAD model of the electric vehicle.
This model was then showed to a number of different experts in different areas, in particular a manufacture expert.
He critiqued a number of aspects of the design as
Requirements
generate
Model
critique
Common KADS

Figure 5.1: From design teams to formal models.

This illustrates how my knowledge elicitation method utilised an iterative process whereby transcripts of expert designers were used to generate *Common*KADS expertise task models which were then used as the basis for further discussions and knowledge elicitation with the designers.

5.11 Discussion of the methodology

Up to this point in this chapter I have outlined the thinking behind what I believe is a novel approach to the modelling and elicitation of the concurrent design process. However, it is also necessary to critically evaluate the thinking behind this approach and any steps that can be taken to accommodate short-comings in the approach.

A perceived problem with the approach was in the way that the already developed models may have affected an experts thinking during knowledge elicitation interviews and discussions. This is analogous to the problem found in techniques such as sorts where it is the knowledge engineers preconceptions about problem-solving in the domain that drive the elicitation and modelling process. This is particularly relevant to a process such as concurrent design.

Concurrent design is currently a relatively new and 'fashionable' way of approaching the design process. It was felt that designers would not wish to appear either ignorant of the process or 'behind the times' by not agreeing with the initial top-down models for concurrent design. Hence, wherever possible, these models were used as an initial means of stimulating discussions with design personnel but were not presented in any way as a process which the expert should be using.

As will be shown in later chapters, a number of additional practical problems also affected the way in which the methodology was used and how this acted to refine the methodology itself during the course of the research.

5.12 Chapter Summary

Knowledge elicitation is a crucial process in the development of knowledge-level models. This chapter has outlined a number of existing knowledge elicitation and modelling techniques, particularly those that have been used within the context of the design process.

Means of data collection for deriving models of expertise can be broadly classified into either standard or contrived techniques. The strengths and weaknesses of each technique are outlined. The limitations of the so-called contrived techniques in cases where the knowledge engineer does not have a deep understanding of the domain have been outlined. The automation of the knowledge elicitation process was also felt to be unrealistic for this study. A number of researchers have also used protocol analysis to analyse expert problem-solving behaviour, particularly in design. A number of these studies are described and key findings outlined.

Different approaches to expertise model development, including *Common*KADS expertise model development have been outlined. I have characterised these different approaches as being either bottom-up or top-down in nature. Each approach has its' own strengths and weaknesses, although other researchers have suggested that different approaches may be successfully combined in the development of particular *Common*KADS models.

Because there is no definitive methodology available for the analysis of design behaviour and it's incorporation into models of expertise, my chosen approach has been to develop what is essentially a novel methodology for analysing concurrent design. This utilises elements of both the top-down and bottom-up approaches.

My approach involves developing initial top-down models for concurrent design using analysis of the literature followed by refinement of the models via discussions with academics. These initial models were then used to drive knowledge elicitation sessions with design case studies and the developed models were refined and abstracted in a more data-driven, bottom-up manner. This approach makes use of some of the standard techniques for knowledge elicitation (interviews, thinking aloud, transcript analysis, ethnographic study and direct questioning). The effectiveness of each technique is assessed as the studies progress.

The next chapter describes the different case studies utilised in the research to analyse concurrent design activity and goes on to outline some of the findings resulting from the use of my novel approach to knowledge elicitation and modelling and how this has allowed me to refine this approach.

6 Testing and using the novel approach with different case studies

6.1 Introduction

In order to analyse design activity and design teams, I have undertaken a number of different case studies. A number of different companies, (who would consider themselves to be engaging in product design along concurrent engineering principles) were approached and agreed to cooperate with the research. The companies initially appeared happy for their designers to be interviewed and analysed as they worked. The designers in the case studies are all developing products which must be manufactured and assembled in some way. Certain constraints are common throughout the domain however some constraints were very prominent in one case study but are not considered in another.

This chapter then goes on to outline how I tested my approach to knowledge elicitation and modelling before using it with the different case studies. This was effectively a pilot study to improve and refine the methodology. Based on my experience of using the approach with the different case studies, a number of conclusions as to the validity and effectiveness of the approach are then discussed.

6.2 Case study 1

Company 1 manufactures components for use in the aerospace industry. Here functionality, reliability and safety of the product are key requisites. For each project, design teams are formed comprised of a designer (or number of designers for large projects). These designers spend most of their time working at CAD stations. Each project team also has a number of personnel with different expertise (particularly in materials and manufacturing) acting in a support role to the designer(s). This is achieved by having design teams situated in close proximity in an open-plan type office.

Specifically, case study 1 looked at the development of a new hydraulic locking mechanism for helicopter rotors. The key requirements for the product are reliability and function. Cost is also a consideration, but is does not have the same over-riding importance that it was seen to have in other case studies.

6.3 Case study 2

Company 2 manufactures bathroom fittings and furnishings. The functionality of products in this market is assumed to suffice and key requirements for market success are the product aesthetics, cost and time to market.

The specific project analysed involved designer 2 developing detail designs of bathroom items (toilets, sinks etc.). In this case, the appearance and form of the product are key. The conceptual design of products for this company is done by an outside design agency, while the company's designers concentrate on the later, detailing aspects of the design.

6.4 Case study 3

This involved a freelance designer (3) working on a number of projects. These were principally the development of casings for domestic boiler heaters and toilet cubicles. A consideration of the manufacturing and assembly methods employed by the companies he worked for influenced the designer as did time-scales and cost constraints.

6.5 Case study 4

Involved a team of designers (4 and 5) developing a radically new concept for a battery operated vehicle for use by the elderly and disabled. Designer 5 worked for the company who would produce the vehicles while designer 4 was independent. In this case, the major constraints on the designers were the manufacturing and assembly technologies available, the cost and also the time-scales that could be attributed to the design process. One of the key points of this case study was the use made of outside contractors to perform some of the more important manufacturing functions.

6.6 Case study 5

This involved a designer working in a company who manufacture plastic bottles for the bottled drinks and consumer items market. The designer was working in a consultancy role for the company on a short–term contract. The manufacturing techniques used to produce the finished designs are relatively static and well understood, so the main constraints on the designer are form, cost and time.

6.7 Commonality shown by the different case studies

Despite all the case studies being situated in the manufacturing engineering domain, they show an interesting diversity of products and approaches. Generally, these are relatively small-scale design studies, with one or two designers and a number of additional personnel involved. Because of this, each person involved may not just represent one life-cycle perspective. For instance, the designers tended to have a general understanding of a number of different life-cycle areas, which they used to mentally weigh up the pros and cons of different design decisions. In the case studies, designers either worked individually with occasional input from external sources or as part of small design teams with more frequent and formal interaction. In all cases, I will refer to them as 'design teams'. Even with only one designer 'designing', the designer may bring several different perspectives to bear on a design problem.

I will now discuss the use of my approach to knowledge elicitation and modelling with designers from the different case studies.

6.8 Testing my approach to knowledge elicitation and modelling

While I discuss the knowledge elicitation and modelling sessions that I undertook as part of this research, the designers and other personnel involved in the design process will be referred to as the 'expert' with myself in the role of 'knowledge engineer'.

Because the experts from the different case studies had limited time to dedicate to my research, I wished to test my approach before using it with them. I believed this would save me wasting the time of the different experts and would also endow the knowledge engineer with an element of skill in the use of the techniques, before their use in the field. I also believed that this would give the experts some confidence in the competence of the knowledge engineer and that this would make the elicitation sessions more productive.

Testing of my elicitation approach was done in two ways. The first was by conducting taped interview sessions with academics on some problem-solving task in which they possessed some element of expertise. These interviews were then transcribed and the resulting transcripts analysed. The other approach was to take existing media which described some problem-solving task and take transcriptions from these media (a typical example used was of a British Institute of Welding's' instruction video for different welding techniques and which applications they should be used for).

Initially it was hoped that the *Common*KADS workbench (see Toussaint et al [1994]) would provide support for the process of analysing expert transcripts. The tool provides a rudimentary text editor into which transcripts can be typed. Different elements of the transcripts can then be annotated (using highlighting icons) as being an element of a different *Common*KADS entity. The different entities in the transcripts can then be directly linked with entities in expertise task models (and also other models comprising the full *Common*KADS model set).

For example, the following excerpt from a transcript obtained from Case study 4 illustrates how this was done (it also illustrates how tasks tend to be represented by verbs, while knowledge roles are typically nouns):

...we made it (task) as a hand made model (role) and we took the tooling straight off the model we'd made...

Initially, this seemed an ideal way in which *Common*KADS expertise task models could be developed from the transcripts. That is, by linking entities from transcripts directly into graphical models and making use of the workbench's graphical editors to develop the models. In practice, however, this approach proved impractical, mainly due to two reasons.

The text editor incorporated in the workbench is very rudimentary and does not provide the level of functionality found in a typical word processing package. This made the input of the transcripts relatively difficult. A typical tape recording of an elicitation session with an expert was about an hour in length. The time taken to transcribe this would be of the order of five to ten times as long. Some of the features of modern word processors, such as spell checkers and word replace functions would have helped to make this task easier. The *Common*KADS workbench runs on a SUN Workstation running the UNIX operating system and this prevented the use of a PC based word processing package to produce the transcripts for later analysis on the workbench. This problem, while time consuming and frustrating, would not in itself have prevented me from using the workbench for the analysis of transcripts.

The main problem was that annotations made to transcripts could not be saved. Hence when a particular transcript had been entered in the workbench and suitable annotating icons added to it's content, these annotations could not be retrieved at a later date. Hence the chosen method was to produce the transcripts using a word processor.

The transcripts were then printed out and different colour highlighter pens were used to annotate elements of the transcript with the different *Common*KADS entities. While time consuming and laborious, this low-tech approach proved particularly effective in allowing the knowledge engineer to develop initial expertise task models which could then be re-visited after a suitable period of reflection. In this way, the developed models could be refined by subsequent analysis of the same transcript.

Testing of my approach in this way took several weeks of effort but did allow the knowledge engineer to refine and develop the approach and also to pinpoint some of the limitations outlined.

6.9 Findings from the use of my approach with the case studies

Having tested and refined my approach to knowledge elicitation and modelling with different test cases, I then began to use the technique in 'the field' with the case studies to analyse practicing designers and other personnel involved in the process of concurrent design.

The knowledge elicitation sessions were conducted, whenever possible, in the experts' place of work and involved real design scenarios the expert was either currently involved with or had previously worked on. This was in accordance with my desire to analyse concurrent design in as realistic a setting as possible. In general, my knowledge elicitation sessions with the experts proceeded as follows.

On the first knowledge elicitation sessions with the expert, the format of the *Common*KADS graphical models of expertise were described to the expert. The existing KADS and *Common*KADS models for design, together with my developed 'top-down' model (see the next chapter) were used to further illustrate the characteristics of the models and also to get the expert thinking and talking about their own design processes.

These meetings with the expert were always tape-recorded and most of the experts were quite happy for this to take place. Videotaping of meetings was also utilised in order to try and further deduce how design, critiquing and negotiation occur in a group setting. However, because of issues regarding confidentiality this technique could not be used as often as desired.

After the first elicitation session with each expert, the tape-recorded interviews were transcribed. From the transcripts, the initial models of the particular experts' problem-solving expertise were developed. This was an exceedingly time consuming process and involved a number of analysis sessions of a single transcript to develop and refine the models of expertise.

6.10 Using graphical models to communicate with designers.

The developed *Common*KADS graphical models represent knowledge roles, inferences and tasks, which depict how the expert performs some problem-solving task.

At follow up interviews, the experts were then asked if the models were an accurate representation of their problem-solving behaviour. The designer would then inspect the models to see if they concurred with what they believe is their problem-solving behaviour.

The most common misconception on the part of the designers was that the models were in the form of flowcharts and that some kind of time-dependency is implied by the models. This is not the case. The models actually show what inferences and sub-tasks the expert uses for a particular task and which knowledge roles are inputs and outputs to the different sub-tasks and inferences. Generally, the experts were quickly able to come to terms with the models and evaluate them. During these sessions, the experts would pinpoint where the knowledge engineer had misunderstood elements of earlier sessions. Typically, an expert would notice a missing knowledge role from an inference or an inference having the wrong knowledge role as input or output.

6.11 An example of an expert refining a model

In this section I will look at an excerpt from case study 3 which involved a follow up elicitation session with designer 3. Initial models for designer 3's problem-solving behaviour had already been developed at an earlier session with the designer. These consisted of a high-level model of the designer's overall design behaviour and a number of different models which were expansions of different sub-tasks comprising the overall model. The initial model, shown in Figure 6.1, shows designer 3 altering a design on the basis of what are perceived to be problem areas in the original design proposal.

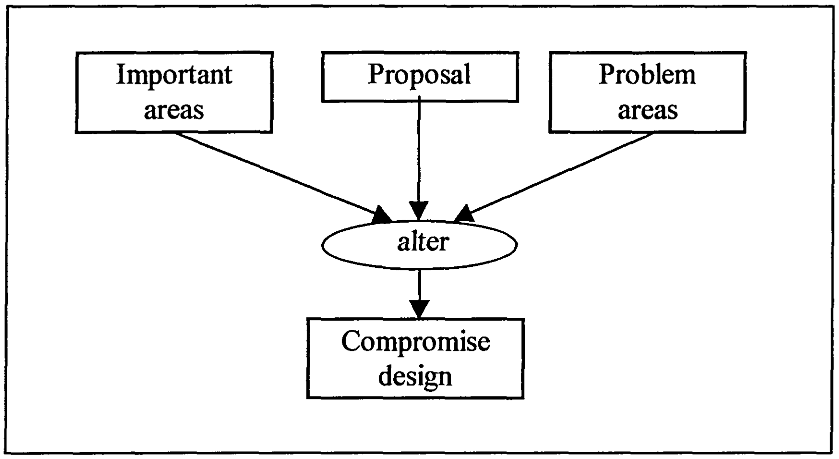


Figure 6.1: Designer 3's initial model for the 'alter' process.

This model shows the designer taking 'Important areas', a 'Proposal' and 'Problem areas' as input knowledge roles to the 'alter' task, which results in a 'Compromise design'.

When my initial model for the 'alter' sub-task, shown in figure 6.1, was presented to designer 3, the following dialogue ensued (knowledge engineer's comments in italics):

"This model here is supposed to represent that at a greater level of detail. So they're presented with a proposal which breaks down into a number of different areas. And so this sub-task here for instance would very well break down into something at that level of detail as well".

Brief pause. "So why does that go round the outside of there?"

"Basically this inference or subtask here is 'alter' which alters that initial proposal with input from 'problem areas' and 'important areas' to come up with a compromise design".

"Right. Where you use the word compromise , well it depends what you mean by that. When I read the word compromise I think of something that is sort of - it's mediocre in a way - now the designer's trick I think is to arrive at that but that it is not in any way giving the appearance of being compromised. Because if it were truly a compromise, it might not be worth doing. The result might be so awful that there would be no point proceeding, OK it's been sorted out from all sorts of points of view, timescales, production and what have you but the product is so bloody awful that there's no point in proceeding, so I think I'd somehow like to see that word 'compromise' in inverted commas or something. It's a modified design, which may or may not be a compromise, but it's essentially modified. I might be able to modify it in such a way - it might even be better - it might not be worse".

"So modify implies just that it's different".

"Yes, whereas compromise somehow suggests that it's worse. It often is worse, you're right, but I wouldn't like to think that is the way it has to be".

Hence, this briefly illustrates how one sub-task was modified by subsequent knowledge elicitation sessions with the expert to that shown in figure 6.2.

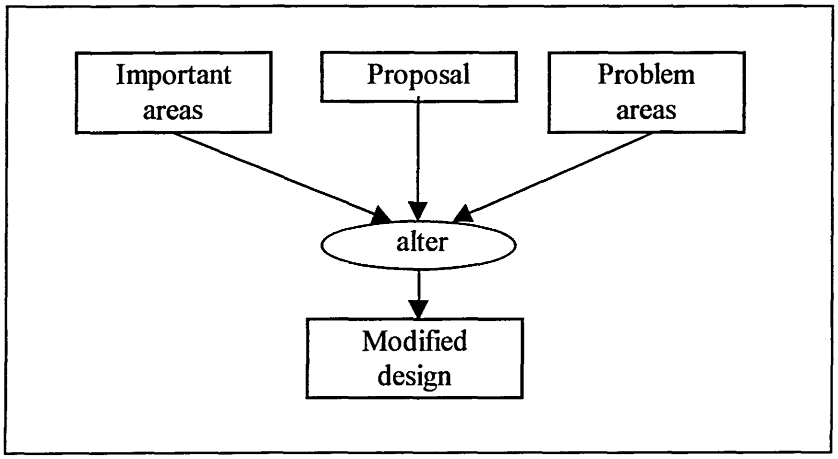


Figure 6.2: A refined model for the 'alter' sub-task.

This model was based on a follow up interview with the designer where the role 'Compromise design' (which suggested the design was somehow lacking) was replaced by 'Modified design'.

6.12 Problems with language

Initially, models were generated using fairly succinct and brief language to describe knowledge roles and inferences, the terminology used being derived from the interview transcripts. This approach worked well enough when the time lag between initial and follow up interviews was small (i.e. a matter of weeks at the most). The designers were able to recognise their own terminology and relate to the models fairly easily.

However, problems became apparent when a designer had to attempt to relate to another designers' models. In case study 4, two designers had worked on the development of the same product. At an interview with the second designer, the designer started to 'dry up' in conversation. On order to further encourage his thoughts, the models developed for the first designer were shown to him as a prompter. While he could generally follow the scope of the models, some of the terminology was unfamiliar and meaningless to him. As a result he was not able to engage with the models as well as he might have been able to.

While the use of concise and succinct language can be seen as desirable for use in generic models, particular models used for discussion with particular experts should not be so concise with their use of language. As a result, the models developed for the second designer in case study 4 also have an accompanying key to describe in more detail the numbered roles and inferences represented by the task models. I believe this to be a good compromise. The models are not so cluttered with text that the structure of the models becomes lost but the expert can still consult the key if they are unsure about a particular part of the model.

6.13 Advantages of my approach

Concurrent design as a process presents a subtly different set of problems for the knowledge engineer when attempting to analyse the problem-solving behaviour of human experts compared to more co-operative modes of design.

For instance, one designer had realised retrospectively that one of the sub–contractors he was working with had misled him about the technical feasibility of an aspect of his design. The sub–contractor had attempted to persuade the designer that it would be necessary to remove some of the more complicated curves from a moulding. This was not because the curves would be impossible to produce (although they would be difficult), the sub–contractor had realised that to produce the curves would require him to work weekends while a simpler design could be more easily produced in a shorter time. Such a realisation would not have become evident from a transcription of the discourse that occurred between the designer and the sub–contractor at the time of the meeting.

This also illustrates how concurrent design is different from more co-operative modes of design in that the goals and motivations of the different perspectives involved in the concurrent design process can be radically different. The 'optimality' of the design is key to the designer while the different perspectives will work to their own agendas. These differences in the goals of the participants were much more marked where outside contractors were being used to produce different parts of a design. However, this also seemed to occur, although to a lesser extent, with downstream perspectives from within the same organisation.

Another major advantage of this approach is believed to be the way in which the expert is kept within the model development loop. Because the expert plays a major role in the development of the expertise task models, they have a certain ownership and stake in the correctness of these models.

A number of examples from the literature have been analysed in earlier chapters which show where hypotheses and models of design problem-solving behaviour have been developed from transcript analysis of expert designers performing some task. However, no reference was found in these studies of these models and hypotheses subsequently being presented back to the relevant experts for comment or critiquing. I believe that it is important to keep the expert within the knowledge acquisition 'loop' as much as possible. I believe that by using a coding technique to analyse a transcript and then develop models of expertise straight from this coding scheme, the expert is distanced from the modelling process and may have difficulty relating to any subsequent tool based on the developed model of expertise.

My developed models for concurrent design, in line with *Common*KADS models developed by other researchers, are intended to support knowledge engineers and software developers who are implementing computer-based support for a problem-solving process. In this case, concurrent design.

Typically such computer-based tools will then be used to support existing experts (or even the same expert) in their tasks. Where the expert has had a role in the development of the models underpinning such computer support, I feel the expert is more likely to accept the output from such a computer support system.

6.14 Disadvantages of my approach

A number of disadvantages also became apparent with the methodology used. In the retrospective case studies, it was felt that designers would attempt to rationalise their activities and the picture being presented was how they felt the design process should have progressed, not how it actually happened at the time. However, I believe that this limitation will be evident in any study of design (whether retrospective or real–time) in that when a person is under scrutiny they will tend to be more conscious of their activities than if they were working in a normal, unobserved manner.

Another disadvantage was in the amount of time taken to analyse transcripts from design experts. A transcript consisting of approximately an hour's discussion would take up to ten hours to transcribe. Because of the domain-specific vocabulary being used, it was necessary for the knowledge engineer to transcribe recordings of meetings. Once the transcripts had been produced, it was then a time-consuming task to either develop or refine models of expertise, based on the transcripts. Typically model development would consist of a day spent producing initial models or refinements, followed by a further day or two revisiting the transcripts to make any necessary changes. As a result, an hour's interview time could take at least a week to produce models of expertise.

6.15 Chapter Summary

In this chapter I have presented details of the different 'real-life' case studies I have used to derive models for concurrent design. These case studies involved a number of designers and other personnel involve in the concurrent design process.

I have then outlined how my approach to knowledge elicitation and modelling was tested and refined by utilising a number of different test scenarios. The approach was subsequently used with personnel from the different case studies to develop and then refine models of the different experts' problem-solving behaviour.

While using my approach to elicitation and modelling with the different experts, a number of refinements were made to the approach itself in order to make it more effective. I have also outlined a number of strengths and weaknesses of the approach and why I think this approach is particularly suitable for analysing expert problem-solving behaviour in a domain such as concurrent design, or indeed any other domain where the developed models of expertise are intended to act as the basis for subsequent computer-based support for the process.

In the next chapter, I go on to discuss the development of an initial top-down model for concurrent design based on existing models for design and the available literature.

7 The development of my initial 'top-down' model for concurrent design

7.1 Introduction

This chapter discusses the development of a top–down, knowledge-level model for concurrent design and some of it's associated sub-tasks. This was based on analysis of theory in the literature of how concurrent design occurs. Development of the model was heavily influenced by different texts and published research on concurrent design, artificial intelligence and design thinking. In addition, extensive discussions with academics with extensive expertise in these disciplines acted to refine the models.

I begin by outlining how I see concurrent design as a series of sub-tasks of propose, critique and negotiate. I then go on to discuss the development of top-down models for these sub-tasks and their subsequent incorporation into an overall task model for the concurrent design process.

7.2 How does the process of concurrent design occur?

My thinking behind the development of a top-down model for concurrent design began by analysing respected design texts in the mechanical engineering field and assessing to what degree the prescriptive models for design supported concurrent design. The consideration of downstream life-cycle constraints is not a central tenet of the models for design presented in Pahl and Beitz [1984] or Hubka [1982]. However, Pugh [1984] does imply that these constraints should act as an important input to the process of design.

This early research also indicated how the process of critiquing could be used as a mechanism whereby these downstream considerations could be used to influence the process of design. The work of Fischer and colleagues at the University of Colorado (see Fischer [1990], Fischer and Mastaglio [1991], Fischer et al [1991a], Fischer et al [1991b], Fischer et al [1993a] and Fischer et al [1993b]) were particularly influential in shaping my thinking on how the process of critiquing might occur.

Important research in the field of AI has also indicated the importance of the critiquing process in design. In particular, the work of Chandrasekaran [1990] had a great effect on my thinking. Chandrasekaran suggests that designers utilise a task-based process of propose-critique-modify in order to 'solve' the overall design task. Maher [1990] also expounds this view.

Chandrasekaran goes on to suggest that the modify task may, in some situations, be viewed as a new design problem to be solved. This is a view which I take as I believe the 'modify' task does not show significantly different features from the 'propose' task in order to consider it as a separate task in it's own right. This is supported by Top and Akkermans [1994] who state that *"we do not consider revision, redesign or reformulation as an independent subtask since the actions involved in it are represented by the construction task itself"*.

Hence, I began to see the concurrent design task as a series of sub-tasks of propose (where a solution or partial solution to the design problem is generated) followed by a critique (where downstream perspectives could criticise the generated solution or partial solution from their own perspective).

However, if a number of different perspectives are involved in critiquing a design solution or partial solution, the very nature of the perspectives will dictate that differing, sometimes conflicting, critiques may be offered. In fact the very nature of the concurrent design process I am considering implies that different personnel involved in the process may well have differing viewpoints regarding certain aspects of a design. I consider this type of design as being similar to the 'contested collaboration' process outlined by Sonnenwald [1996]. However the types of 'role' that I consider the different critics play are based on the type of specialist downstream life-cycle knowledge they bring to bear on the design problem. This is rather different to the more organisationally-oriented 'roles' considered by Sonnenwald who considers roles including 'sponsor', 'inter-organisational star' and 'inter-group star'. Sonnenwald goes on to suggest that this contested type of design can have negative consequences on design outcomes. Baker [1993] also tends towards this view of negotiation - *"negotiations are often initially characterised by conflicting attitudes"*.

Hence there needs to be some arbitration or negotiation mechanism whereby the differing critiques may be reconciled in some way. I considered the 'negotiation' task as being where these differing viewpoints are reconciled. Klein and Lu [1989] use the term 'conflict resolution' to describe this process.

I therefore began to consider the concurrent design process as being composed of three crucial sub-tasks of propose, critique and negotiate. Based on these initial observations, an initial task decomposition for concurrent design is outlined in figure 7.1.

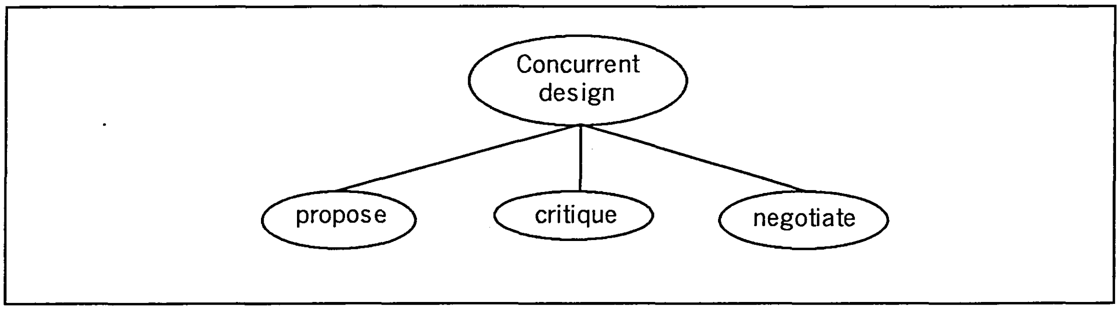


Figure 7.1: Task decomposition for concurrent design.

This shows the sub-tasks that the concurrent design task was initially believed to be composed of. However, the task decomposition does not indicate the possible decompositions of the sub-tasks themselves or important knowledge roles acting as input and output roles to the sub-tasks.

This task decomposition is represented in the graphical format used by *Common*KADS for expertise task type knowledge as shown in Figure 7.2.

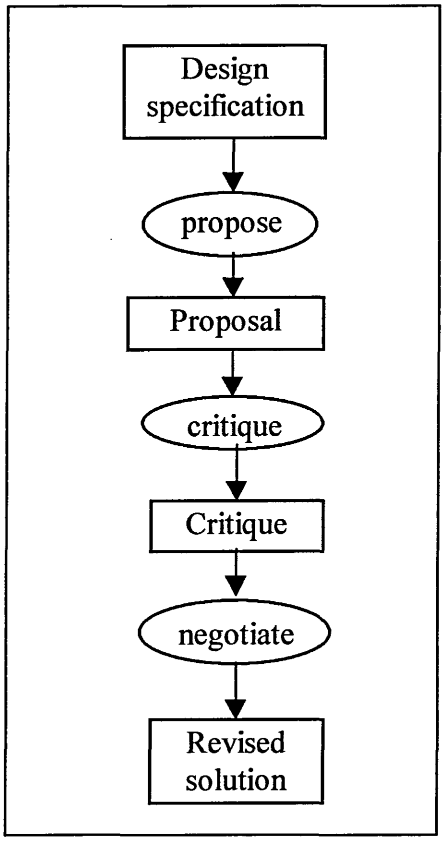


Figure 7.2: My initial expertise task model for propose – critique – negotiate.

This task model shows the knowledge roles that I initially believed act as important input and output roles to the propose, critique and negotiate sub-tasks implied by concurrent design.

The task decomposition outlined in Figures 7.1 and 7.2 outline the critical sub-tasks that I assumed played a critical role in the concurrent design process. However, it does not consider any additional knowledge roles that may be required as inputs to the different sub-tasks and also the possible expansions of the sub-tasks themselves. I therefore began to look in more detail at the different knowledge roles and methods that might be applied to the sub-tasks for this view of design.

7.3 Concurrent design as 'propose–critique–negotiate'

I began to consider the different knowledge roles acting as inputs to the sub-tasks comprising the concurrent design process. I used a number of different design scenarios to help develop my ideas and outline how the different tasks and knowledge roles would interact. As an example, consider this scenario which has a designer developing a mechanical winch for use in an outdoor, marine environment. The designer was assumed to have access to expertise from a materials and a cost perspective.

For the propose task, the design specification clearly plays an important input role to the generation of a design proposal. A fully developed specification will give the designer a key insight into both the required functionality of a design together with the context within which is to be developed and used. It is debatable as to what exactly constitutes a complete design specification and a number of different definitions and viewpoints exist in the literature. My initial view was that a design specification consists of a set of functions which the completed design would be required to satisfy, together with a number of constraints dictating to an extent how the design could be developed and implemented. For my simple example, a typical function would be 'the winch must be able to lift a load of one tonne'. A constraint on the other hand would be that 'the design must be assembled by non-skilled personnel'.

Another important knowledge role that could be input to this stage would be previous designs a designer may have been involved or was familiar with. For instance, a designer who had previous knowledge of winch designs would very likely make use of this knowledge when developing a new design.

It was also assumed that the design proposal task would not be used to generate a complete design in one single stage. Rather, the design would evolve through a series of stages of propose, critique and negotiate. In this way, a design proposal could generate a solution for a small part of the overall winch design problem or possibly make some other decision, which advanced the design in some way.

Hence my design scenario begins with the designer proposing the use of a mild steel material for the casing of the winch. The critiquing task is then used to outline areas in which a design is deficient from a particular perspective and to outline any constraints specific to a particular perspective that the given design proposal had neglected to accommodate. In the case of my scenario, the two downstream perspectives, materials and cost, would offer radically different critiques of this particular design proposal.

The cost expert's reaction would be of the form *"Yes, that looks Ok. Mild steel. is a relatively cheap material"*. However, the material's experts contribution would more likely be of the form *"No, that would not be a good material to use, you have neglected to consider the effect of salt water on the component. You should consider using another material such as a plastic or aluminium for the casing"*.

Hence I began to consider a number of different knowledge roles as being outputs from the critiquing task. These some form of assessment, any possible additional constraints the designer has neglected to consider together with a possible counter proposal.

Clearly the critiques offered by the different perspectives are conflicting to a degree. The materials expert's advice to use an aluminium material for the component casing will conflict with the cost expert's feeling that the cost of the component should be minimised. Hence there would then be a negotiation process to determine whether this alternative material proposal would be incorporated within the design together with a decision as to whether the constraint of the salt water environment should be allowed to act an important constraint on any future design decisions.

This iterative process would then continue, advancing the evolving design at each stage. Hence, in order to formalise my view of design as an expertise task model, it was hypothesised that a concurrent designer, or design team, work in the following way:

- With input from previous designs and the product design specification, the designer proposes a solution (or a partial solution) for some part of the overall design problem (This assumes the design process is split up into manageable 'chunks' in some way). This will modify the current design model in some way to give the 'advanced' design model.

- The proposal is then critiqued from a number of different perspectives. The output from critiquing is assumed to consist of further constraints which were not immediately obvious or documented in the design specification, an indication of whether or not the proposal is acceptable and the critique itself. Missed functions or previously unrealised external constraints will be pinpointed by the customer or market needs. However, additional internal constraints, which typify the concurrent approach, will be pinpointed by downstream life-cycle perspectives.

- There is then a phase of negotiation to determine which additional constraints can reformulate the design specification and a new series of design proposals are then made.

The granularity at which this method is used is critical. If the 'propose' method is used to generate a complete design before critiquing occurs, then the method is not concurrent in the generally accepted sense of the term. If the propose method is used to advance the design only a small stage forward, then this nears the concurrent ideal. Based on this, the knowledge roles acting as input and output to the 'propose' task are outlined in my top-down model for the design proposal task, shown in Figure 7.3.

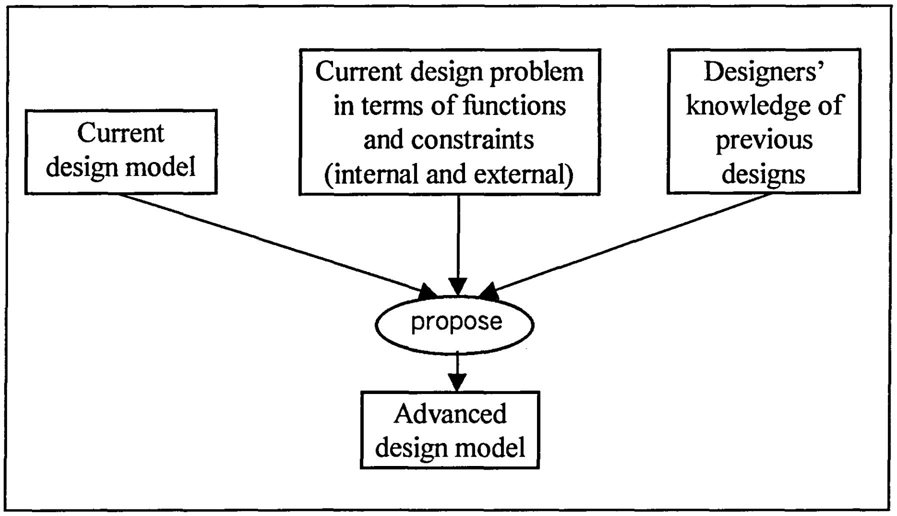

Current design model
Current design problem in terms of functions and constraints (internal and external)
Designers' knowledge of previous designs
propose
Advanced design model

Figure 7.3: The knowledge roles acting as input and output to my initial model for the 'propose' sub-task within concurrent design.

Different methods that have been utilised for the propose task (case-based reasoning etc) are discussed in the text.

Different task structures and problem-solving methods for the propose sub-task within this model have been outlined earlier in this paper (see Chandrasekaran [1990], Maher [1990]) and it was discussed how some of these methods have been formalised as *Common*KADS task models by Bernaras and Van de Welde [1994].

A survey of the literature on critiquing revealed a number of possible different ways in which the critiquing sub-task may be modelled.

7.4 Models for the critiquing sub-task

Possible models for the critiquing sub-task has been developed in a similar fashion to the approach used for the initial model for concurrent design.

A number of researchers have proposed task-like models for the critiquing process. Fruchter et al [1993] *"Critiquing entails analysis and evaluation of the design. In the analysis stage, the performance of the design is predicted. In the evaluation step, the derived performance is compared against the requirements"*. Hence an expertise task model for this critiquing mode is given in Figure 7.4.

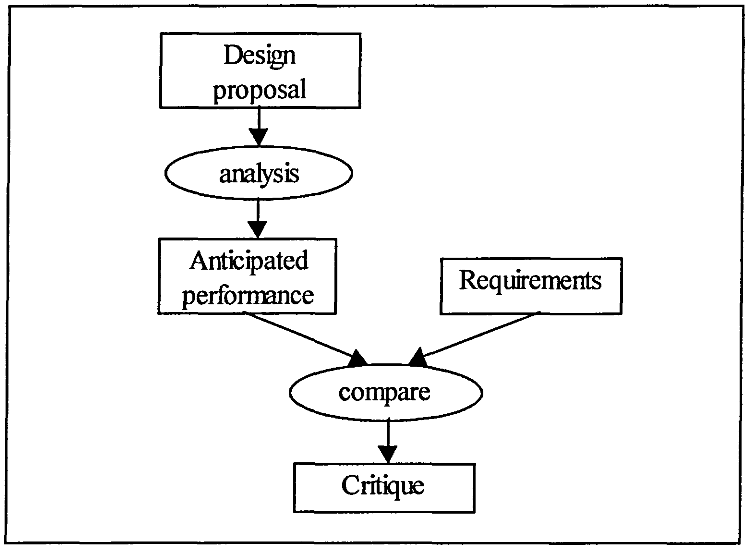


Figure 7.4: A critiquing model - derived from Fruchter et al [1993].

This model is based on an analysis task, where the anticipated performance of a design proposal is estimated. This is then compared with the performance outlined in the requirements to generate a critique.

Fischer et al [1991b] compare differential and analytic critiquing. In differential critiquing, the critic generates it's own complete solution, compares it with the user's solution and points out the differences. This is modelled in figure 7.5. A problem with this type of critiquing is that if the method the critic uses to generate a solution is different to the designer's, there is no common ground for communication as to how the designer's solution can be improved.

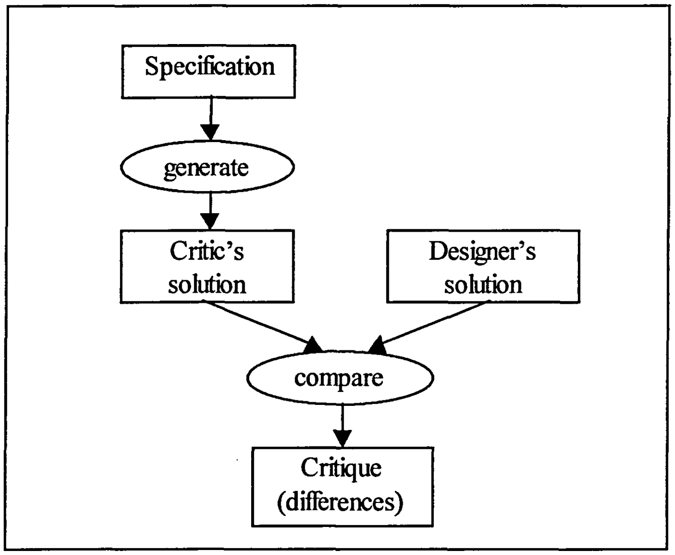


Figure 7.5 A Critiquing model for the differential mode from Fischer et al [1991b].

This critiquing mode assumes a critic generates a solution independently of a designer. The critic's and the designer's solutions are then compared and any differences comprise a critique.

In the analytical case (see figure 7.6) the critic identifies sub optimal features in the users design. A number of different ways this has been computationally achieved include pattern matching, finite state machines, augmented transition networks and expectation based parsers. However, it is felt to be unlikely that these are the methods that designers actually use to achieve these tasks.

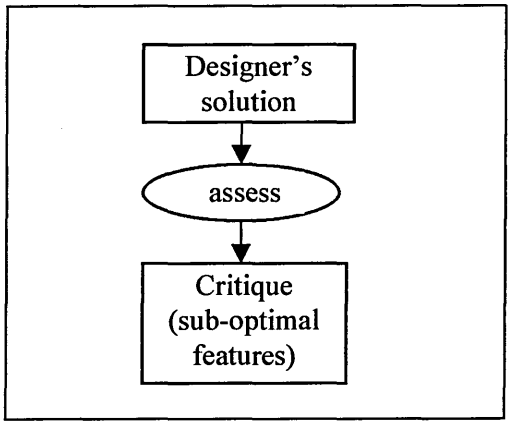


Figure 7.6 A critiquing model for the analytical mode from Fischer et al [1991b].

This mode of critiquing has the critic assess a designer's solution to outline any sub-optimal features. This is a fairly high-level model for the critiquing task, in that the 'assess' task does not give any indication of how any sub-optimal feature sin the designer's solution are pinpointed by the critic.

The critiquing mode used by the critic (passive or active etc.) does not affect the expertise task structure (i.e. the knowledge roles and inferences implied by the subtask). Rather this is a sequencing and control issue.

7.5 My top-down task model for critiquing

This clearly showed there are a number of different ways in which the critiquing task might occur. My 'top-down' model for critiquing is derived from considering an expert critiquing a design from their own perspective. The knowledge roles input to the critiquing task are the design model (which will represent the current state of the design), the specification (which the design is based on) and the relevant knowledge of the individual critics. In terms of the types of critic identified in Fischer et al [1993b], the type of critic envisaged is a combination of the three types; generic, specific and interpretative.

As for my previous example, consider a manufacturing expert assessing a design which features two components arc welded together. The critique could consist of - *"No, this is not a suitable process to use"* (the assessment). *"An alternative might be laser welding because there will be less distortion"* (proposal), *"however the facilities we have for laser welding are very limited"* (additional constraints).

Hence a critique, which can be seen as the output from the critiquing task, consists of a number of distinct knowledge roles. These are an assessment of whether the design is acceptable from this perspective, a more acceptable proposal and an indication of any additional constraints that the designer has neglected to consider.

Chandrasekaran [1990] suggests critiquing is a generalisation of the diagnosis problem type. Diagnosis implies that there is some fault that needs to be diagnosed. Diagnosis then typically proceeds by outlining a hypothesis of what might be responsible for the fault and attempting to verify this diagnosis. However, when a downstream 'expert' analyses a solution (or more usually a partial solution) to a design problem, I believe they do not necessarily start off with the assumption that there is some fault that needs remedying.

My initial thoughts on what the sub–tasks used to critique a design might be suggested:

- Analyse the existing proposal and assess its suitability. This is assumed to be of a qualitative nature.

- Based on the assessment, generate a more suitable proposal or sub proposal if required and outline any additional constraints.

Based on these observations, my top-down model for critiquing within a concurrent design context is proposed in Figure 7.7.

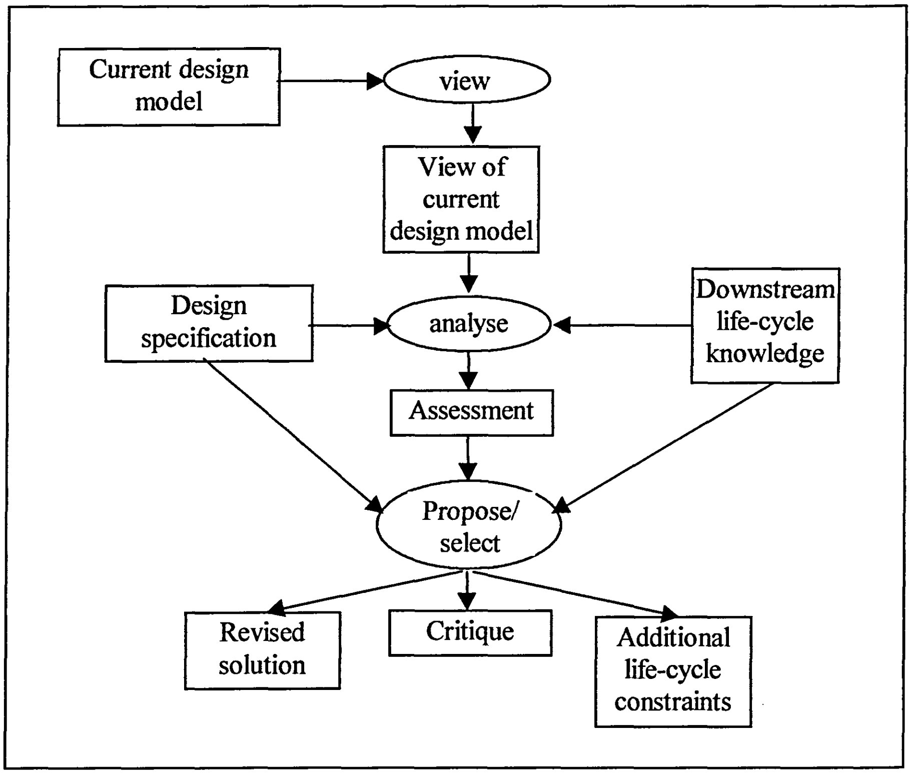


Figure 7.7: My initial expertise task model for critiquing.

This model assumes critiquing proceeds by a critic forming their own 'view' of the designers solution. The critics knowledge of a particular life-cycle perspective and input from the design specification is then used to assess the solution. Based on this assessment, a critique, a possible alternative solution and a critique of the designer's solution are generated.

This can be seen as an application of a problem-solving method that I shall term assess and revise. This method has a number of features of the critiquing modes discussed from the literature. The assess task, where sub-optimal features of a design are pin-pointed, is clearly influenced by the analytical mode of critiquing while the generation of a revised solution is influenced by the differential mode (both from Fischer et al [1991b]). The critiquing mode described in Fruchter et al [1993] is felt to more likely to be used to outline any functional deficiencies in a design, rather than to pinpoint life-cycle constraints a designer has neglected to consider.

This is clearly an initial model. A downstream expert critiquing an evolving design might utilise a markedly different problem-solving method to achieve this. However, the model is an adequate starting point for discussion with experts as to how critiquing occurs.

7.6 Negotiation

My understanding of the negotiation process was less well formed at this stage of the research although I had considered a number of different possible models for negotiation, based on the available literature.

Baker [1993] considers knowledge roles that are important to the negotiation process as being the goals and attitudes of agents before and after the negotiation process.

In the DFI system described in Werkman [1991] different software agents can negotiate in a co-operative or conflicting mode via the use of a blackboard system. The main negotiation strategy utilised by the agents when some conflict is detected is the generation of alternative proposals. A controlling software agent, the arbitrator, has the power to make a final decision if this negotiation strategy fails to resolve deadlock. Again, the goals of the different agents are taken to be the maximisation of their own objective function at each cycle of the negotiation process and this goal clearly acts as an important input knowledge role to negotiation. This form of negotiation is illustrated in figure 7.8.

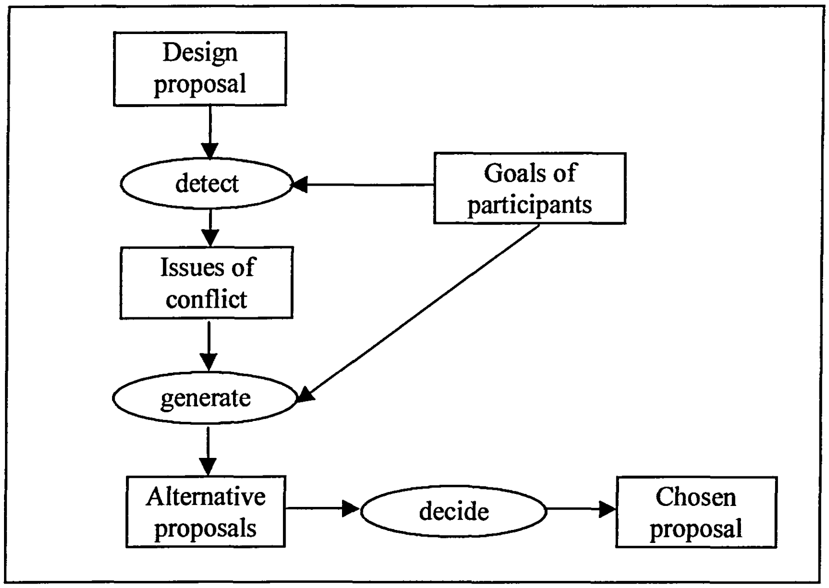


Figure 7.8: My initial model for negotiation based on the generation of alternative proposals as a negotiating strategy.

An important role input to the process is the differing goals of the participating 'agents'.

Klein and Lu [1989] analyse conflict resolution in a more cooperative mode of design than I have analysed. However their findings are still very relevant to my research. They outline a number of different conflict resolution strategies that were observed in their research. These include abandoning goals, alternate plans, adding detailing and partial goal fulfillment. They further suggest that the application of these strategies can themselves set up further conflicts requiring resolution. They also make the observation that their list is not meant to be definitive in any way.

7.7 My developed top-down model for concurrent design

Hence the development of my top-down model for concurrent design has progressed by initially analysing the critical sub-tasks implied by concurrent design: propose, critique and negotiate. I have then considered important knowledge roles acting as inputs and outputs to the different sub-tasks and also possible expansions of the sub-tasks, in particular, the critiquing sub-task.

Based on these mainly theoretical considerations, I propose an initial, top-down task structure for concurrent design having the form shown in Figure 7.9 This has been derived by combining the different models developed for the propose, critique and negotiate tasks which were believed to be central to the concurrent design process. I have considered negotiation as being where different life-cycle perspectives can attempt to resolve differences and reach some agreement as to what constitutes an acceptable design.

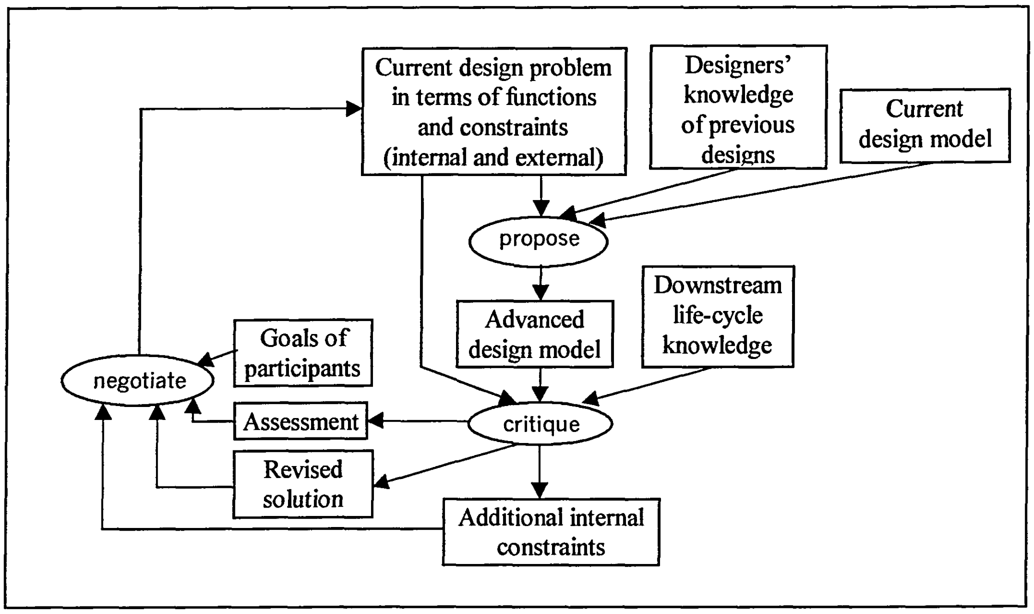

Current design problem in terms of functions and constraints (internal and external)
Designers' knowledge of previous designs
Current design model
propose
Advanced design model
Downstream life-cycle knowledge
Goals of participants
negotiate
Assessment
critique
Revised solution
Additional internal constraints

Figure 7.9: An initial 'top-down' model for concurrent design.

This model is based on initial models developed for the propose, critique and negotiation sub-tasks.

I have considered the central tenet of concurrent design as being the role that downstream life-cycle perspectives play in the process. However, other perspectives, such as the 'voice of the customer' or some role representing market needs are also likely to be an important input role to a number of the sub-tasks, particularly the negotiation process.

The expertise task models outlined in this chapter have been developed mainly through thinking about design and the associated sub-tasks and theoretical considerations about how the design process occurs. My methodology now requires that these top-down models for design are validated and refined by analysing design behaviour in a more bottom-up fashion using the developed top-down models as templates for discussions with designers and design teams.

7.8 Chapter summary

This chapter has presented a number of 'top-down' models for concurrent design and its' associated sub-tasks. These were developed by thinking about how concurrent design occurs and were heavily influenced by researchers in a number of different fields. The models were refined and further developed by extensive discussions with academics involved in the fields of engineering design and AI.

However, while these models gave a general overview of the different tasks that concurrent design imply, because their development was influenced more by thinking about how concurrent design occurs, I believed the models lacked the detail and subtlety that designers actually use when working in a concurrent manner. Hence, in accordance with my previously stated methodology, the next stage was to use my different case studies to analyse designers and design teams working in an industrial setting. The 'top-down' models which had been developed were used to stimulate initial discussions with the different designers and personnel comprising the case studies.

8 Design models - the reality.

8.1 Introduction

This chapter looks at the results of attempting to analyse the working methods and design processes of designers and design teams involved with two different companies (case studies 1 and 2).

These models were developed using my novel approach to elicitation and modelling outlined in a previous chapter. The models presented are the final models that resulted from a number of different elicitation sessions with the design personnel concerned. The models were developed and refined until both the knowledge engineer and the design personnel concerned believed them to be at a suitable level of correctness and refinement.

I then go on to analyse the findings from these two case studies and how this influenced the way in which I approached the remaining case studies in my research.

The expertise task models presented are task specific in that they use the vocabularies of the designers and downstream perspectives analysed. This is in line with current *Common*KADS thinking (see Wielinga et al 1994a) which advocates the use of teleological inference names (i.e. the inference name relates to the purpose it serves). This means that inferences are task specific. However this has the advantage of allowing an expert to comment on developed inference and task models and thus help in their validation and refinement. In addition, it must be emphasised that the initial models developed for concurrent design were used to stimulate discussions with designers and design teams but were not held to be how the process should occur when they were initially presented to designers.

8.2 Case study 1

The way in which the design teams operate concurrently was explained by the Chief design engineer on introduction to the company. The company was reluctant to discuss current projects and would not allow video recorders to be used within their premises under any circumstances. However, once the Chief designer was assured of the confidentiality of the research, the veto on current projects was lifted.

A designer (Designer 1) working at a CAD station was working on a current project. The design processes used to generate the model were discussed. Designer 1 was particularly informative about how different perspectives affecting the design could be considered. Because of the open plan type office, the designer could literally call over an 'expert' in some life–cycle perspective when he needed clarification of the implications of a particular design decision. The interviews were recorded and transcripts from the interviews were used to generate *Common*KADS graphical task models.

8.2.1 Company 1 design models

Initially, a specification is generated by a customer. This will be fairly specifically defined in terms of function, shape, size/weight, performance, materials and processes (this is likely to be an exclusion list - i.e. you will not use this material / process) and reliability. Multi-functional team then generate concepts to suit this. This overall design model is illustrated in Figure 8.1.

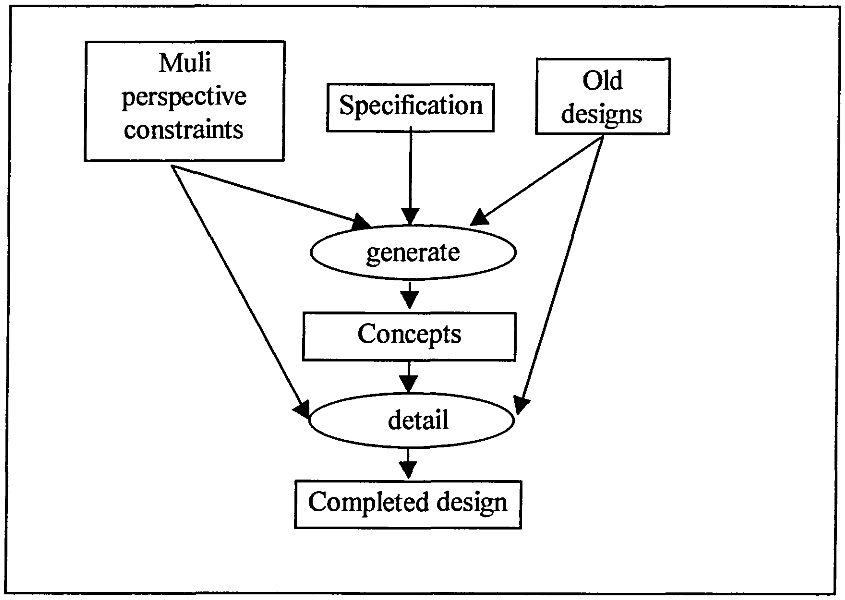


Figure 8.1: Company 1 overall design model.

8.2.2 Company 1 and the 'generate' sub-task

Concepts are focused very quickly. I.e. a concept will naturally evolve, it is unusual for two or more different concepts to be simultaneously considered. This concept is initially developed according to the three most important areas of design, stress and performance parameters. It is also reviewed by development, manufacturing (Planning, NC and jig / tool functions), estimating, project management, reliability, customer support and purchasing. The company 1 philosophy of minimum cost and robustness also acts as a feedback loop throughout this process. After a concept is selected, it is subjected to a preliminary design review. This process is shown in figure 8.2.

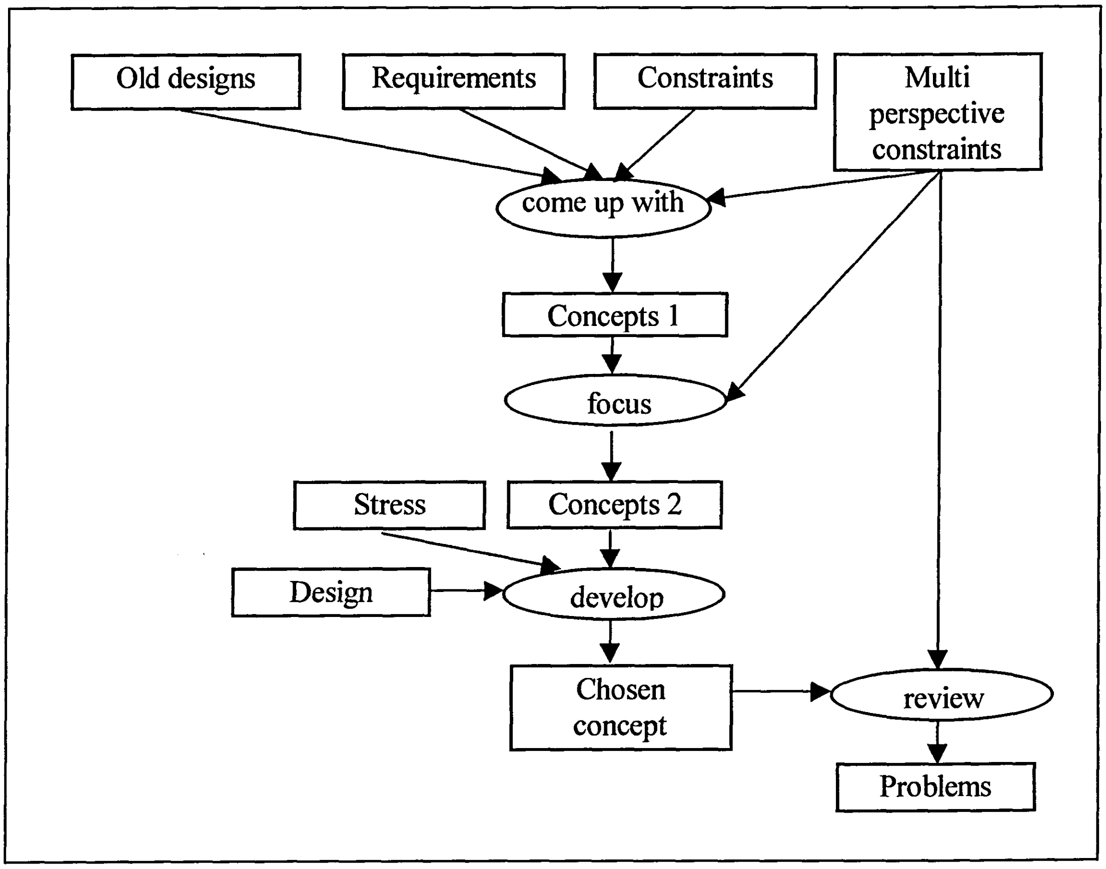


Figure 8.2: The Company 1 'generate' sub-task.

8.2.3 Company 1 and the 'detail' design sub-task

The next stage is 'Product data package' - i.e. detail design and drawings. A number of operations proceed in parallel - design (detail organisation), manufacture (process planning), N.C. and assembly instructions. The final design is then reviewed at the complete design review and the design passes on to operations. This is shown in figure 8.3.

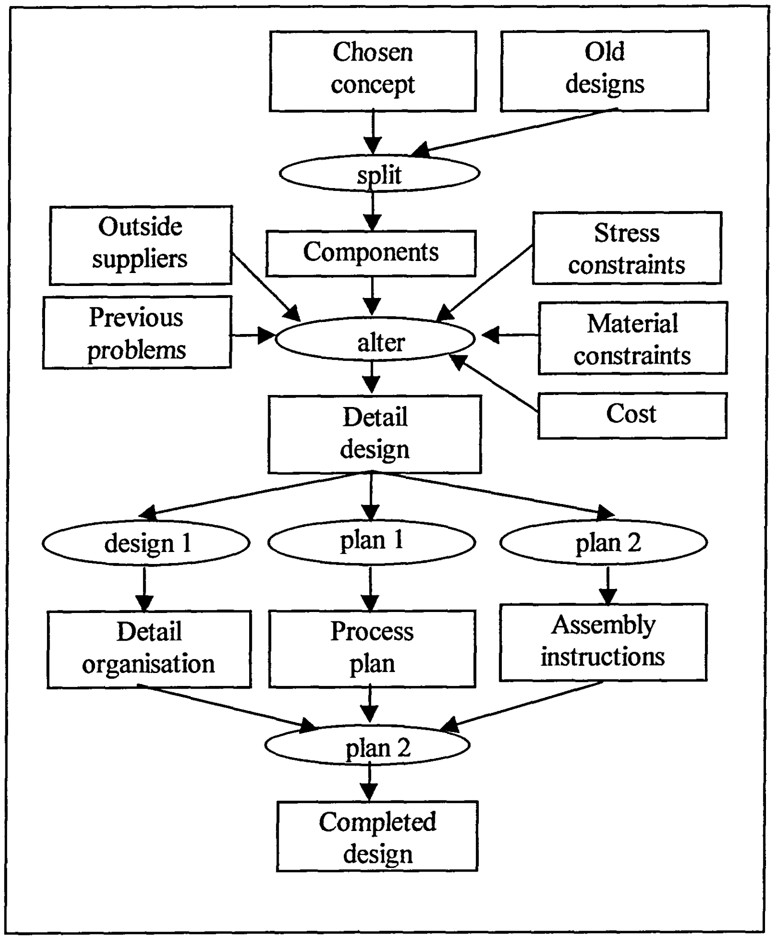


Figure 8.3: The Company 1 detail design model.

8.2.4 Case study 1 and propose-critique-negotiate

As shown in figure 8.4, the review process would be used to allow multiple life-cycle perspectives to analyse the concepts. This process would then outline possible problems in the evolving design. In addition, at a lower level of detail, designers have the option of consulting life-cycle experts at any time during the design process. As a result, such review stages were where propose, critique and negotiate were used.

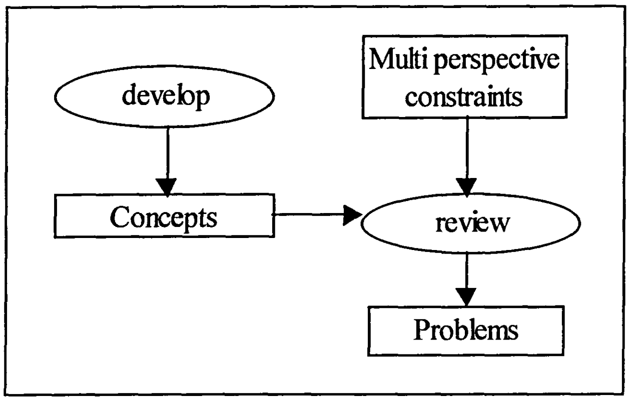


Figure 8.4: Propose-critique from case study 1

When different problems inherent in a design have been pinpointed, the design may be altered on the basis of the generated critiques of the design. This is illustrated in Figure 8.5.

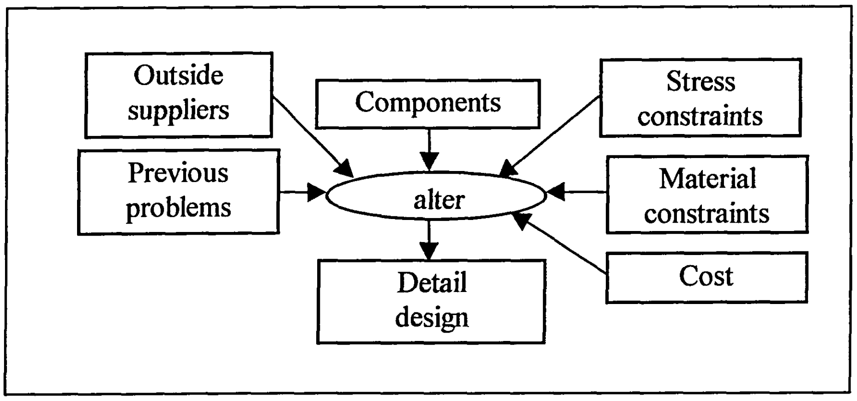


Figure 8.5: A 'modify' process from case study 1

8.3 Case study 2

On first visit, the design department 'skills coach' introduced their understanding of concurrent engineering and how this has been implemented in the design function of the company. The 'skills coach' has been widely engaged in the implementation of concurrent engineering in the company and the resulting organisational changes along the lines of BPR. He gave a slide show of how the company 2 new product development process occurs. However the company were reluctant to discuss their latest ongoing projects because of the problems of confidentiality. To overcome this hurdle, Designer 2 at the company then outlined their typical design process by using an old project as an example.

Designer 2 worked at a CAD station and loaded in an old project. He then ran through the design process used to generate the CAD model. This included details of how different perspectives were allowed to influence the development of the model (the CAD model produced can be considered the 'design' as from here the CAD model is passed on to the rapid prototyping function). During the course of the interviews, Designer 2 was encouraged to discuss how different perspectives were allowed to influence the evolving design. The interviews were recorded and transcripts from the interviews were used to generate *Common*KADS graphical task models of the designers' actions and interactions with other (particularly downstream life–cycle perspective) personnel.

8.3.2 The Company 2 overall design model

Initially, the marketing function analyses the market to determine the requirements of any new product. This will either be an innovative new product or a 'me too' type product, which basically mimics other products in the marketplace. The requirements for the product are then sent to their designers in London (this is an external design consultancy who do all their design work). The design will either be heavily specified if it is a 'me too' type product, however for more innovative products the designer is given more freedom to explore different alternatives. The external design agency then come up with a number of alternative concepts.

Company 2 then has to convert this to a 3-D solid model and do the full embodiment (detailing) of the design. At the moment this causes problems because there is a time lag while the company 2 designer converts the 2-D CAD drawing to a 3-D solid model. However they have just finished training the designers how to use the 3-D modeler and so from now on the designers will pass on a 3-D model to company 2 to avoid this step.

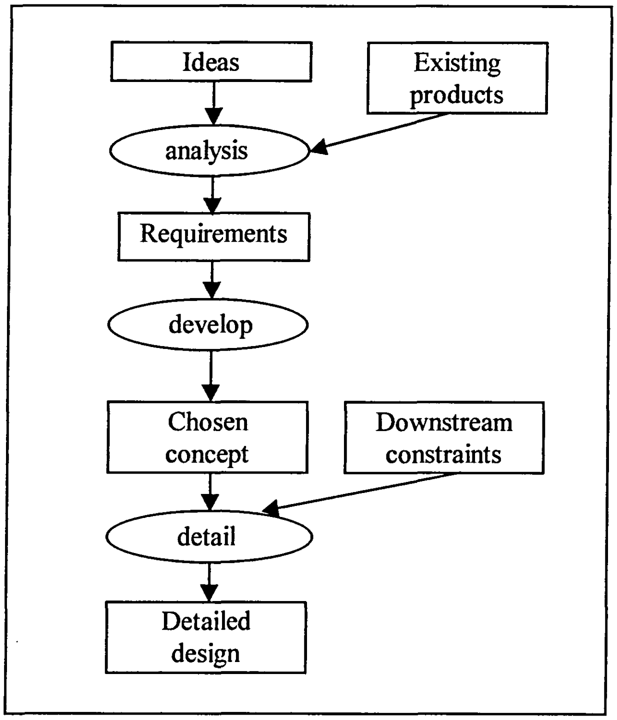


Figure 8.6: The Company 2 overall design model.

8.3.3 Company 2 and the 'develop' task

The major issue in the design is form, what the product looks like has a decisive effect on how it sells. It is assumed that all the products in the marketplace function adequately and so form becomes the main design criteria. The designer will typically come up with a number of concepts (usually a maximum of about 7) for consideration. These are in the forms of sketches and the designer may cut a foam model of the design (this can be done fairly quickly). Company 2 then chooses a few of these (two or three) which are looked at in more detail and some embodiment may be done on these concepts before one is chosen. The designers then give company 1 a 2-D CAD drawing of the chosen design.

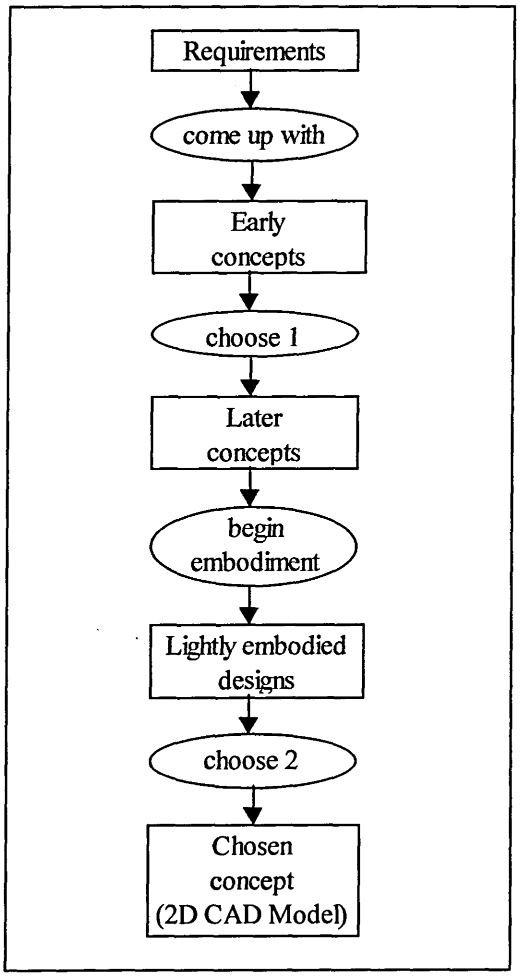


Figure 8.7: Company 2 'develop' sub-task.

8.3.4 Company 2 and the 'detail' sub-task

The company 2 designer performs the embodiment sub-task inherent in this process with input from the manufacturing function. This is however done on an informal basis and usually involves the designer asking the production function to look at something which they feel may be contentious. The main problem they have is that when the first moulds are made up, the resulting items cast from this are distorted. This is because the casting distorts due to it's own weight during the firing process. The level of distortion then has to be gauged and new moulds developed to try and alleviate this distortion. It may take up to four or five iterations to achieve the final production mould shape. This takes time and plays havoc with new product development planning because there is no way of telling how many iterations will be necessary to achieve a satisfactory master mould. Company 2 is currently looking at the use of FEA to analyse how a casting will distort during firing and adjust the mould shape accordingly.

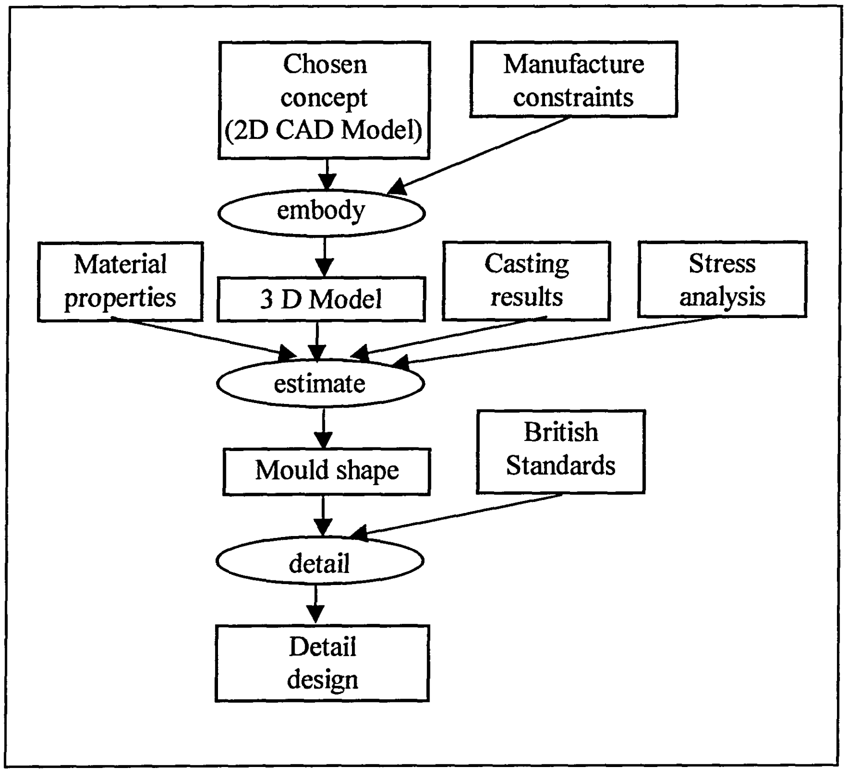


Figure 8.8: Company 2 'detail' sub-task.

8.3.5 Case study 2 and 'Propose-critique-negotiate'

The 'skills coach' was asked if there was much input from downstream perspectives during this process. He said there was some input but the feeling was that if the designer is constrained too much by manufacturing etc. considerations at the conceptual stage, the designs are likely to be compromised. This is one of the reasons why they employ outside design consultants as opposed to moving the design process in-house. It is only when embodiment design is done that things such as manufacture are considered.

There are also British standards that Company 2 work to with most of their products. These relate to the sizes of holes for taps, plugs etc. However for luxury models where they have pinpointed the exact taps which they wish to use, they may ignore the standards in order to realise a design. The kite mark stamp is not vital on all their products.

8.4 Discussion

Follow up interviews that were conducted with the companies in case studies 1 and 2 suggest that the developed models are suitable representations of the way they conduct the design process. However, the feeling of the knowledge engineer was that while the models give a good general overview of the design process at the two companies, none of the nitty-gritty problems of 'every day' designing really came out of the interview transcripts. The designers' accounts tended to gloss over any problems they had encountered. In retrospect, this reticence on the part of the designers is completely understandable. They are highly unlikely to offer a completely critical view of their and the company's design processes as any criticism would likely find it's way back to their superiors. Hence, while the task models are considered to be valid and useful reflections on how design in two typical organisations occurs, they are felt to be slightly 'bland'. I.e. the models tend to reflect the companies quality manual accounts of how design should occur and misses out on a lot of the interesting (and critical) everyday design processes.

These initial attempts to analyse design behaviour also underlined a number of other inherent limitations (not just to this study) of attempting to analyse designers in a real-life industrial setting. The 'ideal' solution would have been to record designer's behaviour (using both audio and video recording) over a period of time, say the complete development of a new product. However unless the interviewer has a very close relationship with the organisation concerned, this approach is almost impossible to achieve, due to a number of reasons which will now be discussed.

8.4.1 Confidentiality

Because of the competitive nature of new product development, companies are understandably reluctant to discuss critical ongoing projects. Both companies refused point blank to allow any video recording of their design teams and had to be gently persuaded to allow tape recorders to be used. In the case of company 1, who sub-contract for other companies involved in the aerospace industry, these security issues were of paramount importance, not only did they have their confidentiality to consider, but also the confidentiality of their client. By contravening this confidentiality, company 1 could well compromise future work with their contractors.

8.4.2 Time-scales

Both companies were engaged in design projects with tight time-scales and budgets. Anything that distracted their design teams from their primary aim of designing was detrimental to these constraints. Therefore while both companies very kindly allowed their design teams and designers to be interviewed, they were very reluctant to consider studies that did not have an immediately obvious short-term benefit to themselves. Hence while the analysis of design behaviour is crucial to the development of more effective support for the design process, the two companies understandably found it difficult to grasp the immediate, tangible benefits of the studies.

8.5 Chapter summary

The design processes utilised by two companies have been analysed. This has resulted in the development of a number of graphical expertise task models of design at the two companies. However there are a number of limitations inherent in attempting to analyse design in an industrial setting. The confidentiality issue means companies are very reluctant to allow recording equipment onto their premises (especially video recorders) and the tight time-scales experienced by designers attempting to bring products to market means they have very limited time available to devote to researchers attempting to analyse their behaviour. This means that the models outlined in this chapter give a good representation of the overall design processes the designers and organisations use. However, they do not unearth some of the real-life day to day issues that designers must confront and the strategies and methods used to overcome these problems.

In the next chapter I go on to discuss studies undertaken with more independent designers and design teams in an effort to overcome some of the limitations inherent in the case studies discussed in this chapter.

9 Design models: The revised reality

9.1 Introduction

The limitations in attempting to analyse design behaviour in a real-life industrial setting have been outlined in the previous chapter. In an effort to further analyse design behaviour, I decided to analyse the design activities of more independent industrial designers. I felt that because of the independent nature of the designers they would feel freer to discuss their design processes and strategies without being restricted by feeling tied to any one companies' practices and methodologies. Also, because of the retrospective nature of some of the studies, these designers would not be violating client confidentiality. Case studies 3 and 5 involve one independent designer while case study 4 analyses two designers who both worked on the same project.

I begin by introducing the context in which each case study was approached and then go on to outline the expertise models developed for the personnel involved. As in the previous chapter, the models presented are the final iterations of the developed and refined models. These resulted from a number of knowledge elicitation sessions with the relevant experts and subsequent knowledge modelling by the knowledge engineer.

9.2 Case study 3

The designer in this case study (Designer 3) has worked with company 2. The designer was initially interviewed about a project he had just completed to design a boiler. In this case, designer 3 did not use a CAD tool to illustrate the example but used sketches and language to discuss the designs. The interviews were recorded and transcripts from the recordings used to generate task models of his behaviour.

9.2.1 Designer 3 overall design model (1)

From interviews with designer 3 it became clear that a number of different overall models were utilised for the design process. The first model presented is the design process the designer uses with one particular client. This proceeds by using the design brief to come up with a number of preliminary proposals for the design. These are then evaluated against a number of different criteria to attribute evaluation marks against each proposal. These marks are used to reduce the number of possible proposals to a more manageable number. This more limited number of proposals is then analysed in more detail in order to select the chosen model. A number of different life-cycle perspectives including the tooling function, time-scales and costs are important influences on each of these different stages of design. The model for this overall design process is illustrated in Figure 9.1.

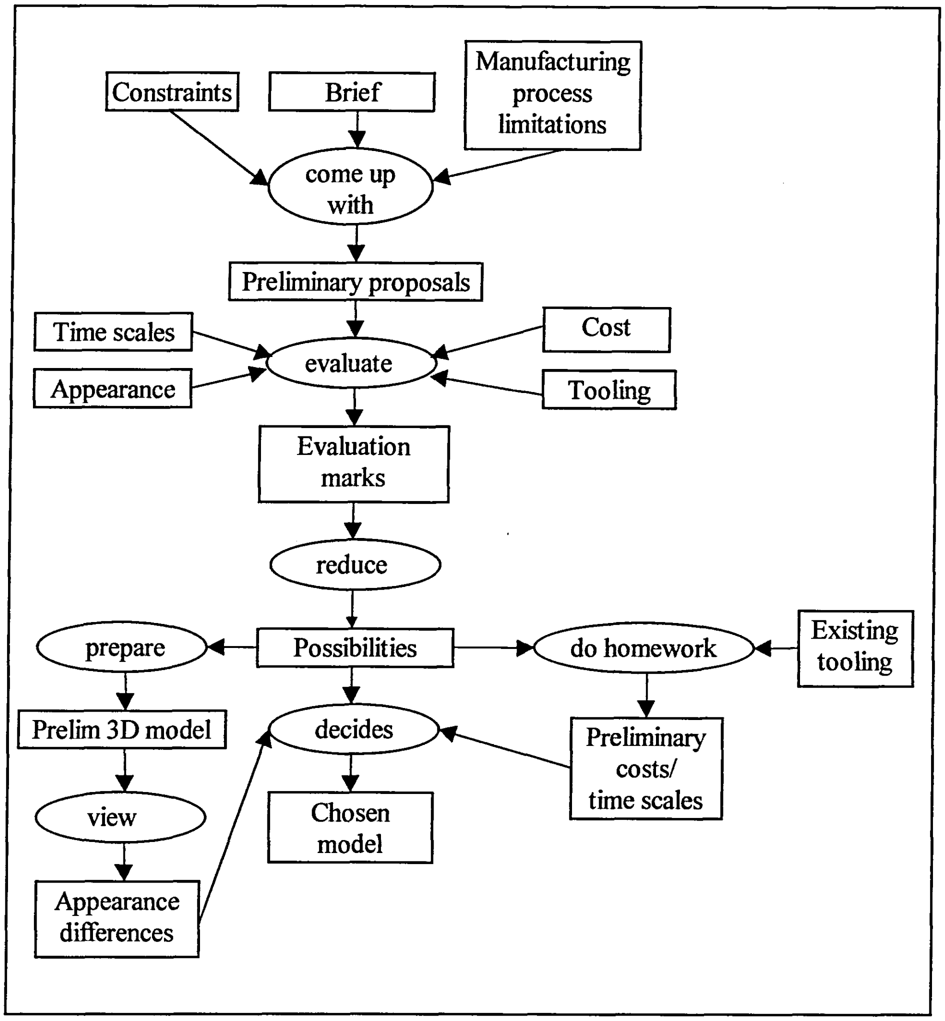


Figure 9.1: The Designer 3 overall design model.

9.2.2 Designer 3 overall design model (2)

However designer 3 was at pains to point out that this process is not necessarily the way in which he would like to design and then went on to use examples of previous design cases to illustrate the way in which he would generally design. When working with a company, designer 3 would like to start from basic principles and assess the marketing aims and objectives of the company together with any constraints the company might have to contend with (however he also outlined a case where the driving force behind the development of a new product had been the need to fill spare capacity in a client's manufacturing plant). A number of different design proposals would then be presented to the company and the thinking behind the designs outlined. The number of proposals would then be reduced via a negotiating and discussion process with key personnel at the company. These key personnel would typically represent different perspectives associated with the product's life-cycle. This model is illustrated in Figure 9.2.

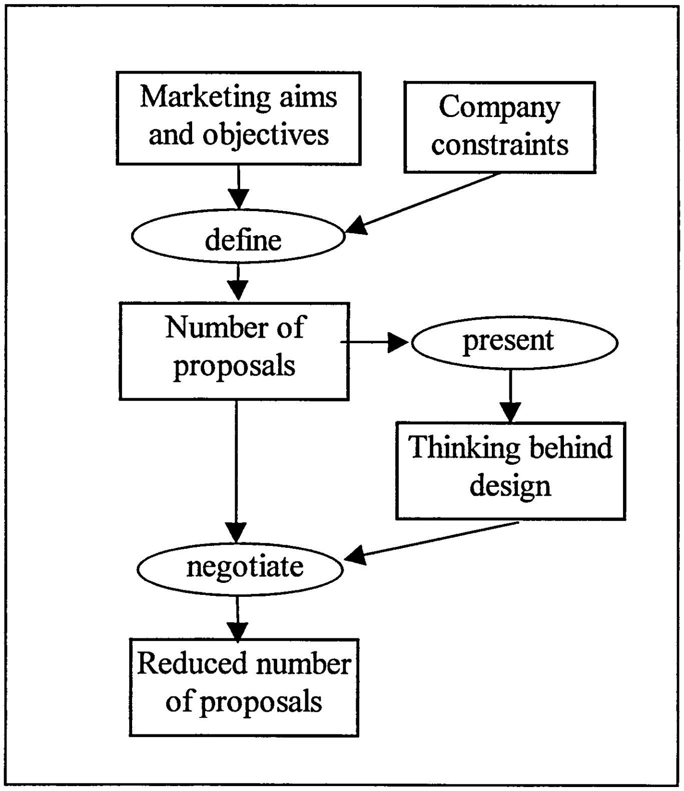


Figure 9.2: Designer 3 design model 2

9.2.3. Designer 3 and propose-critique-negotiate

Designer 3 appeared to work within the 'propose-critique–negotiate' framework that has been discussed. The way in which the designer 'proposed' designs was not extensively discussed although as outlined in previous sections, the market position of the company the designer works with together with any limiting constraints they have to work with are clearly important input knowledge roles. However designer 3 did offer some interesting insights into how the subsequent critiquing and negotiation processes can occur. These are discussed in subsequent sections and illustrated by excerpts from transcripts, where appropriate.

9.2.4 Critiquing

Figure 9.3 shows a typical scenario where the designer proposes design changes to a client who then assesses or critiques the proposal from different perspectives before deciding whether to proceed or not.

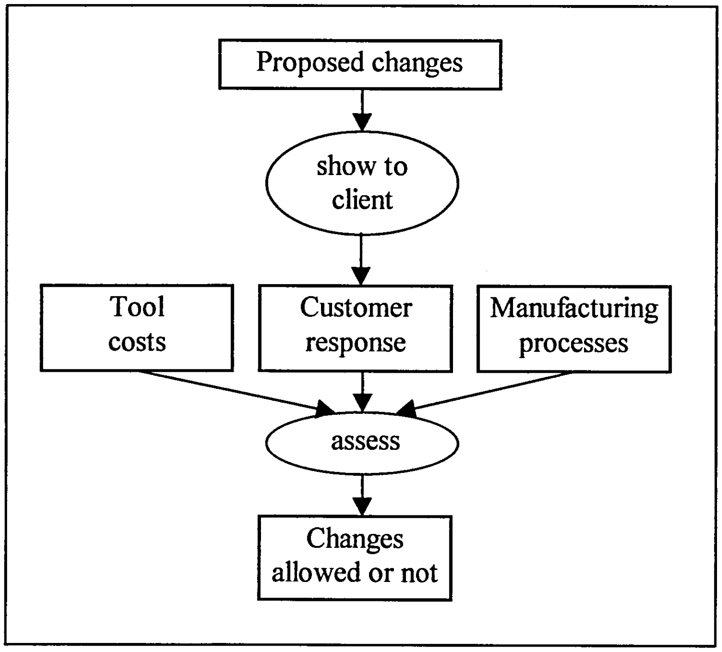


Figure 9.3: Designer 3 and design assessment

Designer 3 will use outside help if needed to analyse any concurrent constraints that will affect the design.

"..but I do sometimes ring up the guy and say let me just throw an idea at you - what would you say if we were to go down the route of doing it in this way or that way and I might get an initial reaction of "you can forget that because the cost of doing that is so enormous" or I might get positive response "yes well sounds interesting, quite a number of benefits" and that might enable me to go then into this stage with a bit more confidence knowing that I'd had a word with one or two people in the background and they are not frightened by the idea. Having said that, if I get a negative response from them in an informal way I would then have a tricky decision about do I carry on with it because I think the benefits of it are so strong that I'm still going to present it even though the people in the background have made hesitant noises about it..."

Designer 3 illustrates how different personnel have different views of the design and the resulting negotiation that occurs because of this.

"we put forward our proposals, the marketing people say yes absolutely fantastic, we've got to have that, the production people say there's no way we can make this and we then start quite an interesting stage of discussion and throwing ideas backwards and forwards and the production people if they're good people, and if they've got in their own way a creative way of thinking about the problems, they might then be coming in and saying well we can't actually do what you're saying there but if we did this or if we altered that we could achieve this which has actually got most of the characteristics which you're looking for there."

Hence, the downstream perspective is effectively offering the 'new proposal' with 'most of the characteristics' of the original proposal as a negotiating strategy. This is illustrated in Figure 9.4.

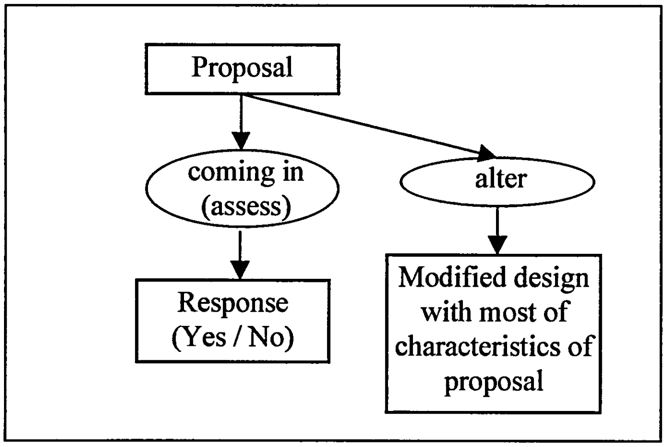


Figure 9.4: Response and modify process from case study 3.

9.2.4.1 Critiquing strategies from case study 3

Designer 3 will himself think through the implications a design implies before unveiling it to scrutiny by downstream perspectives, particularly for creative or non–routine designs, as illustrated in figure 9.5

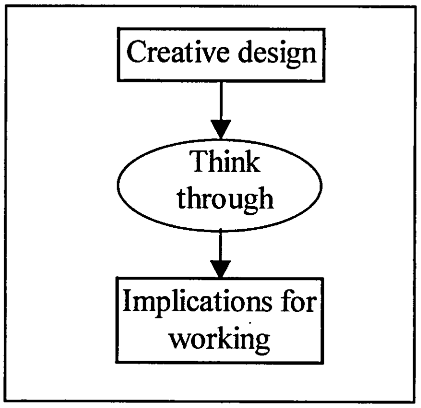


Figure 9.5: Designer 3 'Thinking through' a proposal.

Designer 3 felt that outlining the thinking behind a design before unveiling it to scrutiny by downstream perspectives is an effective strategy to allow the downstream perspectives to understand why various design decisions were taken and why the designer considers the results of these decisions as being desirable in the design. In this way, the downstream perspectives are introduced to the designers' way of thinking before they have a chance to critique the design, as illustrated in figure 9.6.

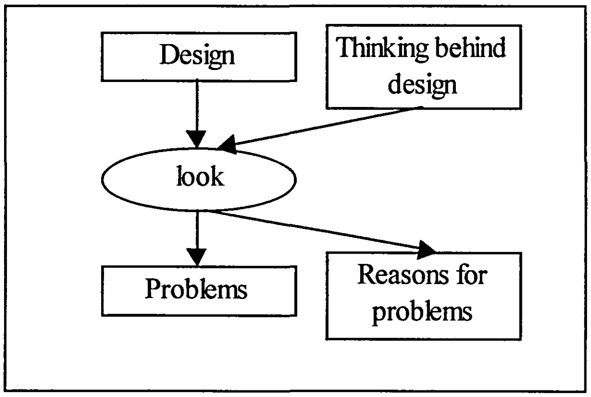


Figure 9.6: Designer 3 and the justification for a design.

9.2.4.2 Who makes the critique?

Designer 3 believes that he would rate the importance of a critique by who made it and also by the potential advantages the critique implies for the design as a whole.

"this might not be a very democratic comment, but also the importance, how I rate the importance of that guy in the company. If it's the teaboy, it doesn't matter how much he objects to this, if I think its right for the company, I'm going to be pushing it forward. If it's the technical director, and I know that what he says is going to have a lot of weight, then clearly I'd be foolish to ignore it."

9.2.5 Negotiation and negotiating strategies

The following transcript excerpts reveal some of the negotiating strategies utilised by Designer 3 and different life–cycle perspectives when the designer discusses the design proposals with clients.

9.2.5.1 Bright ideas as a negotiating strategy

" but obviously one does have to be flexible in some instances because in some instances, some times these people do have brilliant ideas about ways of doing it at a tenth of the price and its nearly as good and you can't ignore that obviously"

Hence different perspectives may not just critique the design, they may also assess a design and then propose different ways in which the functionality of the design can be achieved with more desirable results from their own perspective. There then needs to be a process of negotiation to decide whether these new ideas can be incorporated into the design. This process is illustrated in figure 9.7.

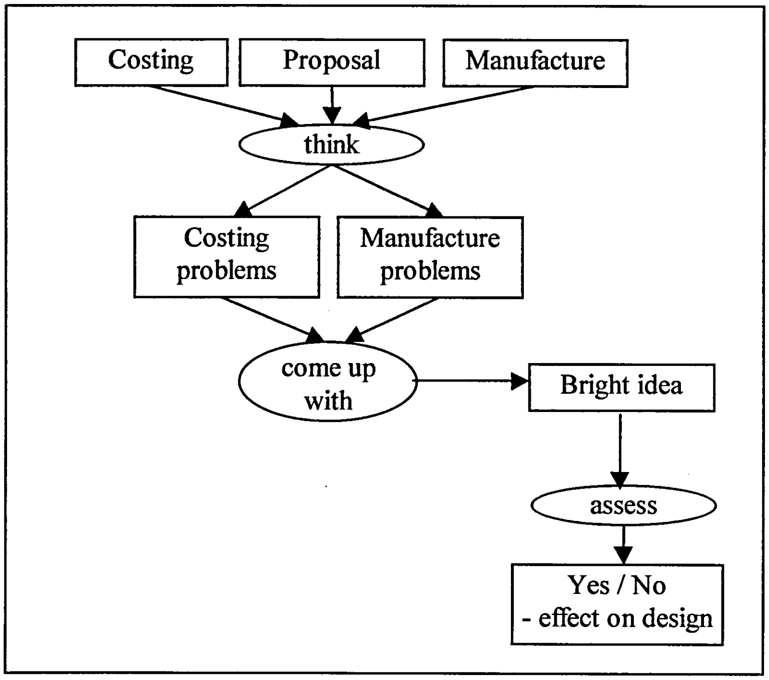


Figure 9.7: Alternative 'bright ideas' in negotiation.

9.2.5.2 Important features of a design

He also believes that an appreciation of what are the important features or crucial issues in the design and which issues are worth standing firm on are vital when determining how downstream constraints should be allowed to affect the design:

"One needs to know, well in my view, one needs to know, perhaps it only comes from experience, you need to know where to be flexible and where to be absolutely inflexible. You need to decide which really are the crucial issues in this design, does it really matter if we substitute this bolt for a standard bolt or not, will that ruin the design..."

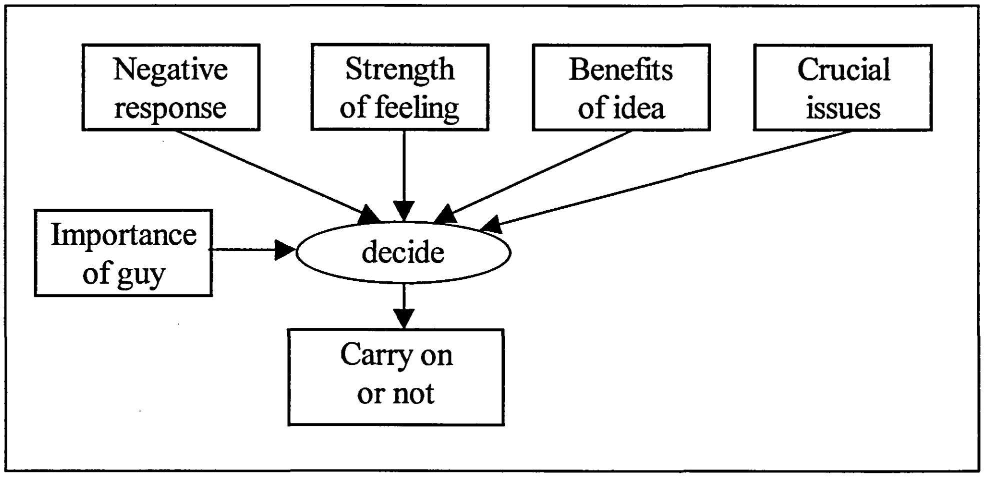


Figure 9.8: Designer 3 and important issues considered when changing a design.

9.2.5.3 An alternative proposal as a negotiation strategy

Figure 9.9 shows how designer 3 may also offer a new idea if the response to a design from other perspectives is negative.

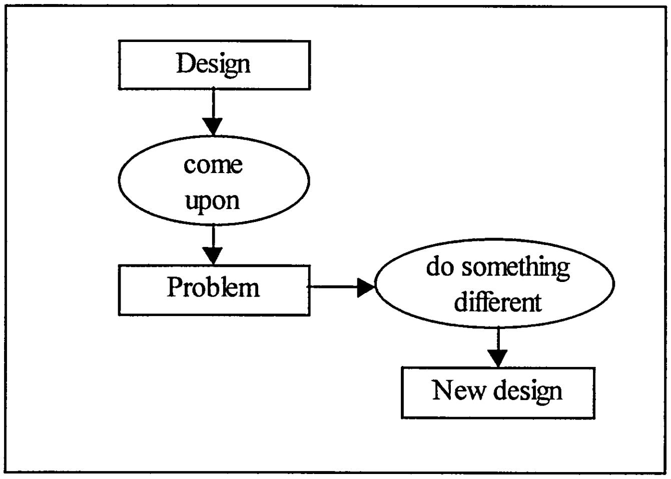


Figure 9.9: Designer 3 proposing new ideas.

9.2.6 Summary for case study 3

Hence this case study has begun to unearth some of the more real-life design issues inherent in a typical concurrent design environment which the previous case studies had to some extent failed to reveal. In particular, the critiquing and negotiation strategies employed by both the designer and the different life-cycle perspectives gives a greater insight into the concurrent design process than that evident from case studies 1 and 2.

9.3 Case study 4

This involved a project where two designers (Designers 4 and 5) developed a novel design for an electric powered vehicle. Designer 4 was an independent product designer whose main area of expertise was in the generation of form. He worked closely with Designer 5, who was head of product design at a company when the design was being developed. Designer 5 therefore also had to consider the business context in which the design process proceeded. Because both designers were heavily involved in the project from an early stage, the development of a brief or specification for the design features more than in the other case studies.

9.3.1 Designer 4: Overall design model

Designer 4 utilised a number of different information sources to inform the brief associated with the project. This brief was then used to drive the development of a 3-D CAD model of the design. This was then used to produce a detailed master-drawing and tool-pattern drawings. The overall model is illustrated in figure 9.10.

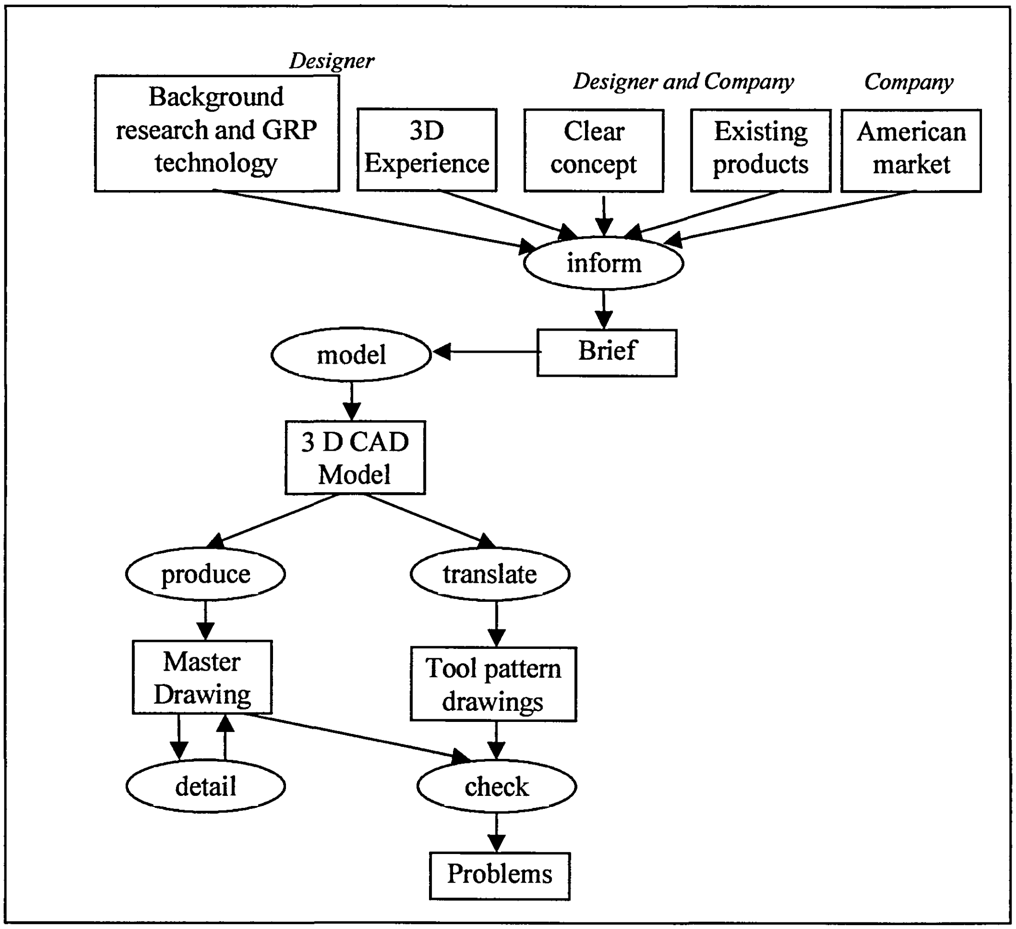


Figure 9.10: The Designer 4 overall design model.

This was elicited from observational studies and interviews with an industrial designer, working in a concurrent design environment.

9.3.2 Designer 4: Specification development model

The development of the specification is a complex task in its own right as illustrated in figure 9.11.

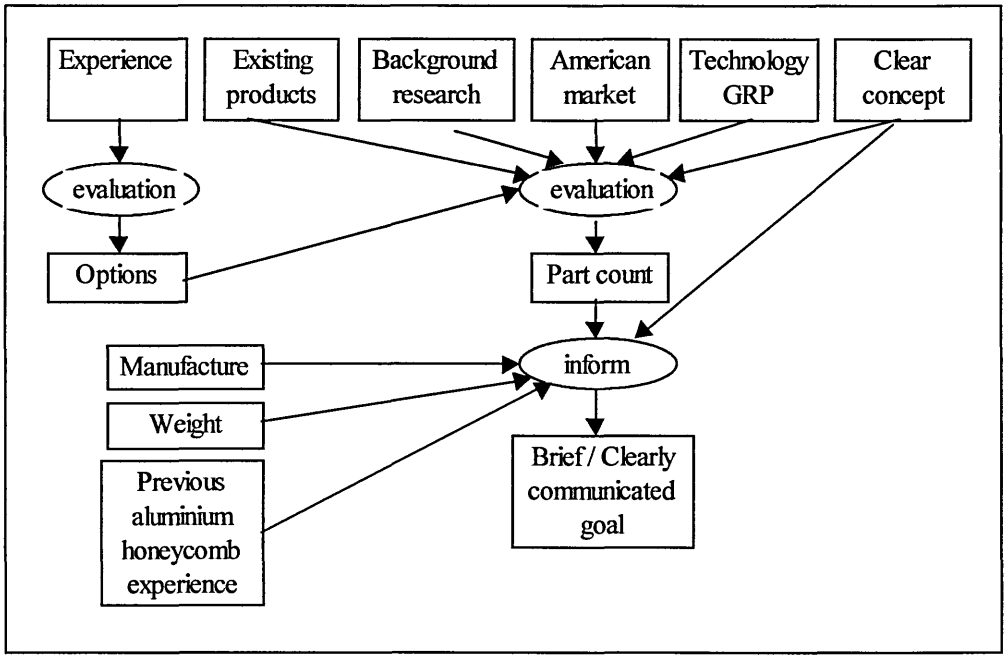


Figure 9.11: Designer 4 specification development model.

9.3.3 Designer 4: Concept development model

Once the 'brief' had been sufficiently resolved, this was used to drive the 'modelling' phase of the project where a 3D CAD model of the vehicle was generated. This is illustrated in figure 9.12

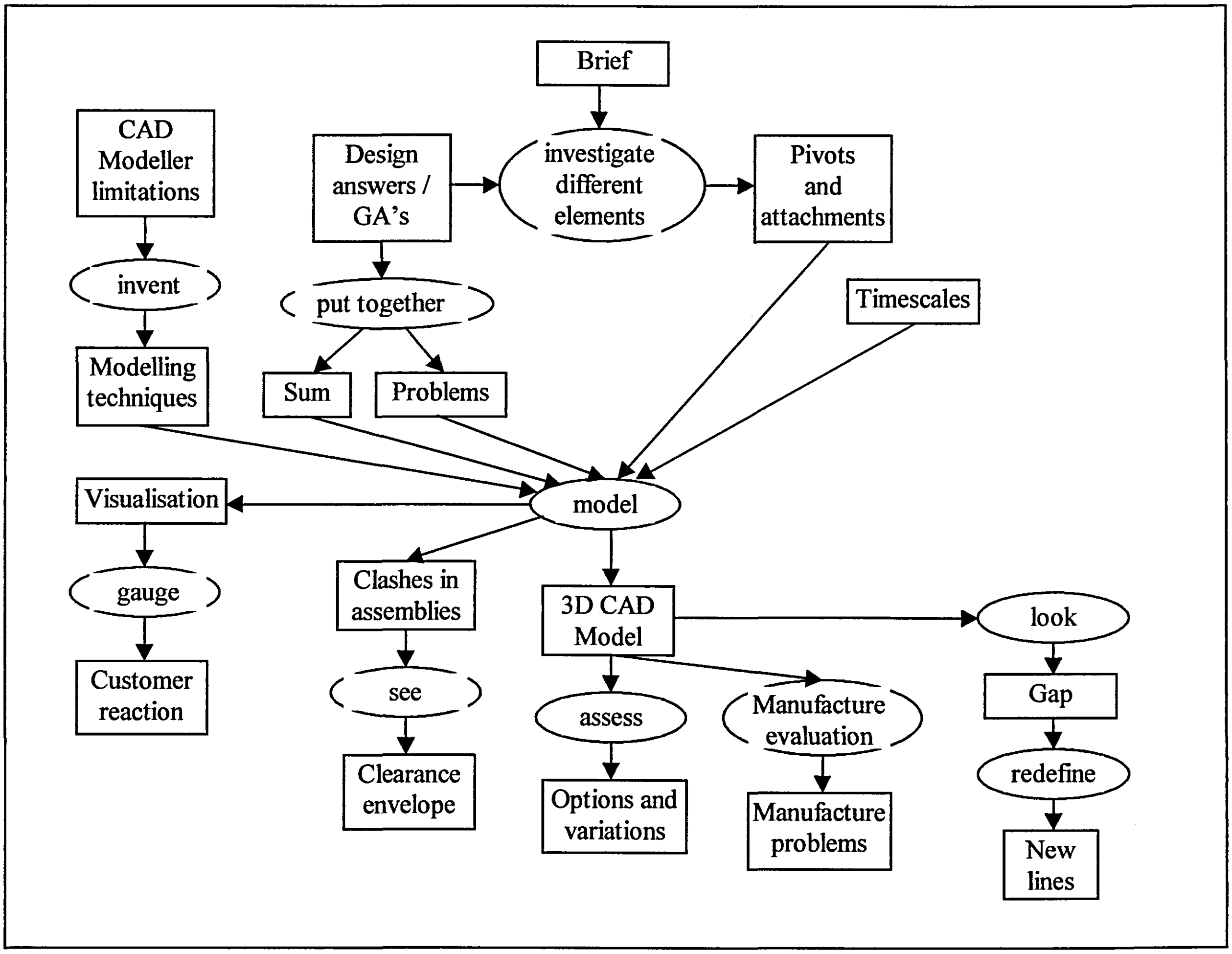


Figure 9.12: The Designer 4 'modelling' sub-task is an expansion of the 'model' task.

Evidence of where critiquing occurs is, for example, provided by the processes called 'see', 'assess', 'manufacture evaluation'.

9.3.4 Designer 4: Detail design model

The 3-D CAD model was then used to develop paper based master drawings for the vehicle and also the tool pattern drawings necessary to produce the moulds for the fibreglass body shell. This is illustrated in figure 9.13

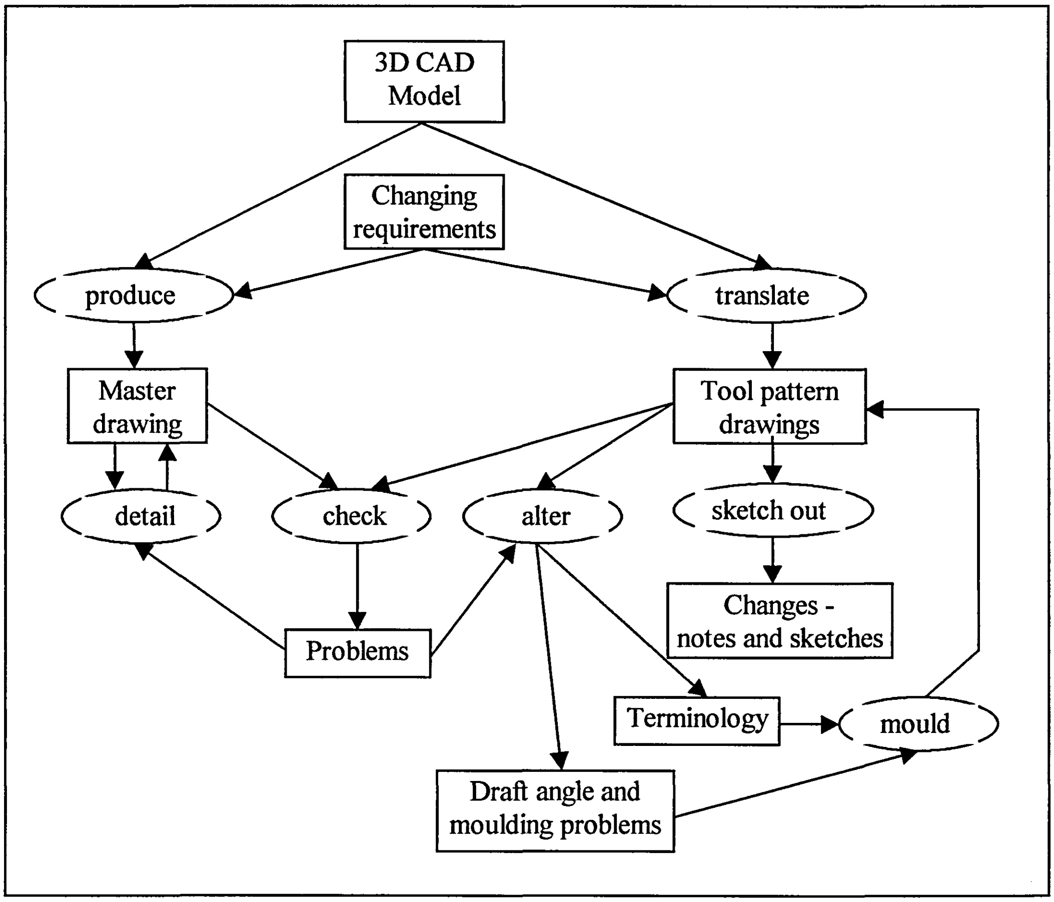


Figure 9.13: Designer 4 detail design model.

9.3.5 Designer 5: Overall design model

The design models presented for designer 5 show a lot of commonality with those presented for designer 4. However, differences emerged because of the different emphasis and importance the two different designers attached to different aspects of the design.

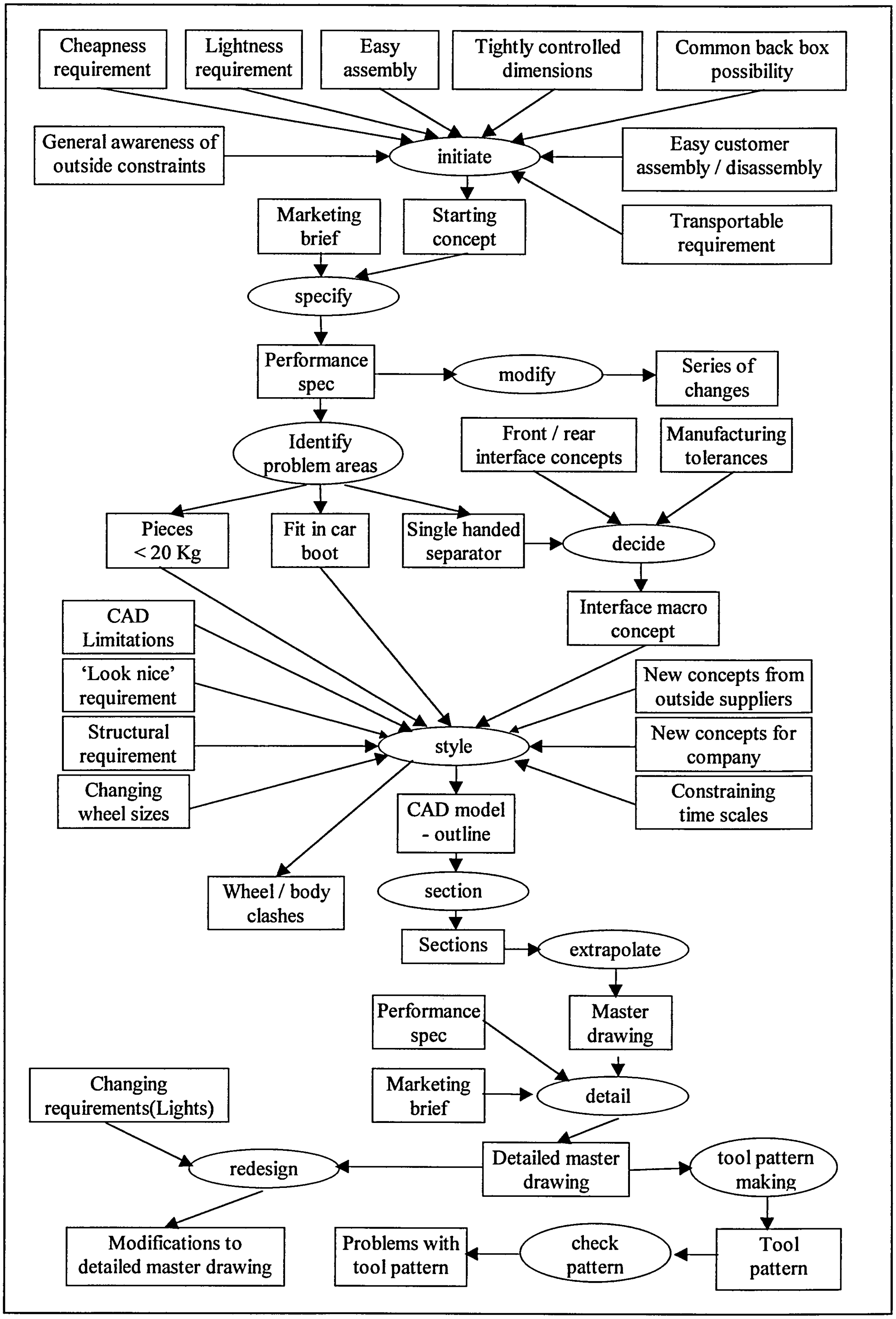


Figure 9.14: The Designer 5 overall design model

9.3.6 Designer 5: Specification development model

Because of the time lags between meetings, designer 5 suggested annotating the different knowledge roles and inferences with explanatory text to make understanding the *Common*KADS models' easier. This is illustrated for the specification development model for designer 5 in figure 9.15

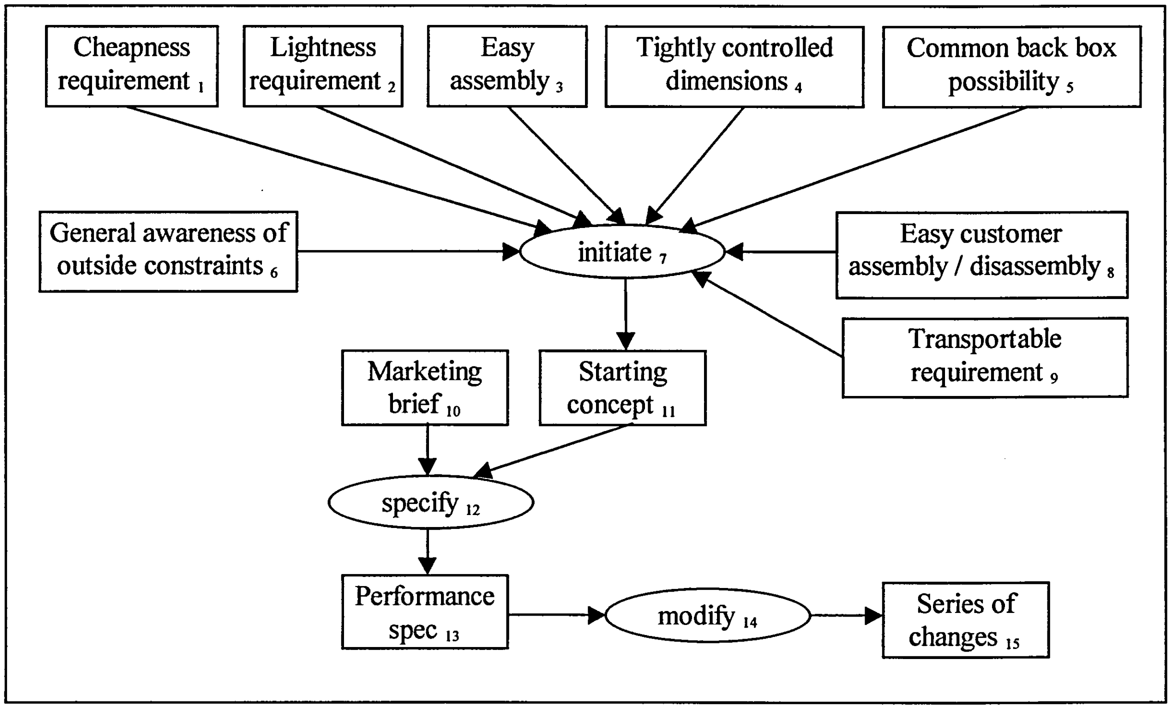

Cheapness requirement 1
Lightness requirement 2
Easy assembly 3
Tightly controlled dimensions 4
Common back box possibility 5
General awareness of outside constraints 6
initiate 7
Easy customer assembly / disassembly 8
Transportable requirement 9
Marketing brief 10
Starting concept 11
specify 12
Performance spec 13
modify 14
Series of changes 15

Figure 9.15: The designer 5 'specification development' model.

1. A cheap product was desired.
2. The product be light enough so that its constituent parts are liftable by a disabled person (Note: disabled, not weak).
3. The product be easily assembled by Booster.
4. Dimensions dictated partly by the Dutch market, minimum turning circles etc.
5. The possibility that a common back box with varying motor sizes be used to produce other models.
6. A general awareness of constraints such as the limitations of composites, welded steel structures etc.
7. The initiate process to generate the starting concept.

8. The customer should be able to easily assemble and disassemble the constituent parts of the product (i.e. the front and back).
9. The product be transportable in the boot of a standard sized car (conceived before the Ford Mondeo).
10. The marketing brief, describing in layman's terms what the product should do.
11. The starting concept based on the outlined requirements.
12. The process of specifying a performance spec from the brief and the starting concept.
13. The performance spec, dictating what the product should achieve in purely functional terms.
14. The process of modifying the performance spec (this was done after the spec had originally been drawn up). E.g. the provision of lights.
15. The changes resulting from changing the performance spec.

9.3.7 Designer 5: Concept development process

The process whereby the concept was generated in illustrated in figure 9.16

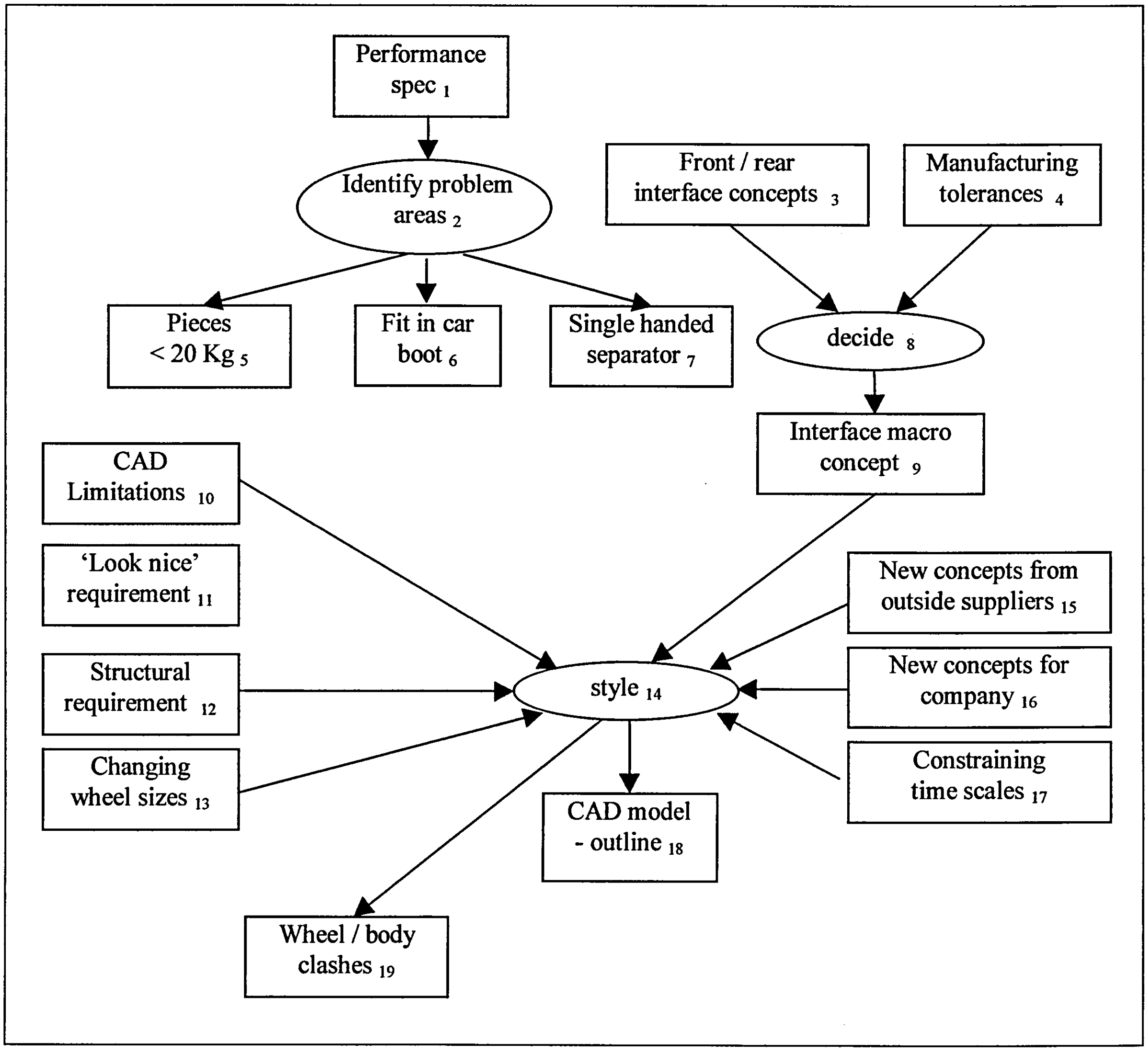

Performance spec 1
Identify problem areas 2
Front / rear interface concepts 3
Manufacturing tolerances 4
Pieces < 20 Kg 5
Fit in car boot 6
Single handed separator 7
decide 8
Interface macro concept 9
CAD Limitations 10
'Look nice' requirement 11
Structural requirement 12
Changing wheel sizes 13
style 14
New concepts from outside suppliers 15
New concepts for company 16
Constraining time scales 17
CAD model - outline 18
Wheel / body clashes 19

Figure 9.16: The designer 5 'concept development' model.

9.3.8 Designer 5: Detail design model

Having developed a concept, this was then detailed as illustrated in figure 9.17

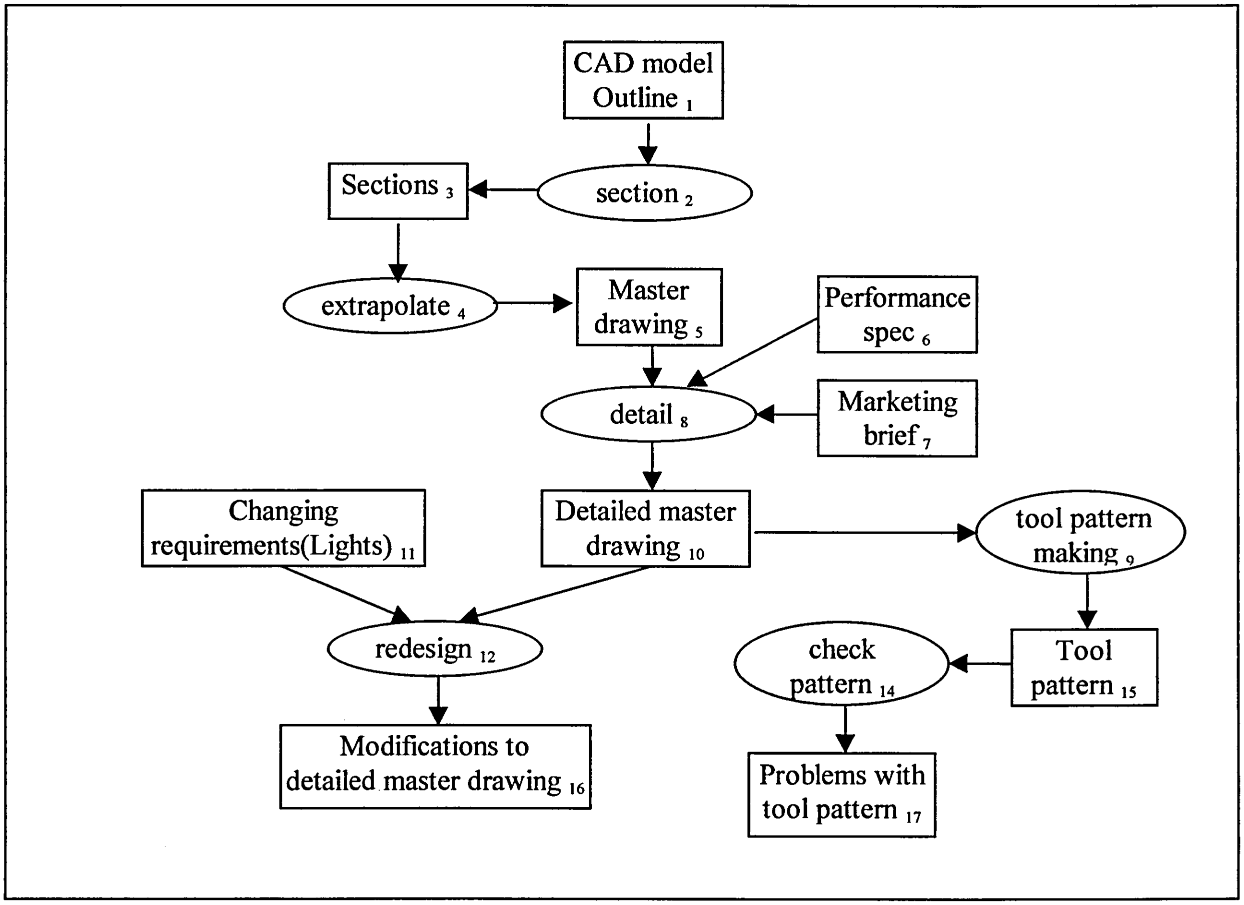


Figure 9.17: The Designer 5 'detail design' model.

9.3.9 Case study 4 and propose-critique-negotiate

An excerpt from interview transcripts with Designer 4 illustrate where different life–cycle constraints may arise.

"We had meetings ... it was just a printout we were looking at and someone said you'd have a problem getting in there with a screwdriver..."

Designer 4 utilised a form of self-assessment to evaluate a design, however this is to evaluate the functional performance of the design, not possible concurrent constraints, as illustrated in figure 9.18.

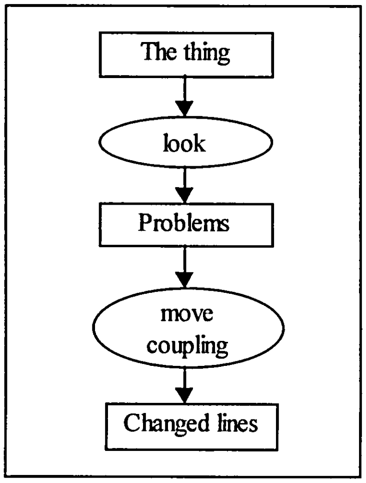


Figure 9.18: Designer 4 looking for functional problems in a design.

Figure 9.19 shows Designer 5 proposing a new design on the basis of perceived cost problems and his management team making a decision as to whether to accept the revised proposal.

"... there was a point where I offered to keep the front bit and manufacture the back bit out of steel which would have knocked thirty or fourty quid ... it would have knocked a few hundred quid off the cost..."

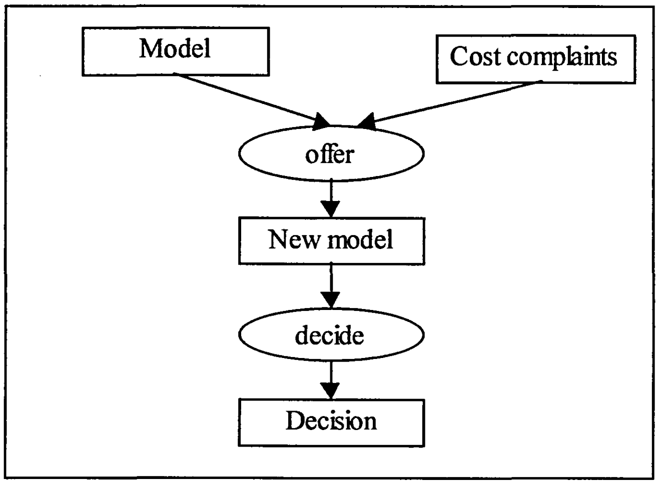


Figure 9.19: Designer 5 offering a 'new' idea.

9.3.10 Summary for case study 4

As for case study 3, this case study has unearthed some of the more realistic issues that occur in a concurrent design process. In particular, the issue of different personnel involved in the concurrent design process having different goals has been made explicit.

9.4 Case study 5

This case study features a designer developing plastic bottles for use in the drinks industry. As for case study 2, the function of the component being designed is assumed (although the designer clearly has to ascertain that this functionality is attained) and form and appearance are primary driving forces for the designer.

9.4.1 Designer 6: Overall model

The bottle designer (Designer 6) would start from a solid block of material on the CAD workstation and slowly modify this to produce a rough idea of the bottle shape. This would then be used to generate the 3-D CAD model and from this the 2-D drawing, as illustrated in figure 9.20.

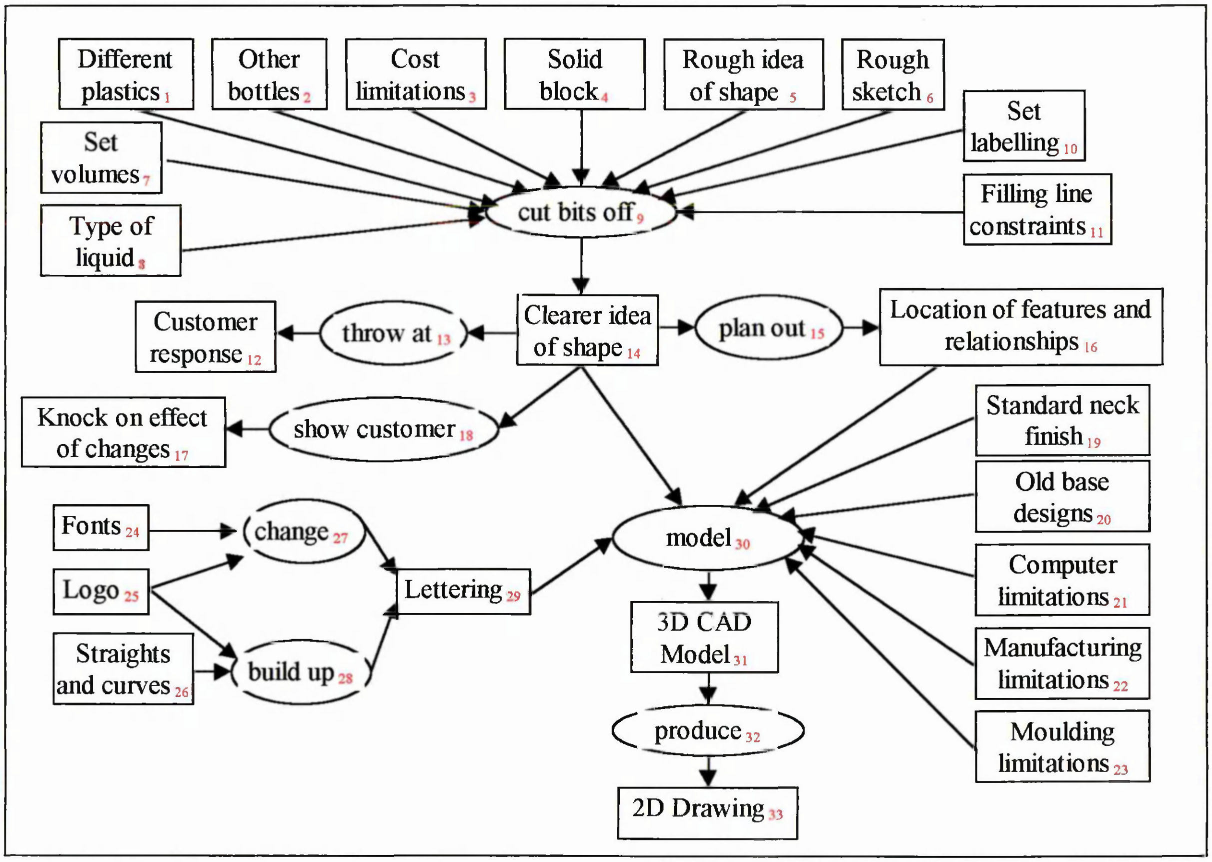


Figure 9.20: The designer 6 overall design model.

9.4.2: Designer 6: 'Cut bits off' model

The procedure via which the 'cut bits off' process proceeds is illustrated in figure 9.21

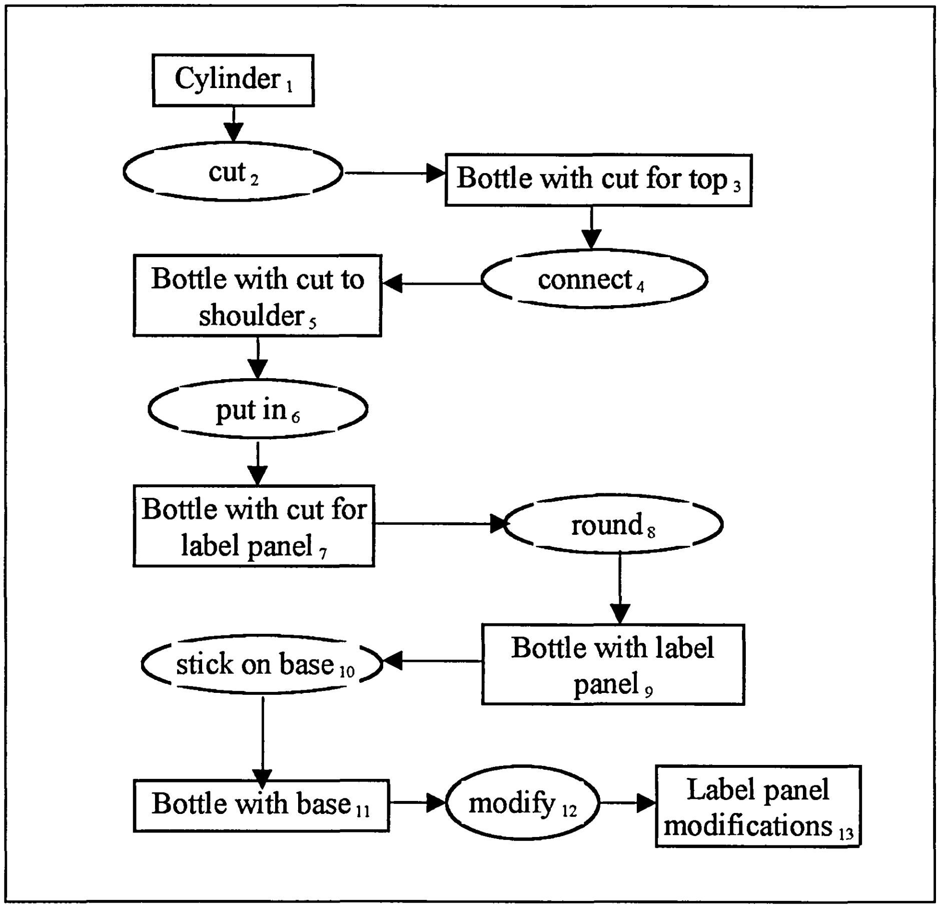

Cylinder 1
cut 2
Bottle with cut for top 3
connect 4
Bottle with cut to shoulder 5
put in 6
Bottle with cut for label panel 7
round 8
Bottle with label panel 9
stick on base 10
Bottle with base 11
modify 12
Label panel modifications 13

Figure 9.21: The Designer 6 'cut bits off' model.

9.4.3 Propose – critique – negotiate

Designer 6 would involve the customer in the design process as much as possible so that the client could appreciate the effect of design decisions. Effectively this is a negotiating strategy the designer uses with the client.

"and then he saw the knock on effect of what his changes were having and then within about an hour and a half we'd got round to virtually what he wanted and he had a good idea of what he could and could not change without changing other things and it's very useful to bring them in so that they can actually understand the design process".

9.4.5 Summary for case study 5

In this case study the 'voice of the customer' has been emphasised as being a perspective the designer must take account of. In this case the designer was in direct contact with the customer. In other case studies, the 'voice of the customer' has been indicated to the designer via some marketing brief or has been contained in the specification in some way.

9.5 Chapter Summary

Interviews were conducted with designers about previous and current design cases they have been involved with or were currently working on. Because the designers were independent and not connected with any particular company, they were much freer and less guarded with their comments about their design processes than was the case with designers interviewed in the previous chapter. From the resulting interview transcripts and follow up interviews, task models for their design processes have been generated.

In particular, the case studies indicated the importance of the propose, critique and negotiate tasks in the concurrent design process. A number of different strategies (or methods) were utilised by the designers in order to both appease and avoid confrontation with different life-cycle perspectives. However, where confrontation was inevitable, the designers were particularly keen to stand fast on important issues, which they felt were critical in maintaining the central characteristics of their designs.

In the next chapter, I bring together the different models and issues derived from the case studies and use these to derive more generic models for the concurrent design process and the organisational context in which it is practiced.

10 Constructing generic models for design and concurrent design

10.1 Introduction

In chapters 8 and 9 I have described the development of bottom-up, expertise task models showing the concurrent design processes utilised by different organisations and design teams. This chapter analyses the different models and draws out the generic characteristics shown by the models. In particular, this chapter focuses on a generic expertise task model for the concurrent design process, what part concurrent constraints, critiquing and negotiation play in the concurrent design process and what methods designers utilised in the case studies. These findings will have important implications for knowledge-based support for the concurrent design process.

I begin by illustrating the context in which the design process takes place in the different organisations. I then go on to look at the nature of the concurrent design task as practiced by these organisations and the then show how this can be represented as a task sequence of propose, critique and negotiate. Based on my analysis, I present my generic model for the concurrent design process. I also note some additional issues that are pertinent from the case studies.

10.2 The organisational context of the concurrent design process

It is important to note the subtle difference between the scenarios encountered in my case studies (where a single designer is effectively at the centre of the process and downstream perspectives are peripheral entities) with other more co-operative studies of group design (e.g. Cross et al [1996], Valkenberg and Dorst [1998]) where a number of different designers, with similar expertise, collaborate throughout the design process.

Typically the design process would be initiated with a brief (either a fully defined specification or an informal brief) from the customer or some form of market analysis (typically performed by the marketing function within the organisation). In case study 3 a situation was noted where a design brief was initiated by the realisation that manufacturing capacity was being under utilised, however the designer involved would not consider this a typical situation.

A designer would then be given responsibility for the design and the designer would mainly develop the evolving design. In the case studies, the designer would report either to the managing director of the organisation or to a project leader who would in turn report to the managing director. A very important point that came out of the analysis was the way in which a designer would take ownership of the design and the rationale behind the design. To the designer, the key goal was to achieve the functionality required of the design. However, this goal was not always shared by the different downstream perspectives involved.

However, in case study 3 the designer outlined the problems inherent in allowing the design of products to be purely consumer or customer driven. As he pointed out, when questioned about new products, consumers are limited by their perceptions of what is possible. A designer with knowledge of new manufacturing techniques, materials etc., will be able to come up with designs that would never have been derived from a purely market driven approach. He cited the Sony Walkman as a good example of a product being developed in such a way. Hence the designers 'original thinking' can play an important role in the design process.

In one case the conceptual stage of the design was performed by outside design consultants and in another case an outside consultant was one of the designers in the case study. Typically the designer would specify components, materials etc. and only call in outside 'experts' when they felt they have a problem. That is, when the designer, not the downstream expert, perceived there was a problem. In this respect, the designer is a 'jack of all trades' in that he or she needs to have an appreciation of downstream constraints (manufacturing, assembly, materials, etc.). After being called on by the designer, the 'expert' would either critique some aspect of the designer's work or where the designer felt unable to proceed, the 'expert' would propose some part of the design.

Companies 1,2 and 3 also claimed to hold multi-functional meetings at specific stages in the design of a product where any problems inherent in an evolving design were ironed out.

The current state of the design (which will be termed the evolving design) was manifest in a number of ways. Conceptual models were generally represented as sketches, CAD drawings or models. For the later stages of design (i.e. detail design) a CAD model would typically be used to represent the design. However, while CAD tools are very good at representing geometrical data, they do not support the design process itself to any great extent. Guan and McCallum [1996] outline some additional reasons as to why CAD systems are typically used in the later stages of the design process.

In some companies, the designer was situated in close geographical proximity to multi-functional 'experts' allowing easy and rapid communication. For example, case study 1 featured an open-plan office where the designer sat in close proximity to other members of the multi-functional team. However in other cases the designer was 'remote' and communication with multi-functional experts was more difficult.

10.3 The nature of the concurrent design task

My task-oriented view of the concurrent design process has outlined some of the problem-solving methods designers and other personnel involved in the concurrent design process utilise to solve a number of sub-tasks associated with concurrent design. In addition, the important knowledge roles that act as input and output to the tasks have been noted.

The different expertise task models for the complete design process derived from the different case study show remarkable similarities. They show that generally, design can be very broadly split up into two distinct phases or tasks, namely analysis and synthesis. In the analysis phase a comprehensive specification is derived from some brief outline of the product's requirements. During the synthesis stage, a design solution is developed from the design specification. In this research, the synthesis stage has been analysed in considerably more detail than the analysis stage. However, there was considerable evidence for the two tasks proceeding in parallel in my case studies.

In order to illustrate the main tasks and knowledge roles utilised by the designers in the case studies for the analysis and synthesis stages of design, I have summarised these in tabular format.

Input Role(s)	Task(s)	Output Role(s)
Designer 2		
Ideas	analysis	Requirements
Existing products		
Designer 4		
Background research and GRP technology	inform	Brief
3D Experience		
Clear concept		
Existing products		
American market		
Designer 5		
Cheapness requirement	initiate	Starting concept
Lightness requirement		
Easy assembly		
Tightly controlled dimensions		
Common back box		
Starting concept	specify	Performance specification
Marketing brief		

Table 10.1: Tasks and knowledge roles for the analysis stage.
These knowledge roles and tasks have been derived from models of expertise developed for the different case studies and utilise the terminologies used by the different design experts. As an example, the 'analysis' task performed by designer 2 has input roles 'ideas' and 'existing products' and output role 'requirements'.

Table 10.1 summarises the major knowledge roles and tasks utilised by designers during the analysis stage of design. We can see from Table 1 that existing products, some idea about the possible form of the eventual design solution, a marketing brief and constraints, both from life-cycle perspectives (easy assembly, cheapness) and functional requirements (lightness) are important roles input to the analysis phase. Tasks utilising terms 'analysis', 'inform', 'initiate' and 'specify' result in the generation of output roles including 'brief', 'requirements', 'starting concept' and 'performance specification'.

Hence I can abstract the roles and tasks from this analysis phase of design into a more generic model, illustrated in Figure 10.1.

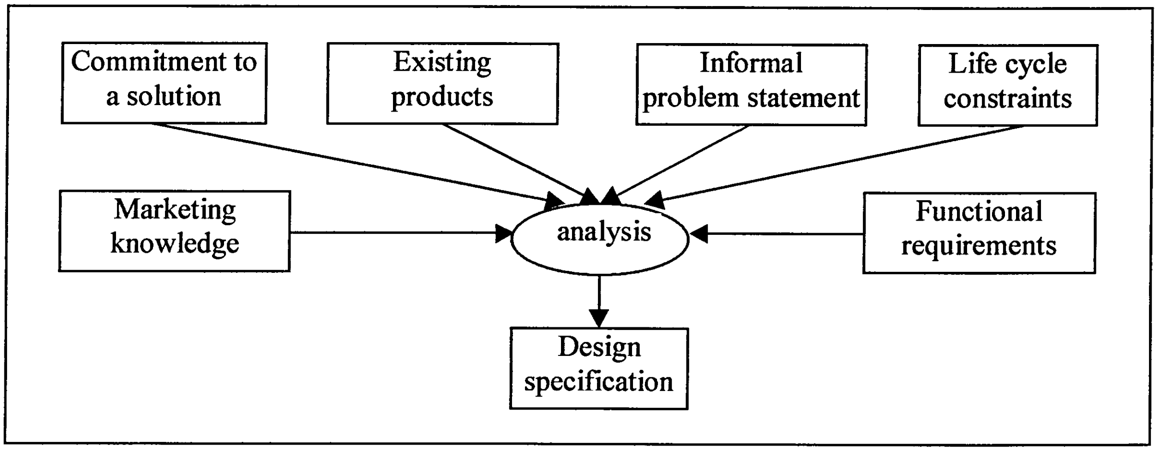


Figure 10.1: The analysis stage of design.

This model is based on the knowledge roles derived from the different case studies and uses abstracted terminology to derive 'generic' knowledge roles and tasks for the stage of design where a specification is generated. This model shows the abstracted roles 'Commitment to a solution', 'Existing products', 'Informal problem statement', 'Life-cycle constraints' 'Marketing knowledge' and 'Functinal requirements' acting as inputs to the analysis task. The important output role I have abstracted to a 'Design specification'. This is typically represented as a set of requirements (or functions) that the product must satisfy. These dictate the actual performance of the finished product. The design specification also contains a number of constraints that limit the way in which the design progresses. Life-cycle constraints are the designer's knowledge of different life-cycle perspectives. These include issues such as general knowledge of a certain manufacturing process' limitations. A commitment to a solution is a general representation of the form that the final design may take.

Only Company 2 did not follow this analysis phase, as the specifications presented to them had already been rigidly defined. For the case study analysed with company 2, they began with a completed design specification, which dictated the performance of the product together with a variety of constraints. Interestingly these constraints were mainly in the form of an exclusion list of the form 'you will not use this particular manufacturing process' etc. Hence what I have termed 'internal' constraints were actually generated by the customer in this case although the company who present them with the specification go through an analysis phase to actually derive this specification.

While developing the design specification, evidence was found for designers already being committed to a solution while the specification was being formulated. In case study 4, for example, the designers already had a 'clear concept' that they intended to use some form of composite material for the 'chassis' of their electric vehicle before the design specification was completed. The other case studies also suggest that the designers at least have some pre-conceived ideas about a possible solution before the design process begins and this commitment to some form of solution stays with them, even though the form of the solution may change during the course of the design process.

It is the synthesis stage of design that this research is predominantly concerned with. During the synthesis phase, a suitable concept is derived from the specification and this is then detailed to arrive at the final design. A possible intermediate stage between the development of a suitable conceptual model and final detail design is embodiment design, where the layout and form of a product are defined. During this phase, more concrete constraints are input from 'experts' in different life-cycle perspectives.

In order to illustrate important knowledge roles and tasks for these two stages, tables 10.2 and 10.3 summarise findings from the case studies.

Input Role(s)	Task(s)	Output Role(s)
Designer 1		
Multi-functional constraints	generate	Concepts
Specification		
Old designs		
Designer 2		
Requirements	develop	Chosen concept
Designer 3		
Constraints	Come up with / evaluate / reduce / decide	Chosen model
Brief		
Manufacturing process limitations		
Marketing aims and objectives	Define / negotiate	Reduced number of proposals
Company constraints		
Designer 4		
Brief	model	3D CAD model
Designer 5		
Performance spec	style	CAD model outline
Marketing brief		
Starting concept		
Designer 6		
Different plastics	Cut bits off	Clearer idea of shape
Other bottles		
Cost limitations		
Solid block		
Rough idea of shape		
Rough sketch		

Table 10.2: Tasks and knowledge roles for the conceptual stage.
This illustrates the different roles and tasks, derived from the case studies, used in the synthesis stage of design. This shows how different roles including the design specification, a commitment to a solution, various constraints (both life-cycle and other) act as important input roles to the development of concepts.

The output from the stages outlined in Table 10.2 I have abstracted to the conceptual model. However, exactly what constitutes the 'conceptual model' for the design is difficult to pin down. It would perhaps seem more appropriate to consider that the designers and design teams become more committed to a particular conceptual model for the design as the design process progresses. Hence a current design model at any particular time may contain elements of a conceptual design and a detailed design. This way of working in which designers can switch between looking at elements of conceptual and more detailed design at will has also been noted by Olson et al [1992] in an analysis of early software design meetings.

Table 10.3 illustrates the important roles and tasks for the detail phase of design. The output role, usually in the form of a drawing, I shall abstract to the term design solution.

Input Role(s)	Task(s)	Output Role(s)
Designer1		
Concepts	detail	Completed design
Old design		
Designer2		
Downstream constraints	detail	Detailed design
Chosen concept		
Designer4		
3D CAD model	produce	Master drawing
Designer 5		
CAD Model outline	Section / extrapolate / detail	Detailed master drawing
Designer 6		
Clearer idea of shape	model	3D CAD model

Table 10.3: Tasks and knowledge roles for the detail stage.
These knowledge roles and tasks derived from the case studies are used where a conceptual design is detailed to give a completed design. This illustrates how the conceptual model, life-cycle constraints and previous designs act as input to the detail design phase.

Based on these findings, we can see that my case studies suggest that the synthesis stage of design consists of two stages of conceptual design and detail design. The problem-solving method, which is used for these two stages, is what I have termed concurrent design. The abstracted model for this is illustrated in Figure 10.2.

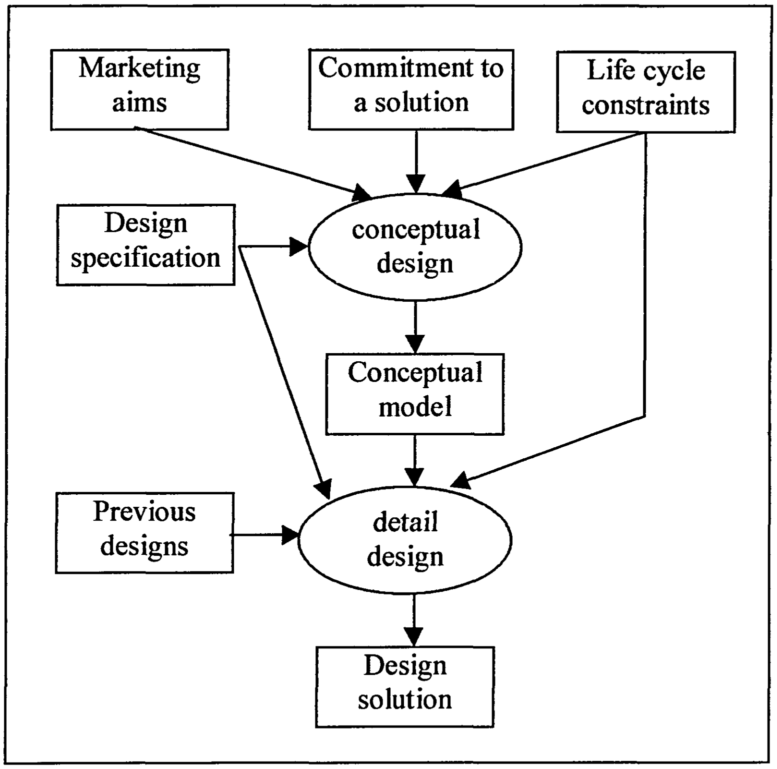


Figure 10.2: The synthesis stage of design.

This can be seen as two stages of concurrent conceptual and concurrent detail design. The problem-solving methods of propose, critique and negotiate are used to achieve both tasks. This model was developed from 'generic' knowledge roles and tasks derived from the different case studies. Published in Barker and Meehan [1999].

The models outlined so far are similar to the original KADS model for design presented in Tansley and Hayball [1993] (see Appendix A) and also agree with the more abstract model of Bernaras and Van de Welde (which sees design as a process of analysis and synthesis). However the intermediate steps (or subtasks) exhibited by the case studies together with the additional knowledge roles acting as input to the tasks expand on the original KADS models and also differ from the models developed by Bernaras and Van de Welde [1994].

The models presented in Bernaras and Van de Welde are fairly high-level or coarse grained. They do not explicitly make clear the way in which a number of different problem-solving methods may be combined and applied to the design synthesis task during the course of its execution. (I.e. from formal specification to completed detail design). Chandrasekaran [1990] and the results of my case studies would tend to indicate that the design process (from specification to detailed design) consists of a number of sub-tasks with various problem-solving methods being used for each sub-task. Also the types of knowledge roles that act as input to the different sub-tasks / inferences noted in my case studies expand on the roles noted in the literature. In particular, the influence of multi-functional internal constraints plays a large part in influencing the design process. However, the degree to which multi-perspective constraints impinge on the design process also varied considerably between the case studies.

One critical point not always made explicit in the literature is the importance that the design specification has in the overall design task. As a knowledge role, the design specification acts as an input role to most inferences that take place during design synthesis (i.e. a detailed design is not derived just from a conceptual design as the original KADS model presented in Chapter 2 suggests, the design specification also acts as an input knowledge role). Because analysis and synthesis were also seen to proceed in parallel, the design specification acting as an input role was not necessarily the final design specification. There is also the possibility of the design specification changing as the design progresses, so this knowledge role must act as an input to all stages of the design process. If computer support for design is to be derived from these models, the availability of the current design specification at all stages to the designer is clearly an important one.

The means by which conceptual and detail designs are developed I have abstracted to the tasks 'concept' and 'detail' in figure 10.2. It is here where I believe the tasks propose, critique and negotiate are utilised. Hence it would be more appropriate to represent these two tasks with the more abstracted 'concurrent design'.

10.4 Propose-critique-negotiate (from interview transcripts)

My previously presented model for design is a general model. It gives the overall process (and implied) subtasks that designers utilise but is at a very coarse level of detail. It is necessary to analyse at a finer level of detail the tasks and methods that the designers in the case studies used in their work. I.e. an expansion of the conceptual and detail design tasks (shown in figure 10.2) and important knowledge roles utilised in this process.

The way in which multi-perspective constraints impinge on the design process seems to occur in a two-pronged manner. At an early stage of the design process (specification development and early conceptual design) the designers seemed to use their own knowledge of these internal constraints to guide them. This implies that a designer utilises some internal form of critiquing to reflect on a product from different perspectives (at this stage, the designers were also seen to reflect on the functional implications of their design decisions). The way in which the different designers consider these constraints is illustrated by the quotes from the interview transcripts.

Designer 3 liked to know what these constraints were at an early stage so that any conceptual design produced was 'informed' by these constraints (i.e. the designer does not want to produce 'ridiculous' designs which are 'impossible' to manufacture and assemble).

"I would like to stress that I do not come up with barmy solutions that there is no way of making and I do understand the way that most products are made and I do understand a lot about those different processes having been in the game for twenty five odd years, ... "

However, while the designers in the case studies acknowledge the existence of these constraints, they try not to let the constraints limit their creativity as designers. Hence while they would not attempt to generate designs that are 'impossible' to design, they still consider their designs as being central and the constraints as something to be worked around.

Case study 1, Skills coach :
"..if the designer is constrained too much by downstream considerations at the conceptual stage, the designs are likely to be compromised."

Case study 4, designer 5 :
"You've got to know generally what you can and can't do with mouldings but that's what I mean you should only bear in mind physical constraints or else you'll end up letting that guy design the thing for you"

The type of interaction suggested by the case studies between designer and other perspectives is not necessarily of a collaborative nature. It would be tempting to think that while team members perceive the design differently, all are united in the common aim of developing a successful product. Clearly, this is not always the case. As pointed out by the designers, other perspectives such as manufacture, assembly etc. are likely to critique things so as to make life as easy as possible for themselves or are likely to perceive different things as being important. In the case studies where the downstream experts were part of the organisation, the design process was generally co-operative in nature although there was evidence of these 'experts' considering their own particular goals before the optimality of the design. However, where a downstream expert belonged to a sub–contracting organisation, clearly their goals were not the optimality of the design but how easy it would be for them to do their job when the time came.

Case study 4, designer 5 :
"At the end of the day, the pattern maker or toolmaker - what you're paying for is time. If he thinks he can do it in half the time by just straightening that curve or taking a joggle out, he'll do it, or rather he'll try and do it."

Case study 3, designer 3 :
"..but if you ask a production engineer to design a product, he will frequently start with the ease of production as being by far the most important consideration and he will not give any serious consideration to the user and marketing needs of the product and I believe that is why I can do something that they cannot do."

Case study 4, designer 5 :

"and almost always what I find important is not what they think is important because we've got such different priorities".

Also, by not allowing life-cycle constraints to influence the early stages of the design process, this can also act to encourage downstream constraining factors to evaluate the degree of their constraints. I.e. this helps to push new manufacturing and assembly processes. However, at later stages of the design process (i.e. finishing conceptual design and detail design), external constraints are allowed to influence the design process. In general, the designers seemed happier to consider these constraints at this later stage, possibly because by then the basic character or format of the design is already defined. Hence different life-cycle constraints can be seen as having a temporal weighting based on the stage of the design process.

Only in case study 1 were downstream constraints allowed to influence the design from an early stage. The reason for this is felt to be due to the type of product being designed. In the other case studies the products being designed were consumer items where form and appearance are of vital importance. Hence this seems to take precedence over other considerations early on in the design process before any downstream constraints are considered. However in case study 1, issues such as the function, reliability and 'solidity' of the product are paramount.

Once different constraints have been outlined by different perspectives, different negotiating and mediation strategies are used to determine if the problems outlined are allowed to influence the design. However, designer 5 outlines the importance of standing firm on certain issues and this theme was strongly echoed by the other designers who had a 'clear vision' for the evolving design.

"There's always a compromise but you don't want to start from a compromising position, you want to start with what you want"

Clearly such 'human' conflict resolution and negotiation strategies are an integral part of the design process.

10.4.1 Propose methods

The case studies showed that designers utilise a number of different problem-solving methods to accomplish the design 'proposal' task.

Case study 4 showed Designer 5 taking a functionally driven approach to the design where a functional specification was generated, which did not make reference to the physical implementation (although Designer 4 also working on the project already had a 'clear concept' of the form of the solution!). There was also considerable evidence for past design experiences being considered when proposing new designs or partial designs. This is effectively a form of case-based reasoning. These tally with typical methods outlined in the literature. However, none of the case studies showed evidence of mechanical based designers using transformational rules to achieve the propose task.

Key knowledge roles input to the design proposal task can depend on the PSM being utilised for the proposal task. For instance, when a form of case-based reasoning is being used to propose a design, an important input role is a library of previous designs. Clearly, by utilising this approach, a designer will also have made a significant commitment to the form of the solution. Where functional decomposition is being used, the designer utilises a library of components, which realise the functionality of the different sub–functions. One particularly important role outlined by a number of designers was the use of a library of generic components which can help to reduce costs across an organisation's product lines.

Extracts from the transcripts also revealed designers thinking in the 'reflection-in-action' mode outlined by Schon [1991]. Instead of developing a design, then assessing it, designers in the case studies attempted to 'think ahead' in order to analyse the implications of certain design decisions both from a functional performance and different life-cycle perspectives. As this is a complex conceptual process, it is a difficult issue to analyse and expand the 'think through' process via a task analysis approach.

Hence, a designer would be unlikely to 'propose' a complete or partial design and immediately be happy with this. Designers seemed to need to reflect on their own actions and critique them before presenting their work to other perspectives to critique. This could be due to designers needing some initial solution to focus on to enable them to progress through the design process. An interesting example of this occurred in case study 4. Designer 4 was principally concerned with the functionality of the designs he worked on. However, he also had an understanding of different life-cycle constraints, one of these being cost. In order to ensure that a new design he was working on could be economically produced in conjunction with his clients complete range of products, commonality of parts was an important design constraint: *"..so you basically had commonality of parts. That was one of the issues that was constantly being assessed - ensure commonality.."*.

Designers would not only pinpoint downstream constraints the design proposal might come into come into conflict with but also the way in which the proposal achieved the functionality required during this reflection stage. However, the actual decomposition of this 'reflect' sub-task is not clearly understood.

Designers would also try and ward off any potentially conflicting critiques by outlining the thinking behind a design before presenting the design to downstream perspectives. Hence an important role output from the 'propose' task is a justification or rationale for the design. The importance of the 'design rationale' and its relationship to the design process is expanded on by Gruber and Russell [1991].

Figure 10.3 outlines the way in which I believe designers propose a design (from the findings of my case studies).

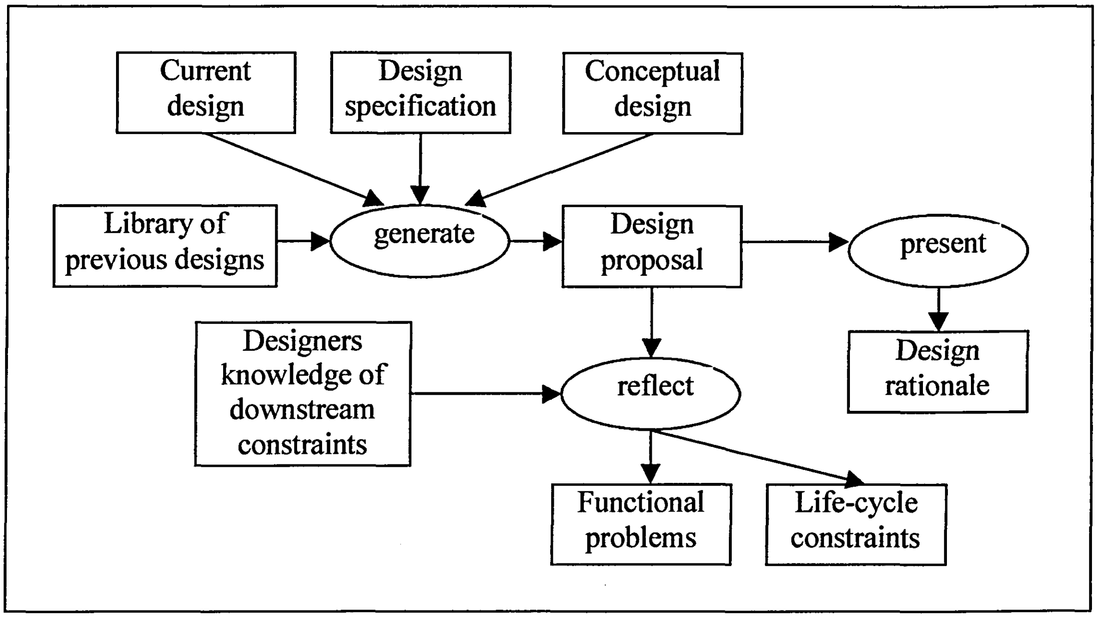


Figure 10.3: Designers and the design proposal task.

This model illustrate how designers will themselves go through a form of internal critiquing where they will first propose a design and then attempt to reflect on any inherent functional problems and life-cycle constraints inherent in their design solution. The recognised life-cycle constraints will be based on the designer's (sometimes limited) knowledge of particular life-cycle issues.

The decomposition of the 'generate' task is dependent on the problem-solving method that is used for the task. One of the most commonly used problem-solving methods exhibited by the designers in the case study was the use of previous designs as the basis for a new design. For example, in case study 1, a new design was very heavily based on a previous design the company had done. However, there had been problems with the previous design and also the functionality required from the new design was subtly different from the previous design.

The designer began by selecting the previous design from an archived database of CAD files. He then split up the complete design into individual components and modified each separate component. He used his experience to tell him when dependencies between the individual components became critical and had to be explored. While modifying each sub-component, the designer would call on downstream perspectives for advice on the design. This process is represented in task form in figure 10.4.

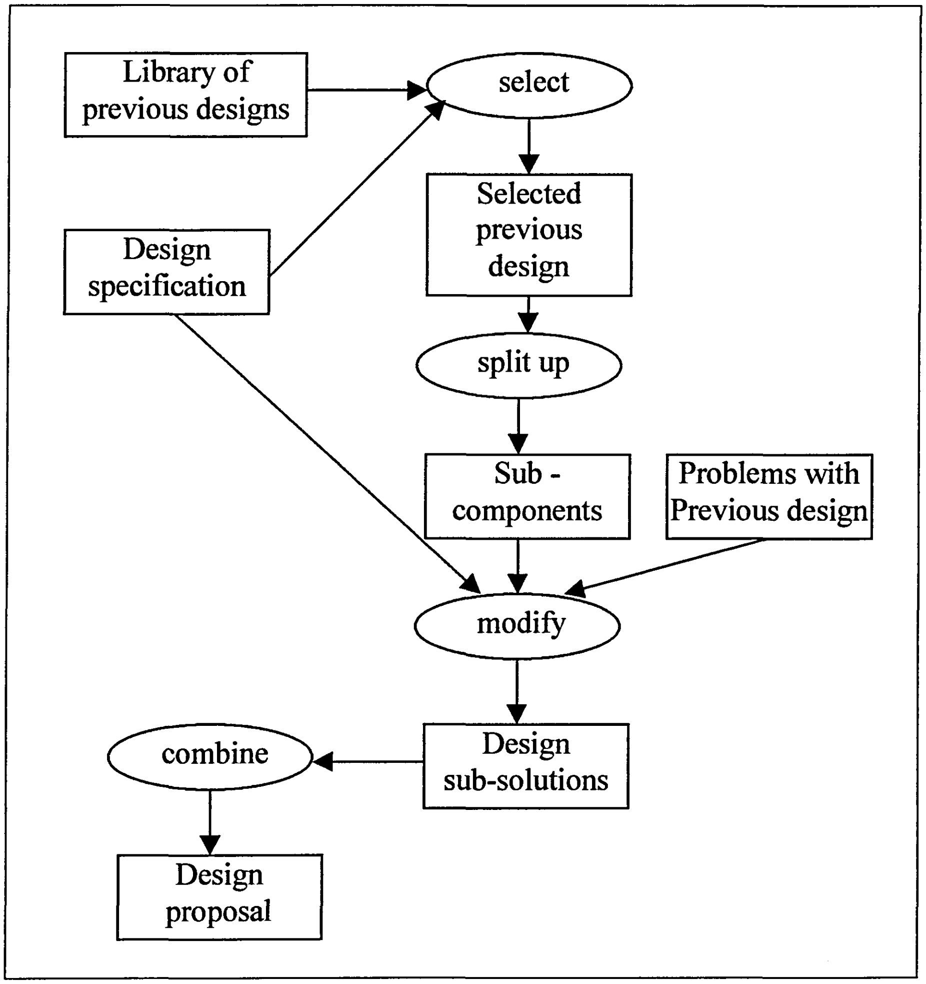


Figure 10.4: Design proposal based on a previous design case.

From the case studies, a common problem-solving method utilised by designers was the retrieval and modification of a previous design solution or case. Problems with the previous design were used to drive the design of the sub-components comprising the new design.

However, while my design case studies indicated designers use different PSM's to accomplish the propose task, it was not clear why they might use a particular method in preference to another. This is represented by *Common*KADS as strategic knowledge about a particular application and is felt to be an area meriting further research.

10.4.2 Critiquing

The critiquing model initially postulated in Chapter 5 is felt to be deficient in a number of respects in the light of my analysis of the case studies. The initial sub-task of the overall critiquing task implied by my initial top-down model is some kind of assessment to determine an 'experts' perspective of the design. The *Common*KADS models for assessment assume the output for an assessment task is some decision class or rank. See Valente and Lockenhoff [1994] for further details. However, the assessment of an external expert assessing a design was not this clear cut. Responses were much more 'fuzzy' ranging from 'this won't do at all' through 'this might be OK' to 'yes, this seems to be OK'. While these responses could be mapped onto some rank or decision class, care would have to be taken with the semantics of the experts' reply to avoid misinterpreting the output from the assessment. Figure 10.5 shows the resulting model for critiquing. The dotted line is intended to indicate that all three 'roles' act as input to the four different sub-tasks.

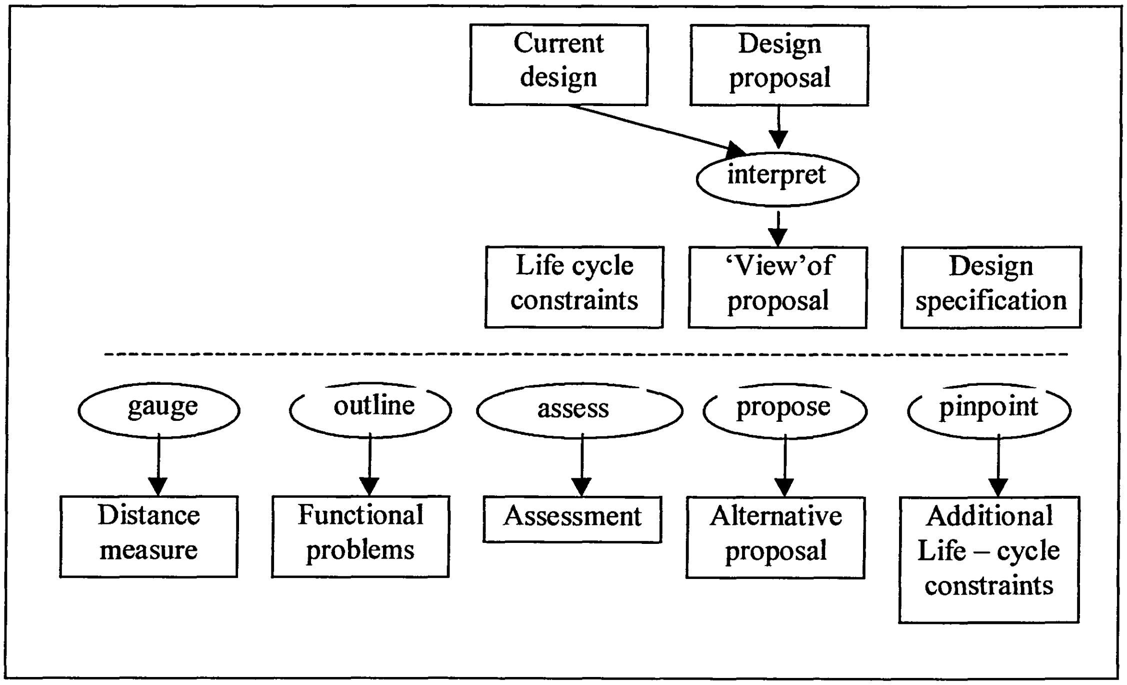


Figure 10.5: Critiquing as a task model.

This expertise task model illustrates the knowledge roles and sub-tasks that were evident in the critiquing task from the case studies. The use of the dotted line departs from the standard *Common*KADS format for expertise task models and is intended to show how the input knowledge roles ('life-cycle constraints', 'view of proposal' and 'design specification') acted as input roles to all the intermediate sub-tasks ('gauge', 'outline', 'assess', 'propose' and 'pinpoint') of the critiquing task while still maintaining the clarity of the model.

10.4.3 Negotiation

Because of the nature of the studies, it has been more difficult to derive a generic task model for the negotiation process. The case studies involved either single designers or relatively small teams of designers. Hence it would be inappropriate to propose a generic model for the task at this stage. However, what became evident from the case studies were some of the negotiating strategies and important knowledge roles input to the negotiation process.

The simplest form of negotiation encountered in the case studies was where the designer was in control of the design process and simply either accepted or rejected any critiques from downstream life-cycle perspectives. Hence if the task is negotiation, this is a solution of the task using a method I will term 'consultation'. The form of the task model for this form of negotiation is given in Figure 10.6.

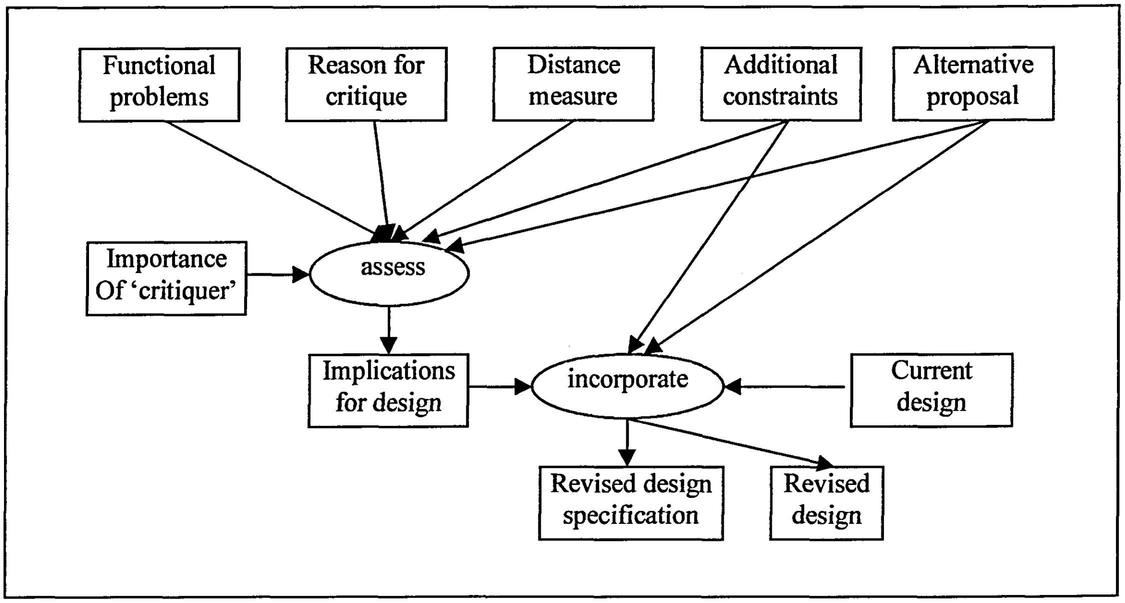


Figure 10.6: Negotiation as a task model (consultation).

This expertise task model illustrates how negotiation can occur when a single 'agent' has a controlling influence over the design process. A particular role input to this process is the relative importance of the agent making a critique of a design proposal. The controlling agent will use this scale of importance to determine whether a critique should be allowed to have an effect on the design.

Appeasement was a widely used strategy. This occurred where a designer would propose a 'toned down' version of the design in response to an unfavourable critique from a downstream perspective. A similar technique was also employed by the downstream perspectives who would themselves offer an alternative proposal to that produced by the designer with some of the features they were not happy with altered. This could lead to complex series of proposal and counter proposal until some agreement is reached. A key aspect of this form of negotiation is the requirement on the part of the agents involved to 'relax' their stance on a particular point in response to a concession from another agent. My proposed model for this form of negotiation is outlined in figure 10.7.

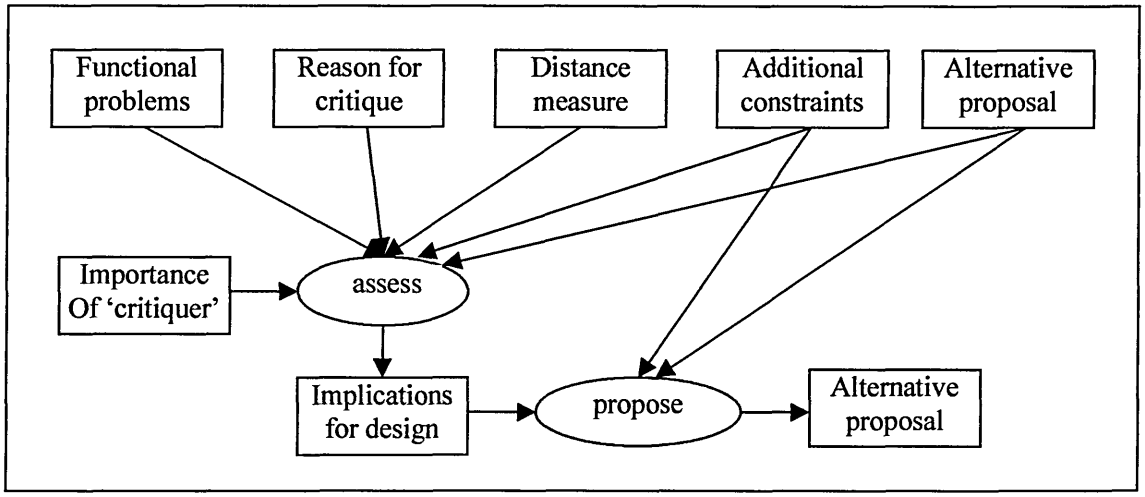


Figure 10.7: Negotiation as a task model (appeasement).

This was one of the more common methods of negotiation encountered. Where an unfavourable critique was delivered by a critiquing agent, the proposing agent would propose a 'toned down' version of the design proposal, which was assumed to be more acceptable to the critiquing agent.

A 'counting votes' form of negotiation was encountered in case study 3. This involved all the interested participants giving their critique of a design and then a system of vote counting was used to determine the most suitable course to take. However, a critical issue with this form of negotiation was the way in which the nature of the design team could critically affect the results – *'...because there's only one marketing person in the team, the decisions get very much an engineering and production bias..'*

Another important issue that came out of the case studies was the type of negotiation that occurred during the concurrent design process. In some cases this would be co-operative in nature. However in other cases, negotiation would be conflicting in nature with radically polarised views being apparent from the downstream perspectives. In some cases, negotiation could not be used to resolve differences between parties. In such cases, the unilateral decision method would be taken by whoever had effective control of the design process.

10.5 A generic model for concurrent design

Using excerpts from interview transcripts and developed task models, it has been shown how different designers and design teams interact during the design process. From the case studies, a more generic model for the propose-critique-negotiate process has been developed by charting ways designers and design teams operate as follows: -

- The designer proposes a solution or a partial solution (the proposal) to the given design problem.

- The designer may then consider the implications of this 'propose' step both from a functional point of view and from the perspective of different life-cycle constraints before presenting the design to different life-cycle perspectives.

- Different downstream aspects can then critique this proposal from their own perspective. The proposal may first need to be represented in a format recognisable to the critiquing agent. The output from a critique can consist of a decision as to whether the design is acceptable or not (this can be seen as a degree of acceptance or distance measure of the design proposal), any problems inherent in the design (the reason for the critique) and a possible alternative solution or sub solution to the given proposal.

- There is then a complex process whereby the design proposal is altered on the basis of these 'design problems'. The case studies generally use a term such as 'alter' or 'modify' to cater for this process. However it is clear that there is some form of negotiation to determine which constraints are allowed to influence the 'alter' or 'modify' process. It is believed the person with control of the design then considers which of these problems can be allowed to affect the design by considering the key aspects of the design as an important input role. This results in a re-formulated design specification with additional life – cycle constraints made explicit.

The design can then be altered on the basis of the outlined problems. This is represented in task model form in figure 10.8.

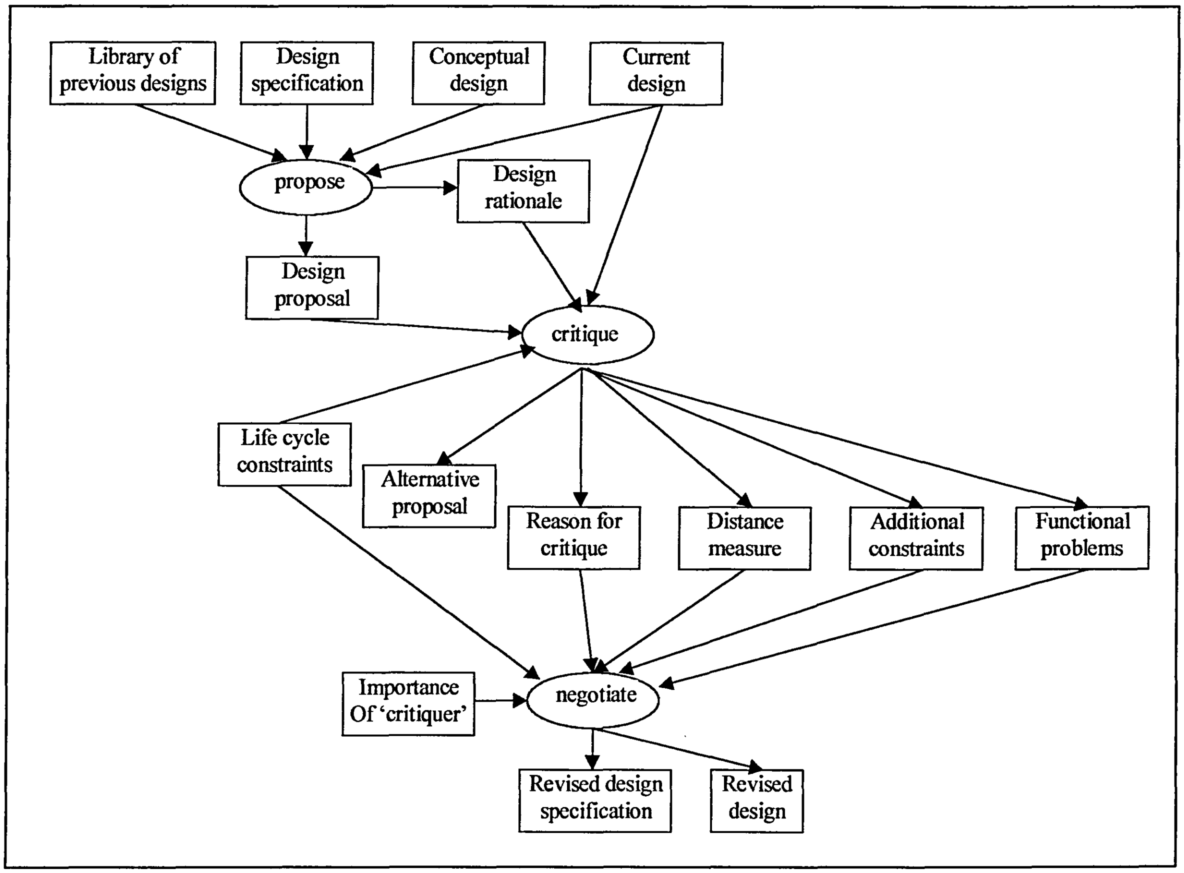


Figure 10.8: A generic model for propose-critique–negotiate.

This model is based on earlier models for propose, critique and negotiate presented earlier in this chapter. These models were derived from analysis of my different case studies. In terms of knowledge roles, 'current design' and 'revised design' and 'design specification' and 'revised design specification' are effectively the same roles. The differing terminology is used to clarify the models.

The terminology used in figure 10.8 is designed to encompass the terminology derived from the different case studies. I.e. 'propose' encompasses 'develop', 'generate' etc., 'critique' encompasses 'assess', 'evaluate' etc.

It is also important to consider the role that a 'modify' task may play in concurrent design. I have modelled the 'propose' task incorporated in concurrent design as consisting of the generation of a 'design proposal' which advances the state of the 'current design'. This 'design proposal' is then presented or made open to criticism by different life-cycle perspectives in the context of the 'current design'. When the additional constraints have been derived from the negotiate task, the designer then considers these constraints in the next proposal step.

An alternative viewpoint would be to consider only the current design and assume that this as a whole is advanced in some way by the 'propose' task. The purpose of the critiquing task would then be to alter or modify this current design on the basis of the given critiques. However, I believe that the difference between these two approaches is essentially a sequencing issue and 'modify' as a task is essentially the same as 'propose'.

When a designer presents a proposal to their 'critics', the design rationale is used to justify the design decisions the designer has made. This may include some history of the design process utilised by the designer. The importance of the history of the design process as a design rationale has been noted by Bañares-Alcántara [1995] in a study of the design of chemical plants.

It must also be noted that by looking in more detail at the (sub) tasks of propose, critique and negotiate that additional knowledge-roles, that expand on those shown in Figure 10.2, play an important role in concurrent design. As I consider the 'conceptual design' and 'detail design' tasks outlined in Figure 10.2 as being achieved via the application of a problem-solving method comprising the propose, critique and negotiate tasks, these knowledge roles effectively act to expand on those roles shown in figure 10.2.

10.6 Other issues from the case studies

It is also relevant to consider other issues, which became apparent during analysis of the case studies.

10.6.1 Component and functional decomposition of the design problem

In general, particularly at later stages of the design process, the overall design task would be split up into more manageable 'chunks'. This would typically consist of different subassemblies or sub-functions comprising the complete solution being developed in relative isolation from each other. However the dangers of forgetting the dependencies between different components or functions are obvious, as illustrated by an example from case study 4.

"The vehicle was designed from the scratch to accommodate two sizes of wheel so that the rear end and possibly the floor pan could be used for - certainly the rear wheel - could be used for a larger vehicle - and the information I had on the tyre sizes was nominal - it wasn't accurate - and so when they came to build the larger vehicle they couldn't get the tyres on."

10.6.2 Generic constraints from the case studies

A number of different life-cycle constraints were evident from the case studies. The different concurrent constraints exhibited by the case studies included manufacture, assembly, cost, electronics and reliability considerations.

As well as the life-cycle constraints implied by a concurrent approach, another problem evident in all the case studies was the fundamentally constraining effect of time-scales on designers and design teams. The effect of time constraints has a significant effect on the process.

A critical factor in determining whether an idea for a design is feasible could be the time-schedules imposed on the product development and manufacture. Figure 10.9 models designer 5 presenting an idea to a client. The client then assesses the idea using time-scales as a constraining factor to dictate their response and a revised time–schedule for the designer to work with.

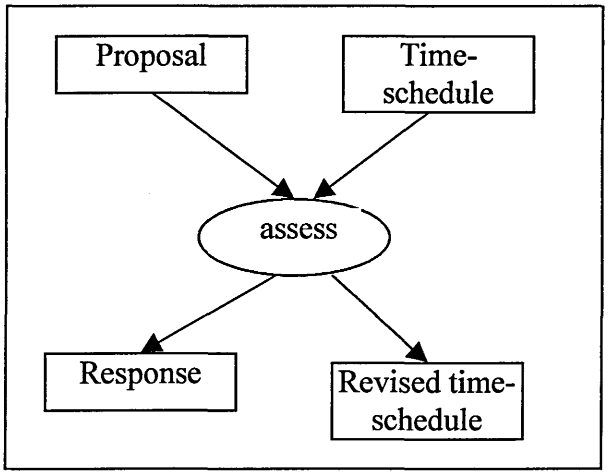


Figure 10.9: The constraining effect of time-scale from case study 4.

This task model illustrates how constraining time-schedules, in the form of a project plan or a particular deadline can affect the feasibility of a particular proposal. The response output from the task will indicate whether the proposal is feasible based on the time-scales while a revised time-schedule may be generated.

The time-scales on product manufacture can also have a decisive effect on which downstream processes can be utilised in the design as shown by figure 10.10. In this example from case study 4, the limiting time schedule available for design and manufacture effectively narrowed down the choice of production processes available to the designers.

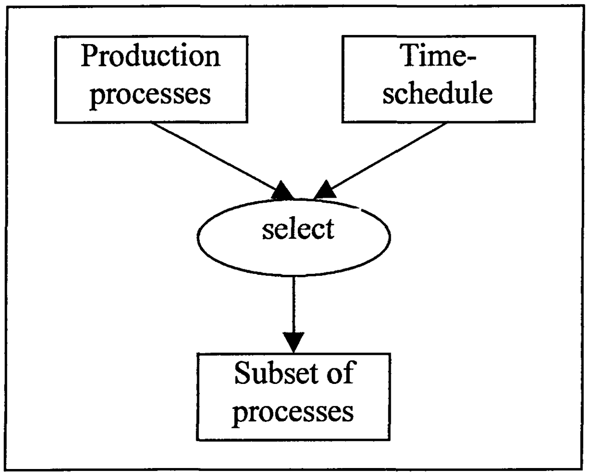


Figure 10.10: Time-scales dictating downstream processes from case study 4.

Given an initial set of possible production processes to utilise for a design, this set of processes can be reduced by the constraining time-scale, which dictates when the implemented design must be delivered.

However design time-scales, like most constraints, are themselves open to review. In case study 4, the time program for the design process was constantly being updated in order to allow less constraining time-scales on design time.

10.6.3 Visualisation and terminology

Analysis of transcripts and the resulting task models suggest that an interim interpretation process is sometimes necessary in order to allow different perspectives to relate to the evolving design.

Designer 4 worked with an outside supplier for the mouldings required. The model he had developed as a 3D CAD model had to be interpreted as sections across the model in order for the moulder to be able to understand it. The moulder was used to being supplied with general assembly (GA) drawings to work with and when asked to comment on the 3D CAD model said:

"yes, OK but when will the GA drawings be available for us to comment on ?"

When the GA drawings were produced from the CAD model, the outside supplier pinpointed a number of problems which could have been sorted out earlier if they were able to relate to the CAD model or if an acceptable 'view' was available to them.

Once a downstream perspective can relate to the design, they are then in a position to be able to comment on or critique the design. By Designer 4 taking hard copies of the CAD model, potential customers were able to relate to the design and give their reaction as shown in figure 10.11.

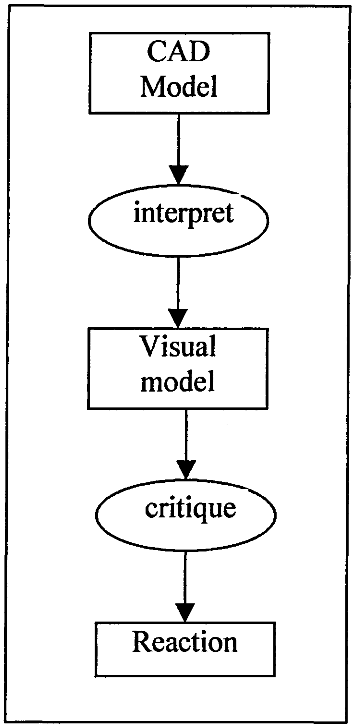


Figure 10.11: Interpret and critique from case study 4.

This expertise task model illustrates how a critiquing agent must first be able to interpret his or her own 'view' of a design before being able to generate a critique. The generalised role 'reaction' is used here to encompass all the roles I have previously defined as being output from the critiquing task.

The most commonly encountered means of enabling different perspectives to form 'views' of a design were the generation of physical mock-ups. The importance of building such prototypes was particularly emphasised in case studies 3 and 4.

Case study 4 also illustrated the problems that can occur when different perspectives use conflicting terminology to describe aspects of the design. Outside suppliers that Designer 4 was working with had an entirely different concept of the term 'part line' to the designer (and in fact the rest of the world as the designer was using generally accepted terminology). The outside supplier continually tried to get the designer to adopt their terminology even though it was incorrect as shown in figure 10.12.

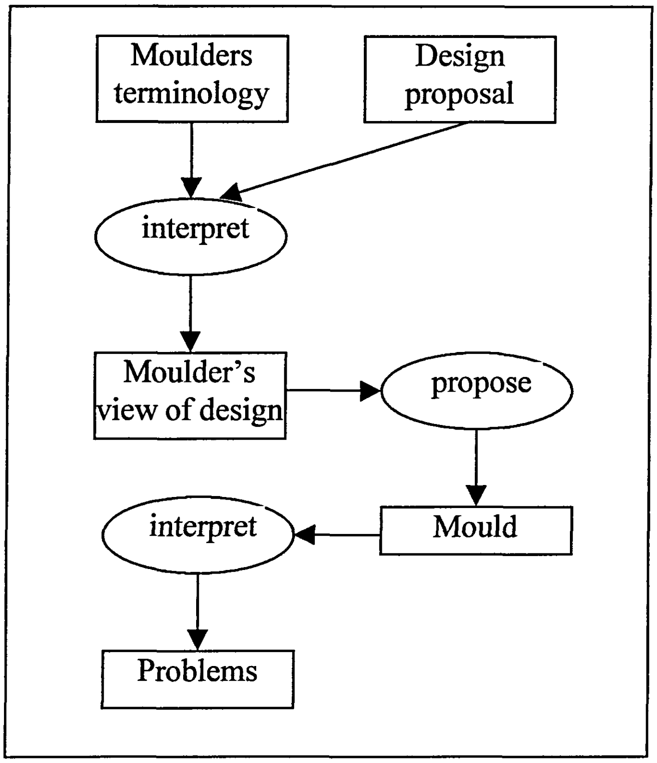


Figure 10.12: Problems with terminology from case study 4.

This task model illustrates how different agents (involved in the design process) use of language can cause problems. This shows how an outside supplier (the moulder) interprets a design proposal from a designer (in the form of a CAD drawing) incorrectly. The 'mould' that is then proposed (or produced) by the moulder is incorrect (and causes problems) when re-interpreted by the designer.

This problem of 'views' was also evident when dealing with other areas of expertise. Designer 4 found that marketing personnel could relate to 3-D models while manufacturing experts, while able to relate to such a model, also needed to see details of mechanical connections before feeling 'comfortable' with the design. This problem with different 'views' is linked with the semantics and ontology's that different 'experts' utilise. The industrial designers talked in terms of 'form' and 'shape', while manufacturing and assembly personnel utilise different means of describing processes and objects.

Because of the inherently multi-disciplined nature of concurrent design, different views of the evolving design will be required by the different participants. Various researchers have outlined the need for different views of a design. In particular, based on their efforts to develop a computer – based design support system, Finger et al [1992] suggest that from an implementational viewpoint *"Perspectives may create local representations for reasoning and analysis but communication is always through the shared representation"*.

Continuing interest in this area of different views of 'the design world' extend to the field of ontological engineering. Ontologies specify a *"conceptualisation": a way to view the world. Every ontology thus incorporates a particular viewpoint"*. (Wielinga and Schreiber [1998]). In Figure 10.5, I have specified a task 'interpret' whereby an agent forms his or her own view of a design. However, I would not claim to be able to describe the complex cognitive processes that must occur for this transformation. I believe this is an area worthy of further research.

10.6.4 Modelling

Visual modelling is an important method used by designers to transform their thought into more concrete form. Visual modelling methods included computer modelling (both 2-D and 3-D) to represent and allow more effective 'views' of the evolving design and mechanical prototyping was also used to allow a more effective 'view' of a design. In fact, Designer 4 would now consider mechanical prototyping a crucial part of the design process after his experience with the case study.

10.7 Discussion

A number of general findings and conclusions can be drawn from the preceding analysis of transcripts and excerpts from developed task models.

Different life-cycle constraints do impinge on the design process. Typically, personnel with expertise in different life-cycle perspectives will analyse an evolving design and outline any potential problem areas. They may also propose different solutions or sub solutions, which make the design more acceptable from their own perspective.

In the case studies analysed, the designer generally had control of the design and dictated which external constraints were allowed to affect the design. The designers used a number of different arbitrating strategies to achieve this. An exception to this occurred in case study 4 where designer 5 offered a revised design solution to his management team but they decided not to implement the new design.

Who has control of the design is a crucial issue when determining which concurrent constraints are allowed to influence the evolving design. In some of the case studies, the designers themselves would effectively have the final say in this matter, However in other studies, it was the management of the organisation who have the decisive influence. The crucial effect of whom has control of the design process and the crucial effect of their views on the eventual design solution can be illustrated by quotes from some of the designers in the case studies:

"you find you're compromising and they'll walk all over you" – Designer 5

"...if the reasons for doing it are powerful enough from the users' point of view, from the marketing point of view ... " – Designer 3.

My findings from the different case studies would suggest that a design will never be optimal from all life-cycle perspectives, some perspective will always take precedence. The controlling influence on the design process will typically dictate which perspective takes precedence. The implications of this not happening are suggested by a quote from Designer 3: *"Ok its been sorted out from all sorts of points of view, timescales, production and what have you but the product is so bloody awful that there's no point proceeding".*

The context of the design situation affected the type of collaboration between designers and life-cycle perspectives. In case studies 1 and 2, the designer worked for the organisation in question. In this case, the designer, manufacturing personnel, assembly personnel etc. were more clearly working towards a common goal. However, in case studies 3 and 4, the designers were more independent and this affected their interactions with these perspectives. Designers 3,4 and 5 were more protective of their designs and only allowed life-cycle perspectives to affect the design if they really had to. This was a more conflicting model of collaboration between designer and life-cycle perspectives than that shown by case studies 1 and 2.

Different perspectives involved in the design process need to be able to relate to the design before being able to sensibly comment on the implications of the design from that perspective. Different terminology's can be utilised by these perspectives to describe the design. Time-scales are also a crucially limiting constraint on designers, limiting both the amount of design time they have as well as limiting the actual form of the design.

10.8 Chapter Summary

In this chapter I have outlined some of the key features and characteristics of the concurrent design task, as practiced by a number of organisations I have analysed in different case studies. These are based on design tasks used by practising designers in a number of small to medium sized enterprises engaged in concurrent product design. A generic task structure for the overall design process and a task structure for the concurrent consideration of multi-functional perspectives during the design process, have been presented. The process of concurrent design is believed to differ critically from more co-operative modes of design because of the conflicting goals of the different perspectives involved in the process.

I conclude that during the process of design, designers use a number of different problem-solving methods to achieve the two distinct sub-tasks within design-synthesis (these are concept generation and detail design) and a number of other sub-tasks within the design process. Concurrent critiquing of the evolving design plays an important part in influencing the evolving design, especially in the later stage of design. Hence, knowledge-based support for the concurrent design process could make use of these and other design models in order to more fully support designers and design teams.

In the next chapter, I discuss the instantiation of these generic models using the *Common*KADS workbench. While it is the expertise model of the *Common*KADS model set that I have concentrated on up to this point, the next chapter looks at how this model is integrated with the other models comprising the model set. In particular, I look at the organisational context in which such models may be implemented via the *Common*KADS organisational model.

11 Model templates for concurrent design

11.1 Introduction

I have developed a suite of knowledge-level task models for concurrent design and its associated sub-tasks using the semi-formalisms of the *Common*KADS methodology. These have been derived from case studies, which analysed engineering designers. These task models have been used to instantiate a 'generic' task model template in the *Common*KADS model set.

One of the main motivations of the *Common*KADS methodology is that generic models can be included as part of a taxonomic library, where they act as templates for developers of knowledge-based systems. *Common*KADS advocates the building of a series of complementary models during the development of a KBS. The thesis up until this point has focussed on the expertise model. The different models comprising the complete *Common*KADS model set are described in de Hoog et al [1993(a)]. They are:

- Expertise model (Breuker and Van de Welde [1994])
- Agent model (Waern and Gala [1993])
- Organisational model (de Hoog et al [1993(b)])
- Task model (Duursma et al [1994])
- Communication model (Waern et al [1993])
- Design model (Van de Welde et al [1994])

The *Common*KADS workbench (Toussaint et al [1994]) is a research and re-use tool, which implicitly supports *Common*KADS model development and the refinement of existing templates for the different *Common*KADS models. Depending on the scope of the intended application, different *Common*KADS model templates can be instantiated to different degrees. I have taken the existing *Common*KADS model templates and instantiated them with what I have identified as generic features, for the purposes of supporting concurrent design within a range of different organisations.

However, while I have developed my generic model templates using a *Common*KADS workbench and the formalisms of *Common*KADS, the intention is that the templates will be useful to all developers of KBS to support concurrent design, not just those who are using *Common*KADS in conjunction with the workbench. To this end, I have designed my models to be as transparent as possible while still adhering to the formalisms of *Common*KADS. In this way, the different model templates can give guidance to developers on the types of issues and knowledge that it may be necessary to model in order to develop KBS support for concurrent design.

This chapter gives an account of the development of my model templates for concurrent design and outlines some of the more pertinent points relevant to the development of concurrent design support systems. First I look at the set of models comprising the full *Common*KADS model set. Then, I analyse in detail the development of my model templates for concurrent design for each of the *Common*KADS models.

11.2 The *Common*KADS model set

The *Common*KADS product, which is taken to be the final project deliverable to a client, consists of a number of different models at an agreed level of development and refinement. These models complement and expand on the models that comprised earlier KADS model sets (Tansley and Hayball [1993]). The summary presented in this chapter is based on an account in Breuker and Van de Welde [1994] and de Hoog et al [1993(a)].

An implicit assumption within the *Common*KADS methodology is that the driving force behind the development of a knowledge-based system is the need to implement a KBS in some organisation or enterprise. Hence the different models incorporated in *Common*KADS reflect this. The *Common*KADS model set supplies template models which need to be instantiated with details of the specific case for which a KBS is being developed. My approach has been to take these template models and further refine them to support the process of concurrent design system development.

It is important to note that for different projects, certain models will be more important than others. This is supported by de Hoog et al [1993(a)] *"For each individual project, certain elements of the CommonKADS model template are more important than others"*. Not all models need to be developed to the same level of detail, depending on the particular characteristics of the problem being considered, as outlined by Van de Welde [1994]. *"...for any particular project, the models in the model set may be developed to a different extent or even not at all"*.

In addition, a *Common*KADS project does not necessarily have to result in the development of a complete system. De Hoog et al [1993(a)] suggest that *"A CommonKADS project that does not deliver an implemented KBS can still deliver a precise design to the client"*. In fact, because of issues regarding feasibility, cost, human factors etc, a *Common*KADS product may never be completely finished. AI and KBS projects have a high risk of non-completion in comparison to more routine information systems projects. Interestingly, it is important to ensure that non-completion is not equated with failure. There is strong evidence that the intermediate outcomes are at least as valuable to the client organisation as the intended project goal (Akkermans et al [1999], Speel et al [1999]).

I will now discuss how projects are managed within *Common*KADS and the different models comprising the full model set.

11.2.1 Project management in *Common*KADS

Project management within *Common*KADS is based on Boehm's influential spiral model (see Boehm [1988]) and utilises a process of review, assess risk, plan and monitor. This is illustrated in Figure 11.1. In the development of a real-life KBS, with large numbers of different people working on different aspects of the project at different times, the project management issues would be critically important. However, because of the scale of my research, in terms of the number of people involved, they were not so important.

In order to facilitate the project management aspects of KBS development, *Common*KADS models evolve through different states with each completed state being a landmark in that model's development. Because of dependencies between the different models, some models have to be developed to a certain state before other models are developed. For instance, the state 'tasks assigned to agents' in the task model cannot be reached before the state 'agents involved identified' has been reached in the agent model. Where this is pertinent to my model development, these states will be noted. The development of each model is managed separately (uneven and combined development) as needed / justified.

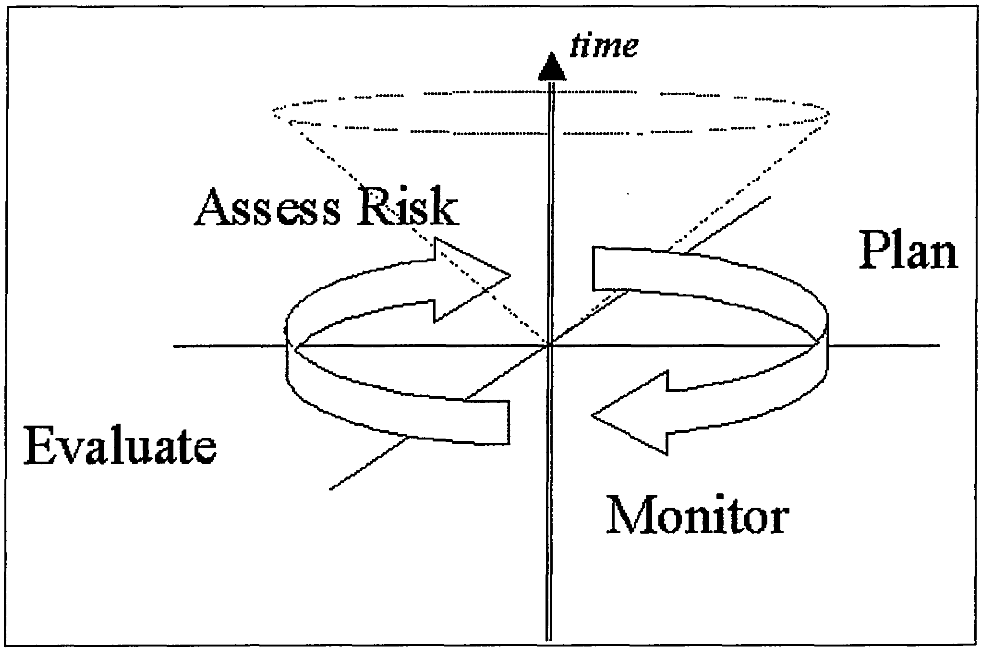


Figure 11.1: Project management and in *Common*KADS.
Project management and control of model development is based on a spiral model of plan, monitor, evaluate and assess risk.

11.2.2 Links between models in the *Common*KADS model set

In order to illustrate the dependencies between different models, Figure 11.2 (adapted from de Hoog et al [1993(a)] shows the 'links' between the different models.

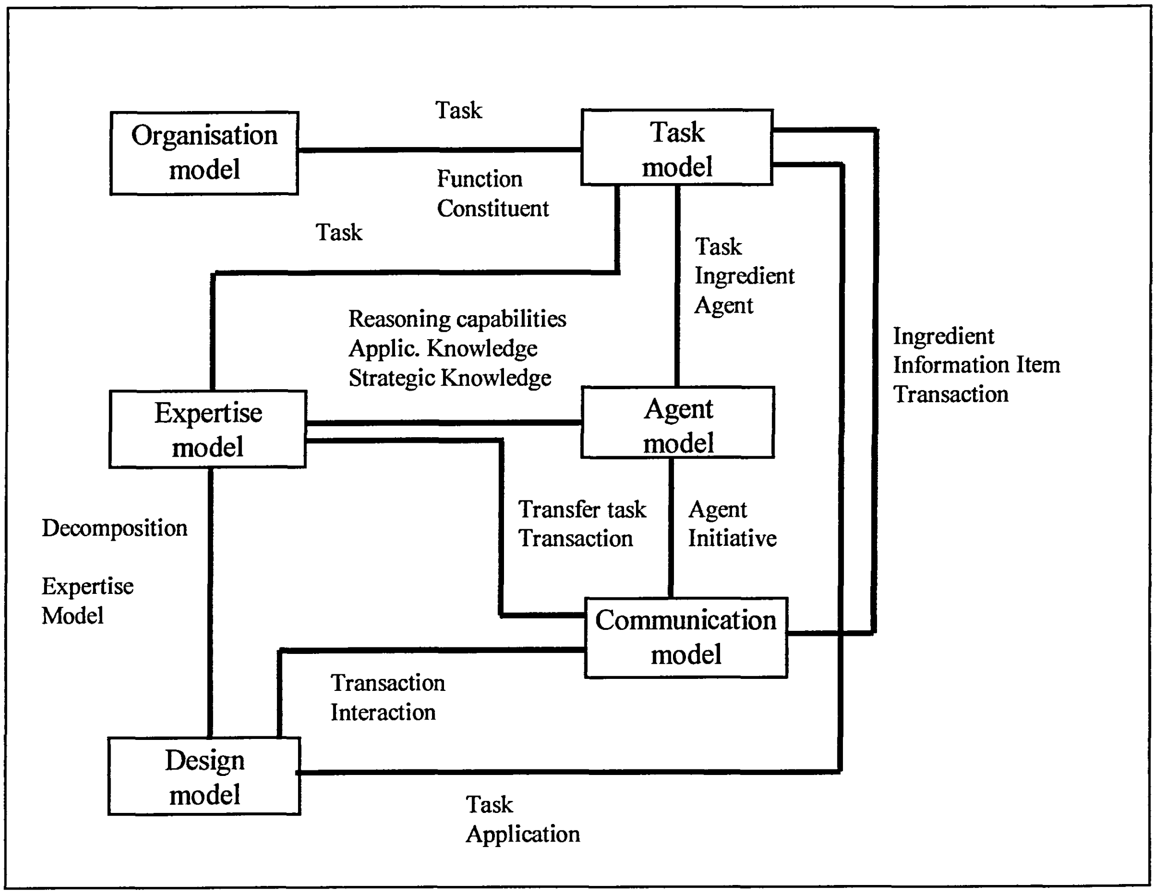


Figure 11.2: Links between *Common*KADS models.

This diagram, adapted from de Hoog et al [1993(a)], illustrates the links and dependencies between the different models comprising the full *Common*KADS model set.

A number of different software tools currently support the development of *Common*KADS models, see Appendix C. I have utilised the *Common*KADS workbench to support the development of my models for concurrent design.

11.3 The *Common*KADS model set and my generic model templates for concurrent design

In order to provide support for potential developers of KBS for the concurrent design process, I have instantiated a number of the different template models comprising the *Common*KADS model set on a *Common*KADS workbench. These templates have been refined using information derived from my analysis of the different case studies described in chapters 8 – 10 and other referenced research. The templates have been instantiated with what I consider to be the generic aspects of the different models. While my research has mainly focused on the development of expertise task models utilised during concurrent design, I have also instantiated other model templates comprising the full model set with what I would consider to be generic aspects of concurrent design. The intention is that my developed generic model templates will act as useful support for developers of KBS to support concurrent design.

I begin by discussing the existing *Common*KADS model templates and how I have instantiated these for the concurrent design process. Where appropriate, I have utilised screen-shots from the *Common*KADS workbench to illustrate my models. However, where these do not show sufficient detail, I have outlined the information in tabular format.

11.4 *Common*KADS ORGANISATION MODEL

This models the target organisation within which the developed KBS is to be situated. To this end, the organisation model supports three main functions. These are the identification of promising areas of KBS application within the organisation; the identification of possible impacts of the KBS on the organisation and finally it should also provide information for some crucial issues in other models (such as the task distribution between agents).

The development of an organisation model clearly implies modelling the processes and functions within an organisation. Because of this, a number of examples have come to light where the development of an organisation model within the *Common*KADS framework has given valuable insights into an organisation's structure (de Hoog et al [1992], de Hoog et al [1993(b)].).

In order to fulfil it's defined role, the organisation model is split into two major components. The global constituents are described only once, irrespective of any possible solution that may be the immediate focus of attention. The global constituents include the problems and opportunities for KBS implementation within an organisation, the organisational context within which a solution may be implemented and the list of possible solutions that may be considered. The variant constituents characterise different aspects of the organisation and are used to address the implementations of different possible solutions. These variant constituents are:

- People
- Knowledge,
- Functions (such as logistics and research),
- Structures (in terms of departmental structures),
- Processes (e.g. production processes etc),
- Power relations
- Resources (both computing and other) that exist within the organisation.

The development of an organisational model (where the goal is to provide KBS support for the organisation) involves tailoring the organisation model template provided by *Common*KADS to the requirements of the given organisation, where the current template offers nothing other then the headings. My aim is slightly different in that I wish to model generic aspects of a number of different organisations that practice concurrent design and would benefit from KBS support for this process. As a result, my organisation model template provides a super-set of aspects, typical of organisations that practice concurrent design. For the purposes of my generic organisational model, I am only considering one possible solution, which is my generic model template.

11.4.1 Global constituents of my generic organisational model

I will begin by outlining the global constituents comprising my generic organisational model.

11.4.1.1 Problems and opportunities

From my case studies, I have determined a number of generic opportunities inherent in an implementation of KBS support for concurrent design, together with potential problems such an implementation may incur. These are outlined in Table 11.1[1]

[1 1] The *Common*KADS workbench I have used to develop my generic models is implemented in Prolog. As a result, entity names are not allowed to have 'spaces' in them. The naming convention I have used specifies the type of entity and then the name of the entity, connected by an underscore. E.g. the organisation opportunity Loss_of_expertise etc.

name	description
Opportunities	
Loss_of_expertise	In any organisation, there is a possibility of human-based expertise being lost to the organisation. A KBS solution can alleviate this problem.
Cost_reduction	Implementing KBS solutions to concurrent design within an organisation can result in long-term cost savings
Improved_design_quality	By bringing life-cycle perspectives to bear on the design process, design quality can be improved.
Reduced_design_time	By bringing life-cycle perspective knowledge to bear on the design process, the overall design time can be reduced.
Reduced_life_cycle_costs	By bringing life-cycle perspective knowledge to bear on the design process, the overall life-cycle costs of the design can be reduced.
Problems	
Knowledge_elicitation	Eliciting knowledge from experts for use in such KBS based systems may pose problems, especially if the expert perceives such a system as a threat.

Table 11.1: The generic problems and opportunities inherent in implementing KBS support for concurrent design within an organisation

11.4.1.2 Organisational context

The generic organisational context within which KBS support for concurrent design may be implemented are outlined in Table 11.2. These contexts refer to factors in the target organisation that can either influence the impact of the KBS on the organisation or the opportunity for KBS technology.

name	description
Fear of KBS supplanting human expertise	Existing personnel are likely to be afraid of computers supplanting their roles and expertise within the organisation
Fear of computers controlling processes	Personnel will usually be afraid to allow control of key business processes to be performed by machines
Risks of changing dynamics	Organisations will already have existing design processes in place. Altering these processes can have a detrimental effect on the successful application of the process
The existence of project champions	This will typically be the chief designer or project manager who has a clear 'vision' of the requirements of the product.
Clear and shared design perspectives	The perception, from a design perspective, of a particular project within the organisation by different personnel can have a key impact on the successful outcome of the project
Clear and shared business perspectives	The perception, from a business perspective, of a particular project within the organisation by different personnel can have a key impact on the successful outcome of the project

Table 11.2: Generic factors within an organisation that can affect the successful implementation of KBS solutions

11.4.1.3 Solutions

The different ways in which KBS support may be implemented within a target organisation are modelled within the Solutions. Each solution is characterised by different variant constituents. I have considered the 'present' solution as being the generic solution and the variant constituents are modelled for this present solution alone.

11.4.2 Variant constituents

These constituents outline the constituents of an organisational model and are related to a particular solution. In my model, I have modelled the different variant constituents for the present or generic solution.

11.4.2.1 People

I have outlined a number of different people who generically play a part in the concurrent design process. Each 'people' constituent is characterised as a stakeholder and has a number of different properties. These are a name and a description. In addition, each stakeholder is linked to an instance of agent in the agent model.

Each stakeholder may also be linked to either a role or an actual person in an organisation. For the generic model, I am not referencing any particular organisational structure, hence each stakeholder is linked to different roles. However, in an actual instantiation, each stakeholder would represent an actual person or KBS agent and hence could be associated with a name.

I have considered four different roles to which stakeholders may be linked. These are the control, propose, critique and negotiate roles. These represent the distinct roles I have identified in the concurrent design process via my case studies.

The generic stakeholders in my model are:

- Control_agent
- Design_agent
- Assembly_critiquing_agent
- Manufacture_critiquing_agent
- Materials_critiquing_agent
- Cost_critiquing_agent
- Environmental_critiquing_agent
- Electronics_critiquing_agent

It must be emphasised that these represent typical agents that may be found in an organisation practicing concurrent design. The list is not intended to be exhaustive and is intended to be extensible by other researchers. Experiences from my case studies indicate that the controlling agent in the design process will typically be either the designer or the project manager in charge of the overall design project. . In addition, each stakeholder may play a number of different roles. For example in case study 4, designer 5 effectively had roles 'control' (as project manager) and 'critique', as he was also production director, with extensive knowledge of the companies manufacturing and assembly capabilities. The properties of each agent are outlined in table 11.3.

name	Properties
Control_agent	**Description:** Agent with a controlling influence on the design process
	links to agent: age_Control
	requires_knowledge: Control_knowledge
	Role: Control
Assembly_critiquing_agent	**Description:** Agent with the ability to critique designs from an assembly perspective
	links to agent: age_Critique_Assembly
	requires_knowledge: Assembly_critiquing_knowledge
	Role: Critique
Manufacture_critiquing_agent	**Description:** Agent with the ability to critique designs from a manufacture perspective
	links to agent: age_Critique_Manufacture
	requires_knowledge: Manufacture_critiquing_knowledge
	Role: Critique
Materials_critiquing_agent	**Description:** Agent with the ability to critique designs from a materials perspective
	links to agent: age_Critique_Materials
	requires_knowledge: Materials_critiquing_knowledge
	Role: Critique
Electronics_critiquing_agent	**Description:** Agent with the ability to critique designs from an electronics perspective
	links to agent: age_Critique_Electronics
	requires_knowledge: Electronics_critiquing_knowledge
	Role: Critique
Cost_critiquing_agent	**Description:** Agent with the ability to critique designs from a cost perspective
	links to agent: age_Critique_Cost
	requires_knowledge: Cost_critiquing_knowledge
	Role: Critique
Design_agent	**Description:** Agent with the ability to propose designs.
	links to agent: age_Design
	requires_knowledge: Proposal_knowledge
	Role: Propose

Table 11.3: The properties of the stakeholders in the generic organisational model

11.4.2.2 Knowledge items

Based on my case studies, I have outlined a number of knowledge categories that are of importance in the organisation and can affect the feasibility of particular solutions. A number of different generic knowledge categories are modelled. These are:

- Design_knowledge
- Assembly_critiquing_knowledge
- Manufacture_critiquing_knowledge
- Materials_critiquing_knowledge
- Electronics_critiquing_knowledge
- Cost_critiquing_knowledge
- Environmental_critiquing_knowledge

Each knowledge category has a number of characteristics. As an example, the knowledge category assembly_critiquing_knowledge is outlined in Table 11.4.

name	Assembly_critiquing_knowledge
description	Knowledge about critiquing from an assembly perspective
Role description	Critique
Real-life task	Assembly life-cycle critiquing
Domain	Generic
Task type	Critiquing
Functions	fnc_Assembly_Critique

Table 11.4: The Assemly_critiquing_knowledge knowledge item.
This represents the generic characteristics of the knowledge pertaining to the life-cycle critiquing from an assembly perspective that exists in an organisation. Only the characteristics of this item that are believed to be generic have been instantiated.

For a particular instantiation, the Domain of application of the knowledge will be defined. The actual knowledge itself is modelled in the domain layer of the expertise model for the instantiation. Typical assembly knowledge includes the need to reduce part counts to minimise assembly time.

11.4.2.3 Computing resources

The computing resources available to an organisation describe either the actual computer resources (hardware) or logical computing resources. Based on my case studies, I have identified a number of logical computing resources that are likely to be encountered within organisations practicing concurrent design. These are outlined in Table 11.5.

name	description
CAD Systems	All of the organisations in the case studies utilise CAD systems to facilitate the concurrent design process
Graphics systems	Many of the organisations in the case studies utilise graphics packages, especially in the early stages of the design process
Server based networks	Many organisations will have an existing IT infrastructure based on a central server attached to a network
ERP (Enterprise Resource Planning)	Many organisations use ERP (Enterprise Resource Planning) systems to control and manage the organisations resources. Any KBS solution may be required to interface with such a system.
Relational database management system	Existing organisational data may be contained in a RDBMS (Relational database management system).

Table 11.5: Some generic computing resources that may be found in organisations that practice concurrent design.

11.4.2.4 Other resources

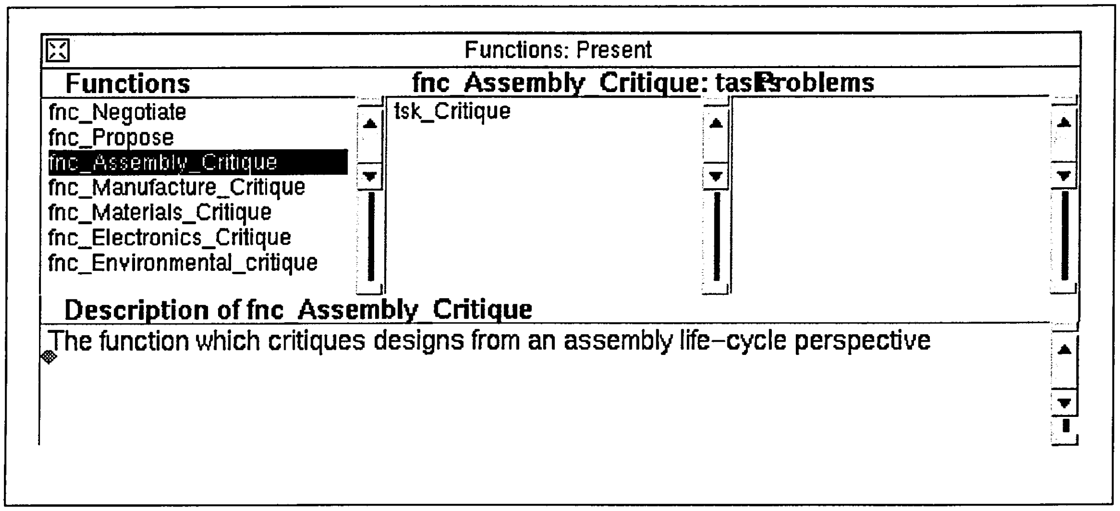
This models any other resources available to the organisation utilised within the concurrent design process. From my case studies, a universally encountered resource utilised during the design process is a repository (or case library) of previous designs. These were physically manifest in different ways, typically as drawings (either paper or CAD based). My case study organisations were found to make extensive use of these resources when developing new designs.

11.4.2.5 Functions

The functions present in the organisation define what tasks are carried out in the organisation. As a result, there is a direct mapping between tasks in the task model and functions in the organisation model. Based on my case studies, I have defined a number of generic functions, which I believe are pertinent to concurrent design.
These are:

- fnc_Negotiate
- fnc_Propose
- fnc_Assembly_Critique
- fnc_Manufacture_Critique
- fnc_Materials_Critique
- fnc_Electronics_Critique
- fnc_Cost_Critique
- fnc_Environmental_Critique

The characteristics of fnc_Assembly_Crtitque are outlined in table 11.6.


Functions: Present
Functions
fnc Assembly Critique: tasks
Problems
Description of fnc Assembly Critique

fnc_Negotiate
fnc_Propose
fnc_Assembly_Critique
fnc_Manufacture_Critique
fnc_Materials_Critique
fnc_Electronics_Critique
fnc_Environmental_critique

tsk_Critique

The function which critiques designs from an assembly life-cycle perspective

Table 11.6: The function fnc_ Assembly_Critique that critiques designs from an assembly perspective.
This table describes how the critiquing function performs a critiquing task from an assembly life-cycle perspective.

11.4.2.6 Processes

Any organisation practicing concurrent design will utilise a large number of different processes to achieve their goals. As an example, the company in case study 1 has identified 5 key processes that facilitate their business goals. These are new product development, personnel and IT processes, customer support, order acquisition and order satisfaction. From the perspective of concurrent design, the important process is new product development. Hence I have identified a generic process, which I term the design_process, common to all my case studies, which is the process whereby new designs are generated. The design processes carried out in the organisation are accomplished by the different functions present.

When discussing processes in an engineering context, a number of different manufacturing, assembly etc processes were identified as being generic from my case studies. Examples include welding, vacuum forming and manual assembly processes. In case study 2, these can be seen as sub-processes of the 'order satisfaction' process. From the perspective of concurrent design within an organisation, I have not documented all the different manufacturing etc processes that occur within the organisation. However, what is critically important is that knowledge of these downstream processes are available as input, in the form of knowledge, to the design process.

It must also be noted that many processes within an organisation may already have been formalised in line with different standards, such as ISO 9000. Hence any knowledge engineer attempting to model organisational processes can use these as a reference for the way processes occur.

11.4.2.7 Structures

The structures editor defines a graphical layout for the organisational structure relating to the given solution. However, for my generic model I wish to model a number of different possible organisational structures. Each structure model is characterised by this graphical structure and possible problems due to implementations of the given structure.

Aspects of organisational structure that have recently come to prominence are the functional and process-oriented views of organisational structure. From a concurrent design viewpoint, a number of organisations have found it beneficial to re-model their previously functionally-oriented organisational structures into a more process-oriented structure. This issue was illustrated by my experiences of case study 2. The organisation originally grouped personnel in functionally–oriented departments (design, manufacture, assembly etc). However, the organisation now sees 'design' as being one of the key processes undertaken by the organisation and now group multi-functional teams around different design projects. Figure 11.3 outlines the functionally-oriented structural view while figure 11.4 characterises the process-oriented structural view.

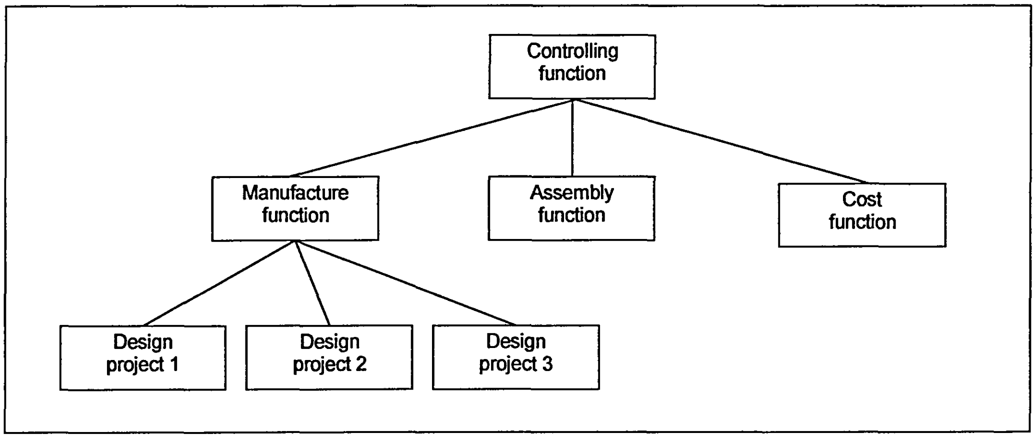


Figure 11.3: Functionally-oriented organisational structures.

These have personnel and resources grouped around different functional centres or departments. These resources are then shared out among different design projects. The problems inherent in this type of structure are that inter-departmental rivalries can inhibit the success of design projects.

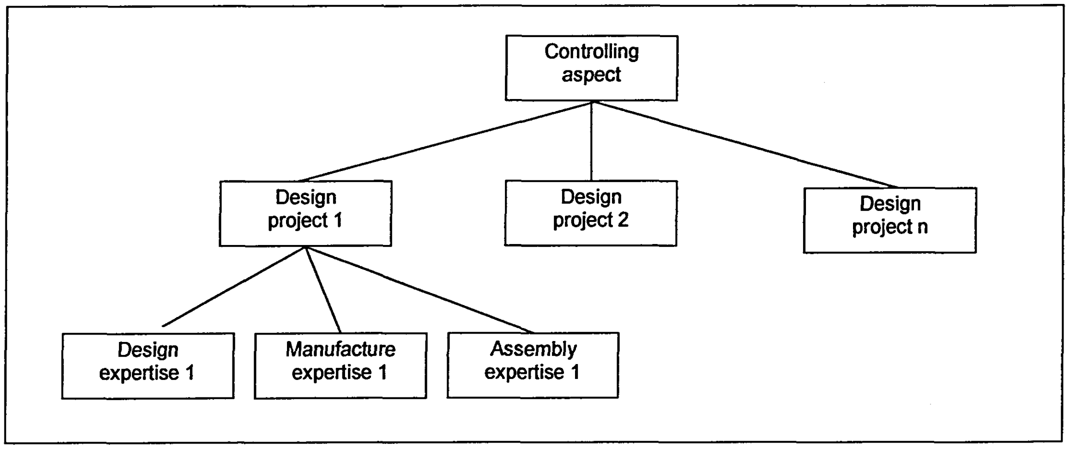


Figure 11.4: Project-oriented organisational structures.

Organisations who have successfully implemented the principles of concurrent design typically group resources and personnel around different design projects. In this way, multi-functional expertise is under the control of the project manager of the design project. This represents a move away from more departmental-oriented organisational structures where resources are seconded from different departments to design projects

11.4.2.8 Power authority

The power-authority links dictate what power authority issues (both formal and informal) exist between different stakeholders defined in the 'peoples' constituent. The findings from my case studies indicated that a number of diverse power relationships can exist within a concurrent design context. For example in case study 3 the designer had an informal controlling influence over the concurrent design process. However in case study 2, formalised authority structures dictated that the controlling was exercised by the project manager. In other cases, a more democratic power-authority structure was found to exist.

A power-authority relation can be defined for each stakeholder who has been created in the 'peoples' constituent with any other stakeholder. Each relation is characterised by the area in which the relation holds, the power base that maintains it's existence (such as the knowledge item owned by a stakeholder), the direction of influence between the stakeholders and indicators that provide evidence for such relationships. While some relationships may be formally defined, others may be more informal and harder to pinpoint. In any KBS it is important to represent the actual power relationships that exist, not those that are nominally in place.

As a result, for my generic model, the only characteristics that can usefully be defined are the power bases that define some relationships. As an example, table 11.7 defines the power relationship that would typically hold between an assembly critiquing agent and a designer. However, while this is what I would consider a generic example, a particular instantiation may have the designer with overall control of the process. For different instantiations of the organisational model, these power-authority relationships will be critically important and need to be defined.

name	pwr_Assembly_designer
area	Assembly_critiquing_agent informs and constrains Designer_agent in the area of assembly life-cycle perspectives
base	Assembly_critiquing_knowledge
direction	Assembly_critiquing_agent influences Designer_agent

Table 11.7: The power-authority relationships that exist between an assembly critiquing agent and a designer agent.

This stems from the assembly critiquing agent having specialised knowledge of the assembly life-cycle perspective.

11.5 *Common*KADS TASK MODEL

This describes the tasks that are performed in the organisational environment where the proposed KBS is to be implemented and can thus be seen as the realisation of some function within the organisation (Duursma et al [1994]). The task model should be independent of the particular agent which will accomplish the task although one of the task descriptors may be a reference to an agent that will perform the task.

Duursma et al [1994] suggest that the task model for the proposed solution need not necessarily be instantiated. The main reason for the instantiation is as a means of risk analysis to ensure that all possible tasks to be fulfilled by the proposed system are identified. For machine based agents, the main features of a task are effectively defined by the expertise task model linked to the agent(s) performing the task.

Each task is defined by an input, an output, a goal that the task will achieve, the related ingredient, it's control, it's features, it's environmental constraints and it's required capabilities. One interesting attribute of the goal of a task is the degree to which the goal is usually achieved. Clearly, a goal may not always be completely achieved.

Initially I considered either design or concurrent design to be suitable levels of granularity to describe the tasks in the task model. This can be further split into two stages of concept development and detail design. These two tasks are both accomplished using the concurrent design task. This task in turn can be further decomposed into propose, critique and negotiate tasks.

However, after experimenting with different levels of abstraction for task decomposition, propose, critique and negotiate were taken to be suitable levels of granularity to describe tasks in the task model. This granularity aspect is expanded on by Duursma et al [1994] – *"if activities are too fine grained, the size of the task model might be too big for practical validation...task analysis is performed in order to find out several things about the activities of people in an organisation."*

Depending on the way in which different tasks are split between human and machine based agents, the overall task structure and hierarchical task decomposition will exhibit many generic characteristics. However, depending on the different life-cycle perspectives brought to bear on the concurrent design task, the task model will be instantiated differently. For instance, if a key aspect of a proposed implementation is to critique designs from a manufacture perspective, the task 'critique_manufacture' would be of importance. In addition, while the task goals, inputs and outputs I would consider to be generic in nature, the control, features, environmental constraints, required capabilities and degree to which a task goal is usually accomplished will depend on the particular implementation considerations.

11.5.1 Entities defined in my generic task model

Table 11.8 outlines the entities defined in my generic task model.

name	**description**
task model context	The task model has been developed in the context of implementing KBS support for concurrent design within an organisation
all tasks	
tsk_Propose	The task whereby a design solution or partial solution is generated
tsk_Control	The overall task which controls the flow of the concurrent design process and has a controlling influence on any decisions
tsk_Critique_Assembly	The task whereby a design solution is critiqued from an assembly life-cycle perspective
tsk_Critique_Manufacture	The task whereby a design solution is critiqued from a manufacture life-cycle perspective
tsk_Critique_Cost	The task whereby a design solution is critiqued from a cost life-cycle perspective
tsk_Critique_Environment	The task whereby a design solution is critiqued from an environmental life-cycle perspective
tsk_Critique_Materials	The task whereby a design solution is critiqued from a materials life-cycle perspective
tsk_Critique_Electronics	The task whereby a design solution is critiqued from an electronics life-cycle perspective
all ingredients	The ingredients defined in the task model define the different ingredients that act as inputs and outputs to the different tasks. These ingredients can conceptually be linked to different knowledge roles defined in the expertise model for the different tasks. Interestingly, the workbench does not provide any facility to provide this link.
ing_Design_specification	The specification of the design in terms of functions the design solution must achieve together with any constraints on the way in which the design may be achieved
ing_Design_solution	The design solution as a relationship between a set of objects. This is the final deliverable from a given design problem.
ing_Life_cycle_constraints	Different life-cycle constraints relating to different perspectives.
ing_Conceptual_model	A conceptual model as a high-level commitment to a solution.
ing_Comnmitment_to_a_solution	An initial commitment to a design solution.
ing_Assessment	An assessment offered as part of a design critique.
ing_Degree_of_acceptance	An indication of the degree of acceptance offered as part of a design critique.
ing_Current_design	The current state of the design – I.e. the state the design is in now.
ing_Design_rationale	A design proposers reasoning as to why the design proposal is as it is.
ing_Alternative_proposal	An alternative to a given design proposal.
ing_Distance_measure	A qualitative assessment of perceived distance from the ideal from a certain life-cycle perspective.
ing_Functional_problems	Any perceived functional problems inherent in a given design proposal.
ing_Additional_constraints	Any additional life-cycle constraints the original generator of a design proposal has neglected to consider or did not have

	knowledge of.
ing_Reasons_for_critique	The reason why a design proposal has been critiqued.
ing_Revised_design_specification	A revised design specification based on any problems pinpointed in the original specification.
ing_Revised_design	A revised design based on a cycle of propose-critique-negotiate.
ing_Design_proposal	A design proposal which advances the current state of the design.
ing_Life_cycle_problems	Perceived problems from a life-cycle perspective.
all features	Task features characterise a task in terms of an abstract language. It is my belief that the task name and description are sufficient for defining what a task is for and hence I have not instantiated any generic task features which will act as properties to help define a particular task
all capabilities	These give a high-level view of the competencies needed for task performance.
cap_Design	Essentially the capability to design. This will require adequate design knowledge, knowledge of the domain the design is relevant to and a working knowledge of downstream life-cycle perspectives
cap_Critique	The capability of critiquing. This requires extensive domain knowledge of the relevant life-cycle perspective in order to be able to critique an evolving design from the perspective
cap_Negotiate	The capability of participating in negotiation tasks
all environments	This represents any environmental constraints that may constrain the performance of a particular task.
env_Working_hours	Human agents are typically restricted to work a maximum of 8 hours a day unless other precedents have been established
env_Time	Most tasks will be constrained in some way by the maximum amount of time that may be taken for task completion.

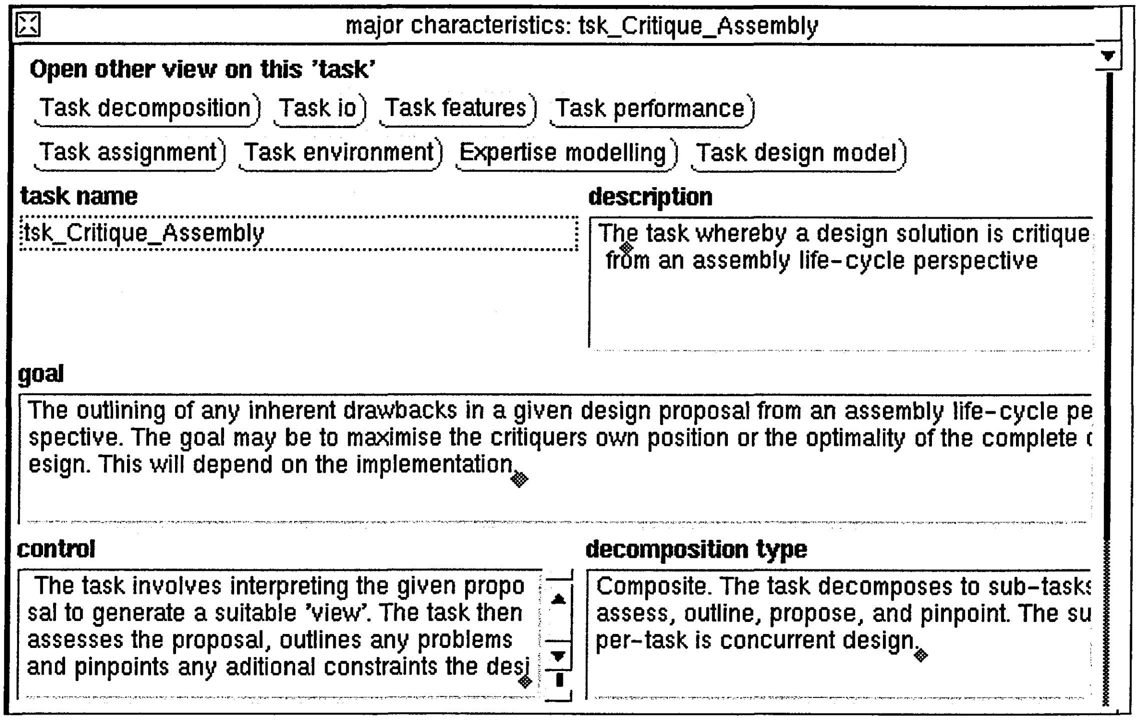
Table 11.8: The entities comprising my generic task model.

11.5.2 Characteristics of each task

Different views on each task can then be explored. It is here that links between the individual tasks and other models are defined.

11.5.2.1 Major characteristics

The major characteristics of a task are defined by the goal, the control and the decomposition type of the task. By way of example, the major characteristics for assembly critiquing (tsk_Critique_Assembly) are outlined in Table 11.9.



task name
tsk_Critique_Assembly

description
The task whereby a design solution is critique
from an assembly life-cycle perspective

goal
The outlining of any inherent drawbacks in a given design proposal from an assembly life-cycle pe
spective. The goal may be to maximise the critiquers own position or the optimality of the complete
esign. This will depend on the implementation.

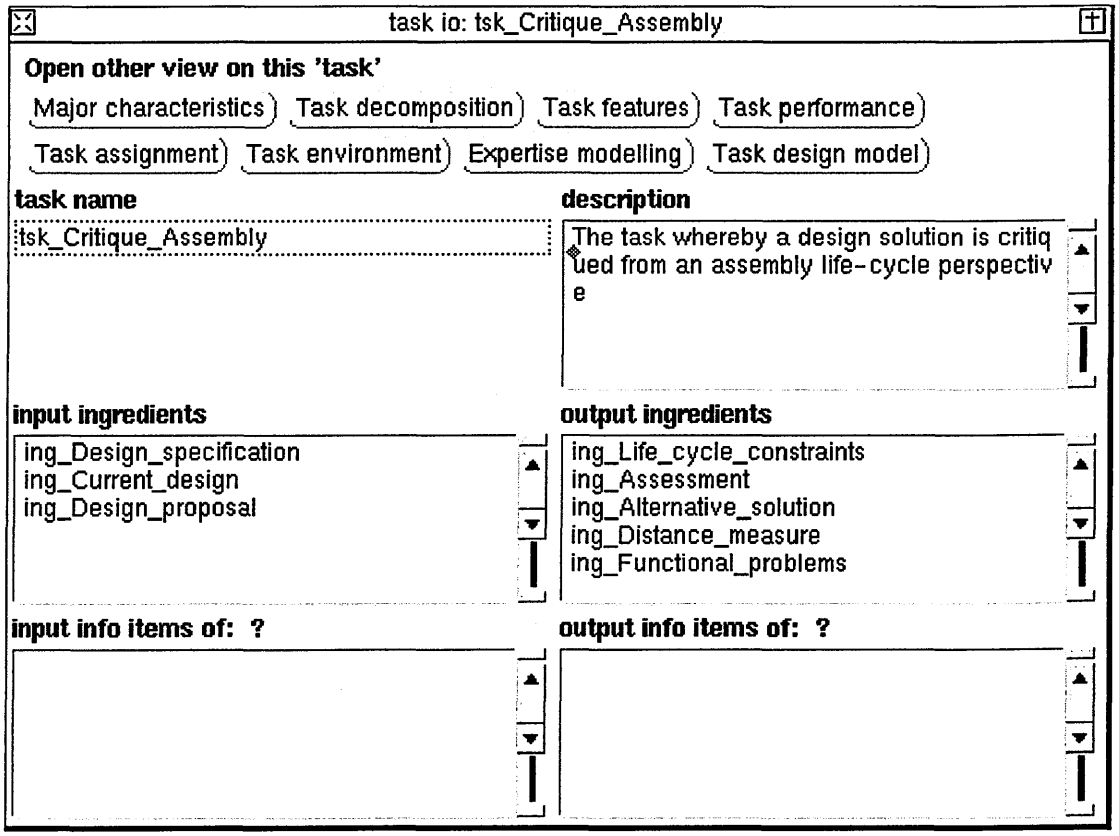
control
The task involves interpreting the given propo
sal to generate a suitable 'view'. The task then
assesses the proposal, outlines any problems
and pinpoints any aditional constraints the des

decomposition type
Composite. The task decomposes to sub-task
assess, outline, propose, and pinpoint. The su
per-task is concurrent design.

Table 11.9: The major characteristics for the task tsk_Critique_Assembly.
Of particular interest is the goal of the task as this dictates how 'concurrent' a particular implementation of concurrent design is.

11.5.2.2 Task input - outputs

The inputs and outputs for the task 'tsk_Critique_Assembly' are presented in Table 11.10.



task name

tsk_Critique_Assembly

description

The task whereby a design solution is critiqued from an assembly life-cycle perspective

input ingredients

ing_Design_specification
ing_Current_design
ing_Design_proposal

output ingredients

ing_Life_cycle_constraints
ing_Assessment
ing_Alternative_solution
ing_Distance_measure
ing_Functional_problems

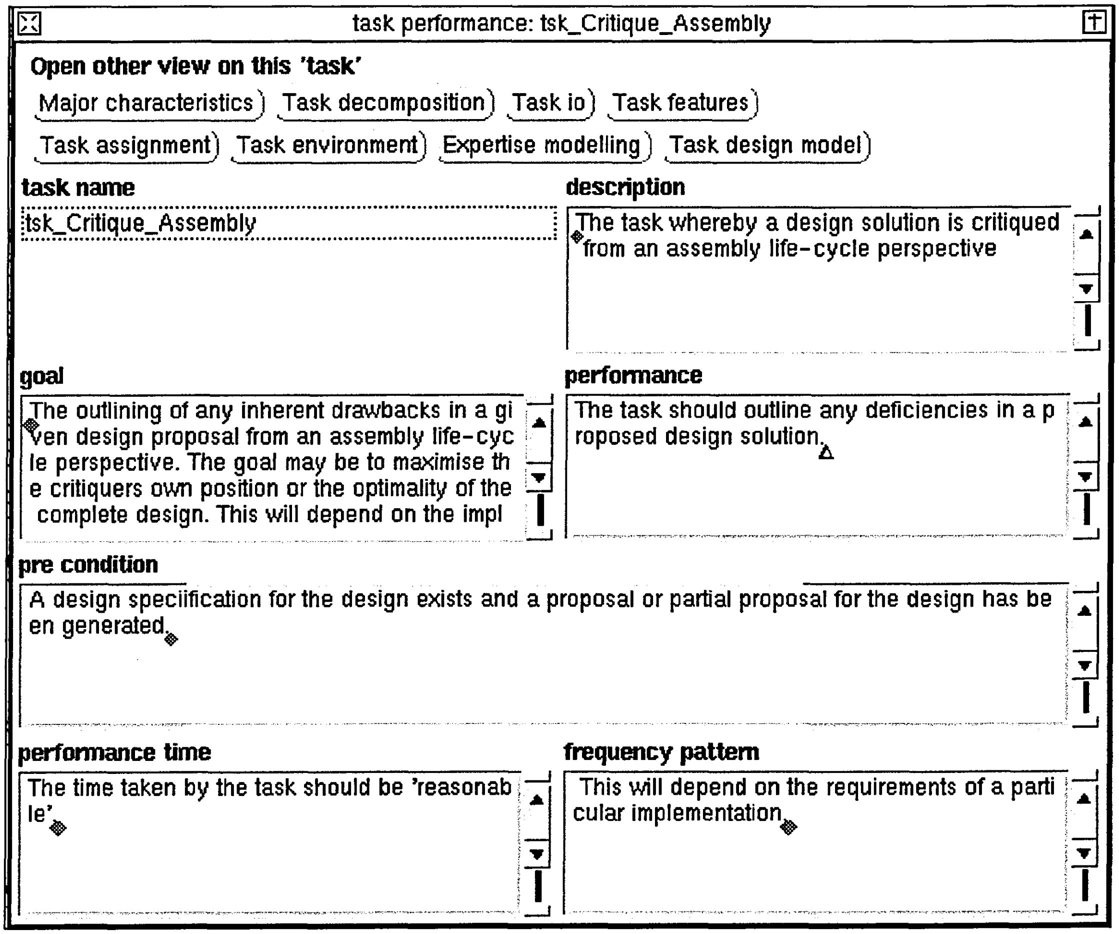
input info items of: ?

output info items of: ?

Table 11.10: The task inputs and outputs for the task tsk_Critique_Assembly.
This table shows the input and output ingredients for the given task.

11.5.2.3 Task performance

Table 11.11 defines performance parameters for the task 'tsk_Critique_Assembly'.



task name

tsk_Critique_Assembly

description

The task whereby a design solution is critiqued from an assembly life-cycle perspective

goal

The outlining of any inherent drawbacks in a given design proposal from an assembly life-cycle perspective. The goal may be to maximise the critiquers own position or the optimality of the complete design. This will depend on the impl

performance

The task should outline any deficiencies in a proposed design solution.

pre condition

A design speciification for the design exists and a proposal or partial proposal for the design has been generated.

performance time

The time taken by the task should be 'reasonable'.

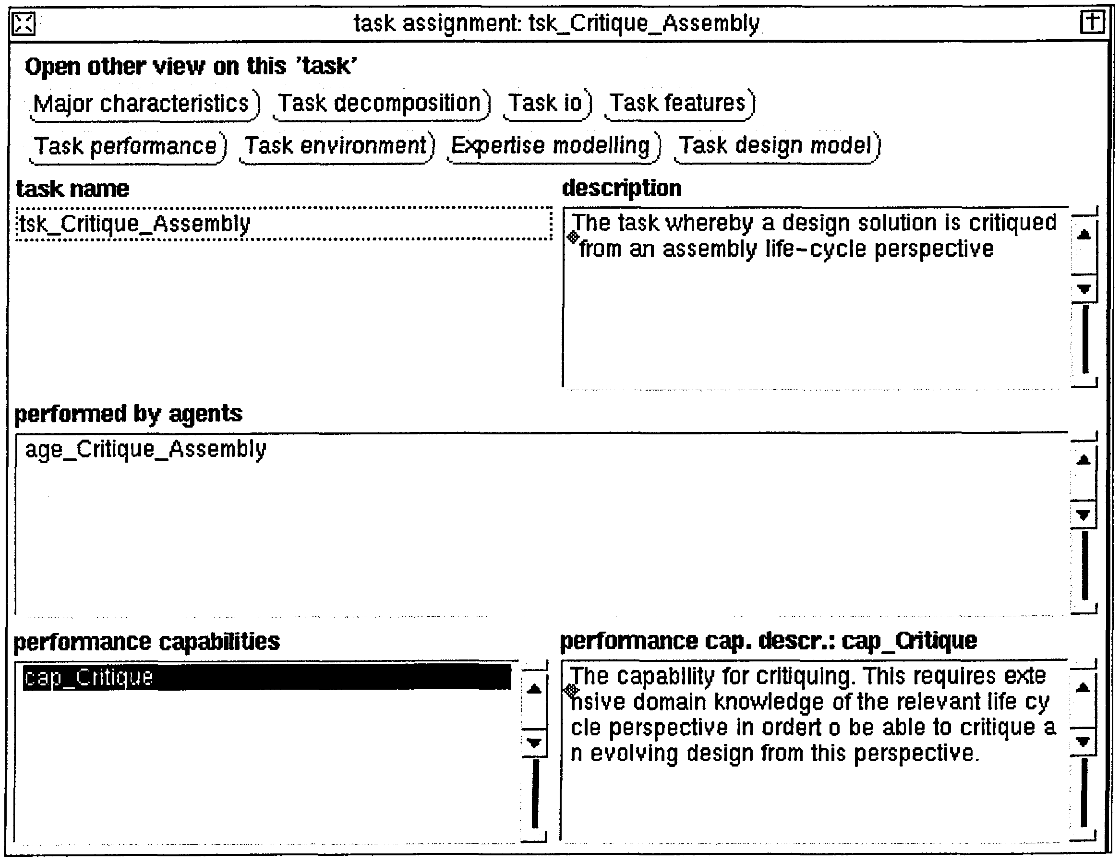
frequency pattern

This will depend on the requirements of a particular implementation.

Table 11.11: The performance parameters for the task tsk_Critique_Assembly.
This defines performance parameters for the given task in terms of time, frequency of use and any pre-conditions that are necessary for task application.

11.5.2.4 Task assignment

The task assignment defines which agents will perform the task and the reasoning capabilities required for the task. Table 11.12 details the task assignment for the task 'tsk_Critique_assembly'.



task name: tsk_Critique_Assembly

description: The task whereby a design solution is critiqued from an assembly life-cycle perspective

performed by agents: age_Critique_Assembly

performance capabilities: cap_Critique

performance cap. descr.: cap_Critique: The capability for critiquing. This requires extensive domain knowledge of the relevant life cycle perspective in ordert o be able to critique an evolving design from this perspective.

Table 11.12: The task assignment for the task tsk_Critique_Assembly.
The 'performed by agents' property indicates which agent in the agent model will perform this task.

11.5.2.5 Expertise modelling

The expertise modelling property assigns a link from the task in the task model to an expertise task model in the expertise model. For example, the task tsk_Critique_Assembly is linked to the expertise task model 'critique'. Ingredients and tasks defined in the task model should map to knowledge roles and tasks or inferences in the expertise model. The expertise model is described in detail in section 11.7.

11.6 *Common*KADS AGENT MODEL

This models all the relevant properties of any agent identified in the task model (Waern and Gala [1993]). An agent may be a KBS, a human or a non-KBS computer system. Where the agent is to be implemented as a machine based agent, the agent will be linked to one instance of the expertise model which describes the agent's problem-solving knowledge. For example, in my proposed model templates for concurrent design, an assembly-critiquing agent would be linked to an expertise model comprising assembly-critiquing problem-solving knowledge. An agent need not be a problem-solving agent (and hence linked to an expertise model). A common agent in most KBS is the data retrieval agent, which reads and writes data to a database system.

'Human' characteristics of the agents such as hearing, smelling etc will be entirely dependent on the implementation. In addition, there are generic statutory constraints, e.g. human agents are likely to be restricted, except in special circumstances, to a constrained working day (e.g. under EEC legislation, to a maximum of a 48-hour week). Human agents in any system will also be constrained by how much time they can allocate to design critiquing and the times they are available, although these have not been modelled as generic aspects. Any constraints on machine based agents will be dictated by a particular implementation.

Because each agent is implicitly linked to a task it is to perform, the agents involved in the system cannot be established until the task decomposition has been completed. From my case studies, I have identified a number of generic agents who participate in the concurrent design process. These include design proposal and critiquing agents who participate in the design proposal, critiquing and negotiation tasks. In addition, other generic agents are a management agent who has a controlling influence and a negotiation agent who can act as a mediator or controlling influence on the negotiation process. For any particular instantiation, a subset of these 'generic' agents may be instantiated.

11.6.1 Generic agents

Table 11.13 outlines the different agents I have included in the generic agent model template and other properties of the agent model.

name	description
agent model context	This agent model contains generic features for an agent model to support the concurrent design process.
all agents	
age_Designer	The design proposal agent
age_Negotiate	A negotiating agent
age_Customer	A customer agent with an interest in the functionality of the design solution
age_Critique_Assembly	The assembly critiquing agent has the task of critiquing design proposals from an assembly perspective and also for taking part in any negotiation tasks, where necessary.
age_Critique_Manufacture	The manufacture critiquing agent has the task of critiquing design proposals from a manufacture perspective and also for taking part in any negotiation tasks, where necessary.
age_Critique_Environment	The environmental critiquing agent has the task of critiquing design proposals from an environmental perspective and also for taking part in any negotiation tasks, where necessary.
age_Critique_Materials	The materials critiquing agent has the task of critiquing design proposals from a materials perspective and also for taking part in any negotiation tasks, where necessary.
age_Critique_Electronics	The electronics critiquing agent has the task of critiquing design proposals from an electronics perspective and also for taking part in any negotiation tasks, where necessary.
age_Critique_Cost	The cost critiquing agent has the task of critiquing design proposals from a cost perspective and also for taking part in any negotiation tasks, where necessary.
age_Management	The management agent has a controlling influence on the concurrent design process
all reasoning capabilities	
cap_Negotiate	The capability of negotiating when different proposals are presented by different knowledge sources
cap_Propose	The capability to propose design solutions or partial solutions
cap_Critique_Assembly	The capability to critique from an assembly life-cycle perspective.
cap_Critique_Manufacture	The capability to critique from a manufacture life-cycle perspective.
cap_Critique_Environment	The capability to critique from an environmental life-cycle perspective.
cap_Critique_Materials	The capability to critique from a materials life-cycle perspective.
cap_Critique_Electronics	The capability to critique from an electronics life-cycle perspective.
cap_Critique_Cost	The capability to critique from a cost life-cycle perspective.
cap_Manage	The capability to manage design projects.
all other capabilities	Different agents may also have other capabilities
all constraints	Different agents may be subject to different generic constraints

con_Max_working_day	Human agents are typically restricted to a maximum working day of 8 hours, except in special circumstances
con_Max_working_week	Human agents are typically restricted to a maximum working week of 48 hours, except in special circumstances

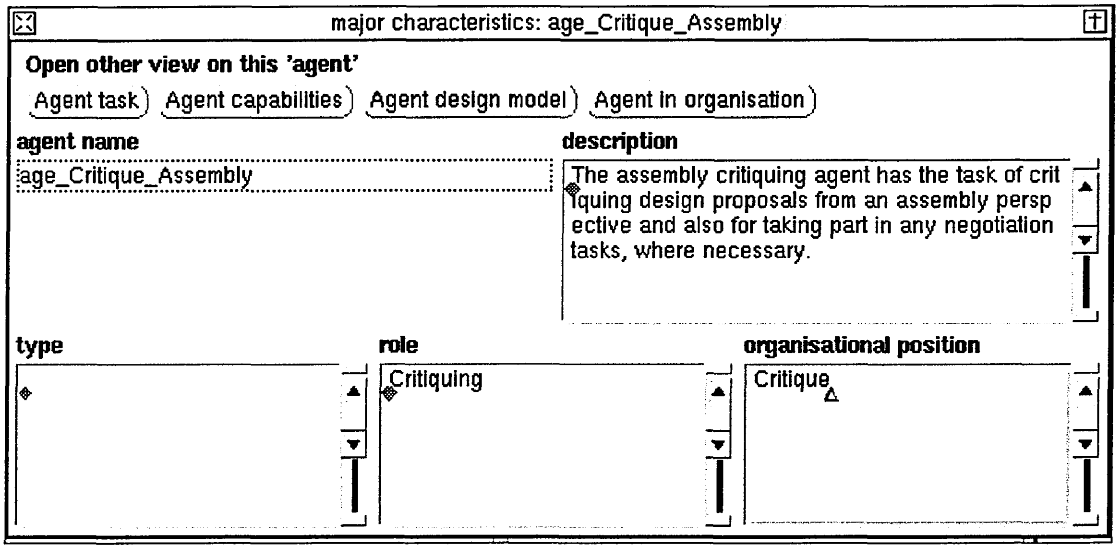
Table 11.13: The generic agents defined in my generic agent model for concurrent design.

11.6.2 Characteristics of an agent

Different views on each agent are used to show how different agents are linked to other models in the set of generic model templates. I will use the assembly critiquing agent and the different views to illustrate how this agent is linked to these other models.

11.6.2.1 Major characteristics

The major characteristics for each agent define the role of the agent together with the organisational position of the agent. As an example, the major characteristics of the agent age_Critique_assembly are detailed in table 11.14.


major characteristics: age_Critique_Assembly
Open other view on this 'agent'
Agent task
Agent capabilities
Agent design model
Agent in organisation

agent name: age_Critique_Assembly

description: The assembly critiquing agent has the task of crit iquing design proposals from an assembly persp ective and also for taking part in any negotiation tasks, where necessary.

type:

role: Critiquing

organisational position: Critique

Table 11.14: The major characteristics of the agent age_Critique_assembly.

11.6.2.2 Agent ingredient, task and communication transactions

The different ingredients supplied and received by the agent are linked to ingredients already defined in the task model and the tasks the agent performs are linked to tasks already defined in the task model. The different capabilities required by the agent are linked to the capabilities already defined in the agent model or if required, additional capabilities can be defined for the agent. These are illustrated in Table 11.15.

agent name	**description**
age_Critique_Assembly	The assembly critiquing agent has the task of critiquing design proposals from an assembly perspective and also for taking part in any negotiation tasks where necessary.
supplies ingredients	Ing_Life_Cycle_constraints Ing_Assessment Ing_Alternative_Proposal Ing_Distance_Measure Ing_Functional_Problems
receives ingredients	Ing_Design_Specification Ing_Current_Design Ing_Design_Proposal
has initiative in transactions	trans_Critique trans_Negotiate_Unilateral trans_Negotiate_Appease
participates in transactions	trans_Critique trans_Negotiate_Unilateral trans_Negotiate_Appease
performs tasks	tsk_Negotiate tsk_Critique_assembly

Table 11.15: The links to the information items links to the communication model and links to the task model for the agent age_Critique_assembly.

11.6.2.3 Agent capabilities

The different capabilities required by the agent are linked to the capabilities already defined in the agent model or if required, additional capabilities can be defined for the agent. As an example, the assembly-critiquing agent has assembly critiquing capabilities.

11.6.2.4 Agent in organisation

The agent is linked to a stakeholder and / or a computational resource defined in the organisational model. As an example, the age_Critique_Assembly agent is linked to the assembly_critiquing_agent stakeholder, previously defined in the organisation model.

11.6.2.5 Agent autonomy and importance

I consider the different agents to be working together (though not always cooperatively) to complete the overall concurrent design task. In my work I have also focussed on the relative autonomy of agents in concurrent design support systems (see Barker et al [1999], Barker et al [2000]) and also at how this autonomy may be altered. To an extent, this may also be defined in the task definition for the task that the relevant agent performs. I have also noted in my case studies how the relative importance of each agent can affect the overall design process. Hence I propose additional properties of each agent would be the degree of autonomy of the agent and the relative importance of each agent. These properties need to be determined at development time.

In addition, another issue that became clear from the case studies was the temporal variation in the weighting given to these levels of agent importance in the case studies. Issues such as manufacture assumed a greater importance as the design process progressed in a number of the case studies. In a system where such issues are important, this dynamic, temporal weighting of constraints would need to be incorporated. That is, the 'importance' of the agents needs to be adjustable on a time-basis.

11.7 *Common*KADS EXPERTISE MODEL

This is the model that has been the main focus of the research up until this point. This describes the knowledge of a particular 'agent' relative to a particular task and its use specific structure see Breuker and Van de Welde [1994]). Hence a particular expertise model is associated with an instance of an agent model. Expertise knowledge is split into application and strategic knowledge. Both types of knowledge are further split into three discrete levels. These are the task, inference and domain layers. My research has concentrated on the development of the task models in the application knowledge layer. The expertise model has been discussed in detail in Chapter 2.

The elements comprising my expertise model have been described extensively in chapters 8, 9 and 10. The main expertise task models developed have been for the propose, critique and negotiation tasks within concurrent design. The models for design proposal and negotiation have been outlined and documented in Chapter 10 hence I do not show the workbench instantiations of these models as generic templates again in this chapter. However, for the critiquing task, I have now identified an additional knowledge role which plays an important part in the critiquing process. This is essentially the 'goal of the task'. Where the optimality of the complete design is the goal of the task, the design specification can be assumed to suffice. However, where the agent is effectively 'self interested', the task goal for the associated agent must be signalled to the expertise modelling process as an additional knowledge role, as illustrated in Figure 11.5. This outlines the workbench instantiation of the task decomposition for the critiquing task, previously presented in figure 10.9.

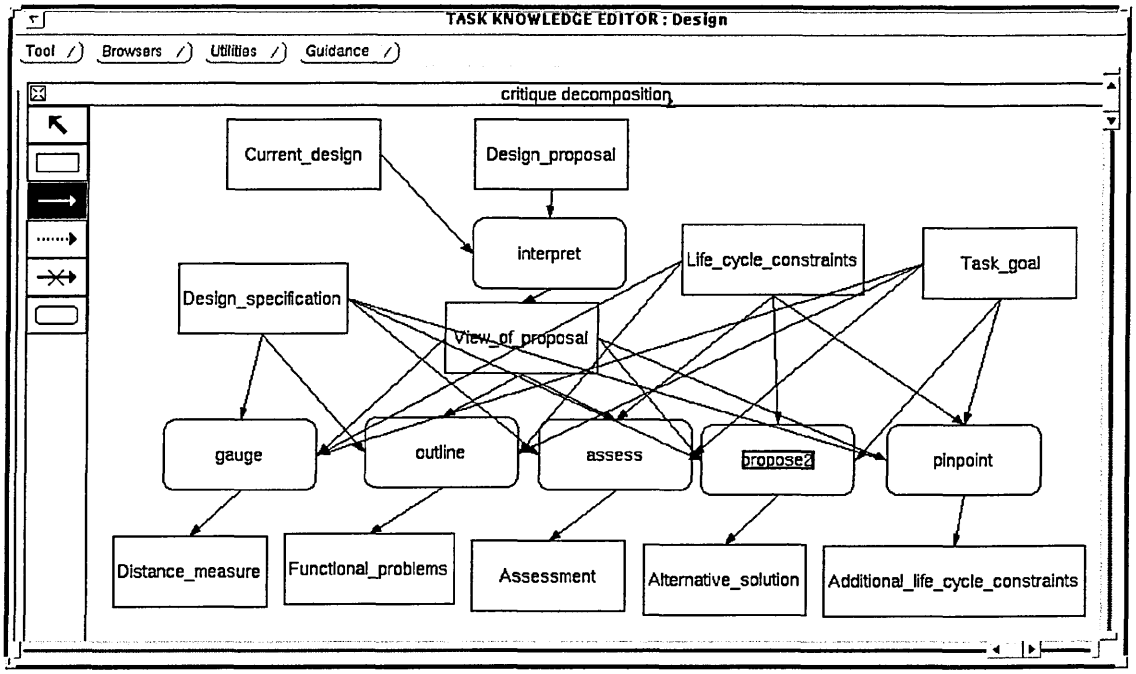


Figure 11.5: The task decomposition for the generic critiquing task

This shows the sub-tasks and knowledge roles that are utilised to derive a design critique from a given design proposal. Similar decompositions exist on the workbench for models previously presented in Figures 10.7 to 10.11.

Figure 11.6 outlines the overall task hierarchy I have constructed for concurrent design while Figure 11.7 shows the task decompositions for the propose, critique and negotiation tasks.

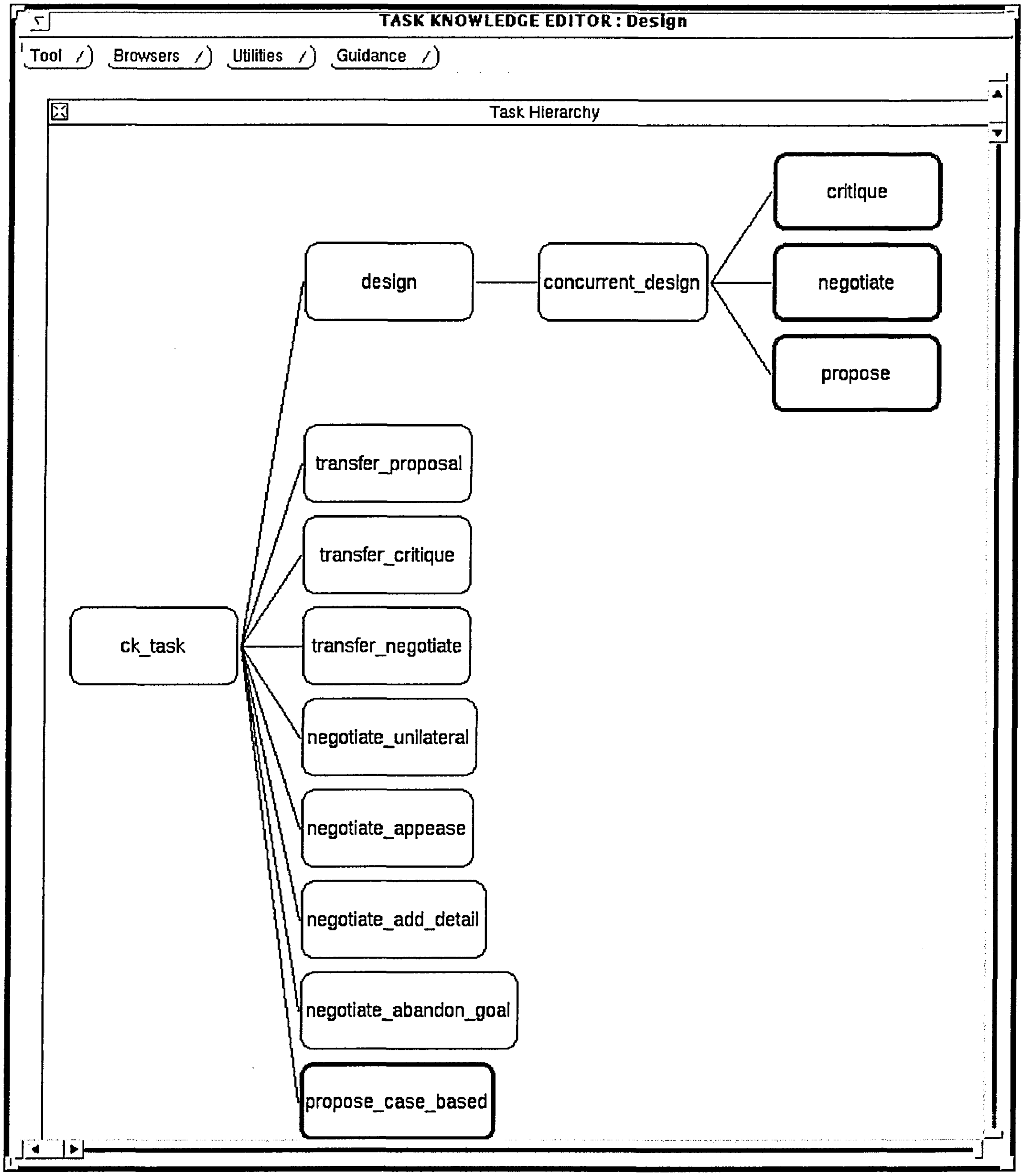


Figure 11.6: The task hierarchy for concurrent design and associated tasks.

The negotiate and transfer tasks shown in this figure represent different types of negotiation and expertise-intensive transfer tasks, linked to the *Common*KADS communication model respectively. These are described more fully in the text.

An assumption of the *Common*KADS model set is that an expertise task model will be directly linked to one agent. Hence, a critiquing agent would have an expertise task model for critiquing and a design agent would have a model for proposing designs. However, for the negotiation task, this is less clear cut. It is believed that different agents will be involved in the negotiation process. However, the task distribution amongst the agents is not clear cut and will not be until the negotiation process in a concurrent design context is more clearly understood. Hence, for the purposes of my generic expertise task model, I have departed from the *Common*KADS norm and incorporated the propose, critique and negotiate tasks within the same expertise task model, which I have called concurrent design. In any implementation, these expertise task models are likely to be split across different expertise models, which reference particular agents.

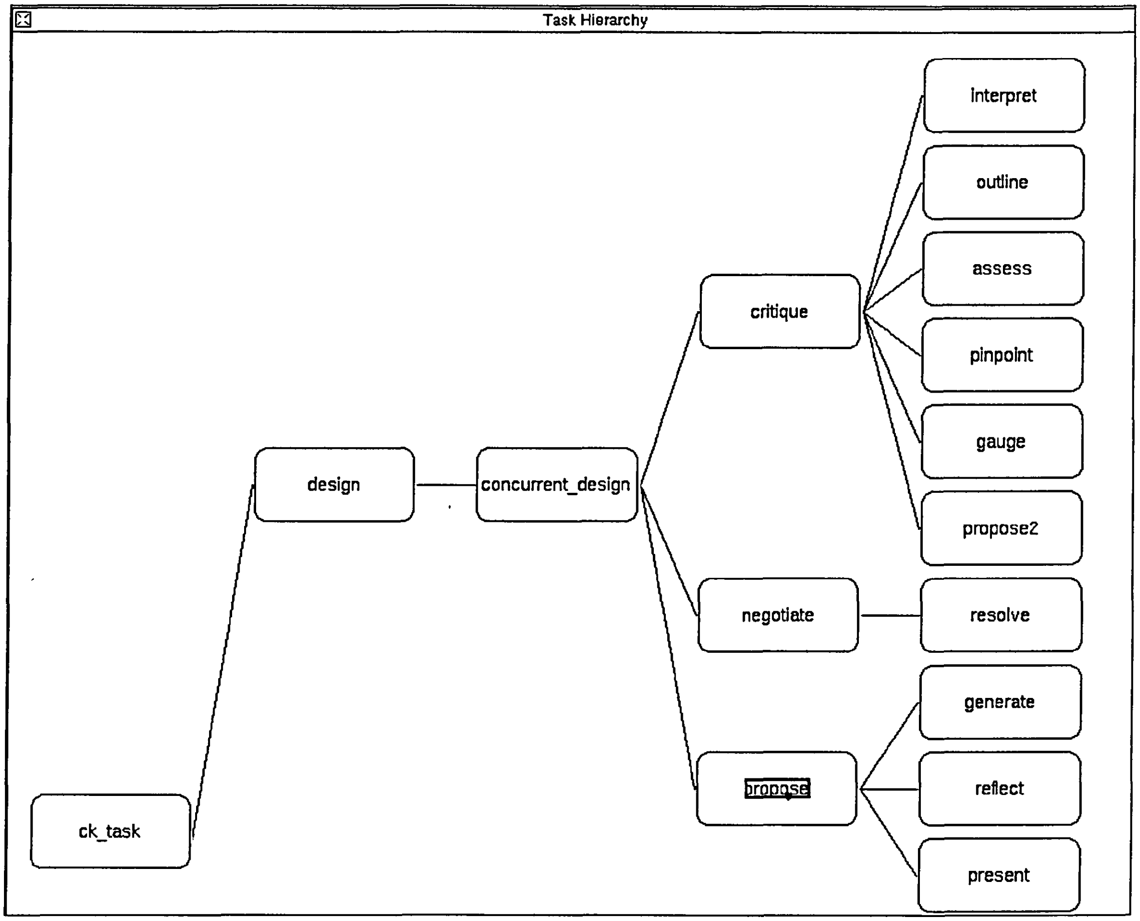


Figure 11.7: The generic task decompositions for the propose, critique and negotiate tasks.

My developed expertise task models have been derived from case studies of designers operating concurrently hence I have a reasonable belief that the models are generic for the process of concurrent design. As a result, the expertise task models outlined in Chapter 10 are assumed to be generic for the possible different agents comprising a proposed implementation. Based on my case studies, I believe my generic model for critiquing, outlined in figure 11.5 is sufficiently generic across a range of different agents and domains.

However, the task decompositions, for the propose and negotiate tasks in particular, will be dependent on the problem-solving methods applied to the tasks and are therefore implementation dependent. This is defined in *Common*KADS by the domain specific, strategic knowledge of design. I.e. what problem-solving methods should be applied to define different task structures. In my generic expertise task models for propose and negotiation, I have outlined general, high-level models for both these tasks. Because of the way the *Common*KADS workbench is currently structured, it is not possible to present different task decompositions for the same task within a single project. However, I believe it is important to present such possible decompositions as generic models. The possible decompositions for the propose task have been well documented by Bernaras and Van de Welde [1994], so I have not attempted to duplicate their work in my generic models. However, possible task decompositions for the negotiate task have not been documented. The distinctive modes of negotiation that I encountered in my case studies I defined as 'unilateral decision' and 'appeasement'. Task models for the two modes have already been presented in figures 10.9 and 10.10. These modes have been modelled as separate tasks from the main design task in my generic workbench models. The negotiation mode ‘majority decision’ I have neglected to model as this essentially involves counting votes.

Other researchers, in particular Klein and Lu [1989], Harrington et al [1995] and Werkman [1991] have documented different models for negotiation, albeit not in the syntax of *Common*KADS. Klein and Lu and Harrington et al distinguish between computational methods of negotiation and human methods. Because I have attempted to model 'human' problem-solving processes within my models, the computational methods they describe I do not consider as being suitable for modelling as generic negotiation models for human negotiation. However, such methods may be useful for computational implementations of any human-based negotiation strategy.

Two of the high-level 'human' negotiation strategies detailed by Klein and Lu (alternate and compromise) I would consider to be similar to my model for appeasement. In addition, the 'unilateral decision' strategy I have described corresponds closely to the 'arbitrator determines the final solution' strategy described by Werkman.

However, the other two negotiation strategies detailed by Klein and Lu, 'abandon goal' (cf. Werkman – ‘concede issue’) and 'add detailing' do not correspond to any of the strategies I found in my case studies. Hence, I have included task decompositions for these modes of design within my generic models. The form of these is outlined in figures 11.8 and 11.9.

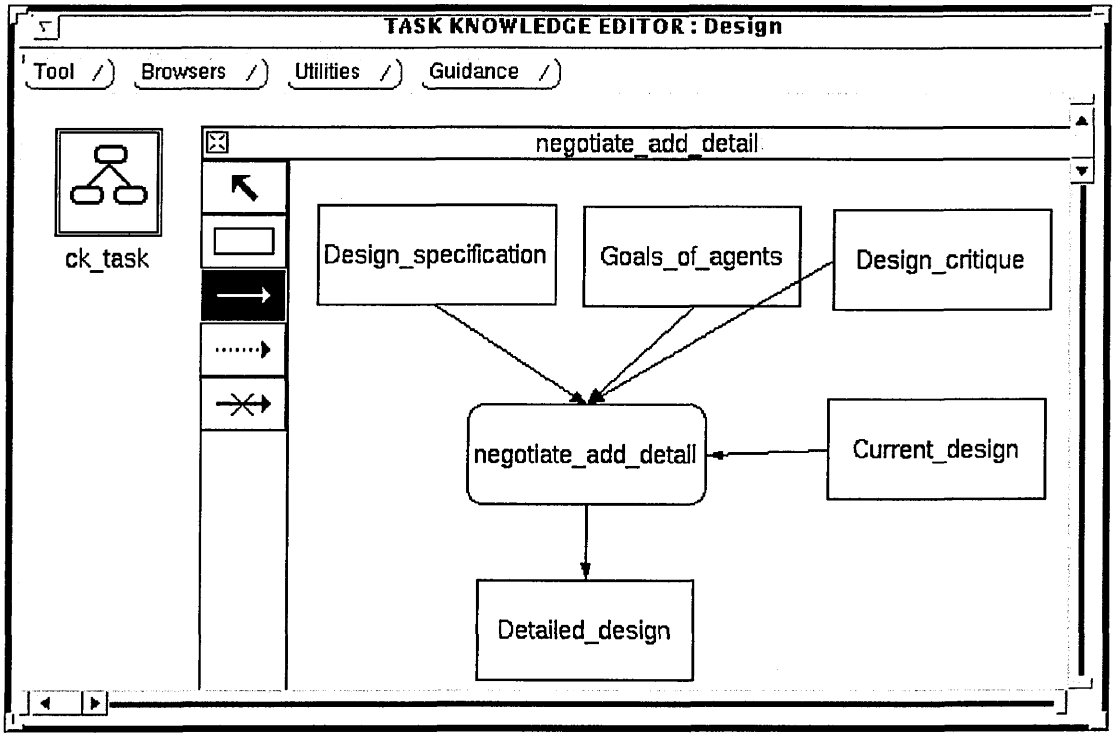


Figure 11.8: Conflict avoidance by adding detail.

By adding detail to a design, the research of Klein and Lu [1989] suggests that conflicts between agents can be resolved.

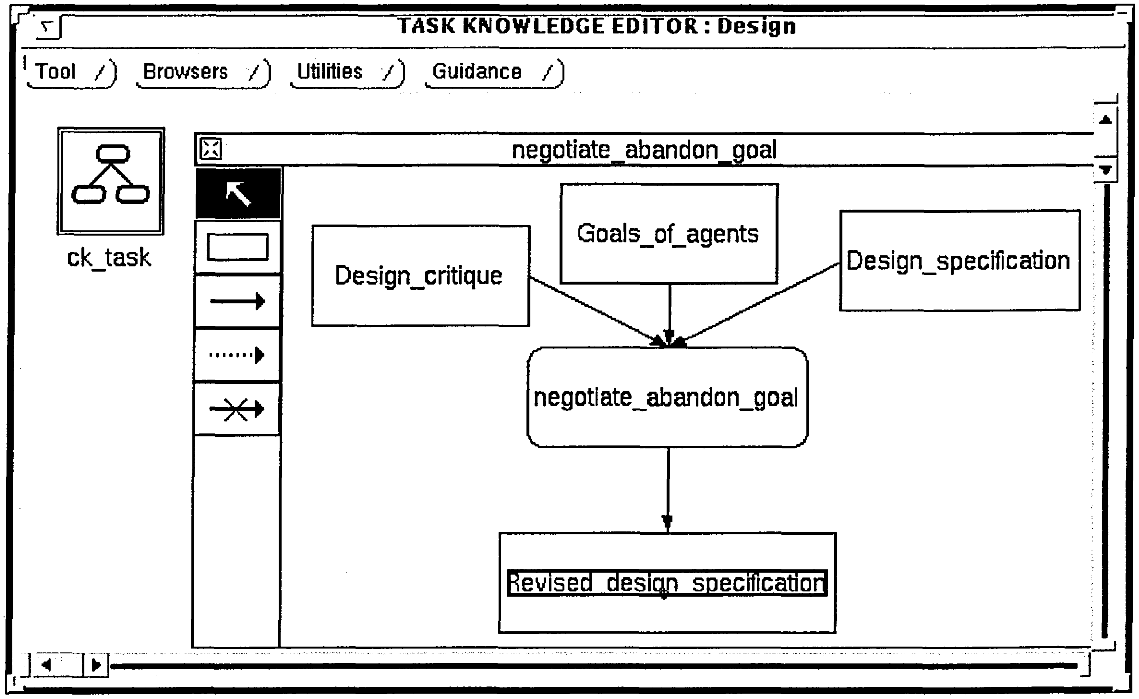


Figure 11.9: Conflict avoidance by abandoning a goal.

By abandoning a particular goal of a design specification, Klein and Lu [1989] suggest that conflicting critiques from agents can be resolved.

It is also worth noting that a particular negotiation session may make use of a number of these different strategies in a dynamic way. For example, some form of appeasement may be used to bring two conflicting agents closer to agreement before a unilateral decision from a controlling influence dictates which path to take. Hence my expertise task models for negotiation are best seen as discrete strategies that may be utilised at different times within the concurrent design process.

As well as the tasks required for actual problem-solving, there are a number of knowledge-intensive transfer tasks, which are required where certain items cross agent boundaries (for example design proposals). The transfer tasks transfer_proposal, transfer_critique and transfer_negotiate have been incorporated in my generic expertise model in order to illustrate how transfer tasks such as presenting proposals to critiquing agents is a knowledge intensive task in itself. Case study 3 gave a number of examples of how the designer would attempt to win round different critiquing agents with various strategies. Clearly, this is a knowledge intensive task in itself, although I have not attempted to model in detail the structure of such knowledge in a task type format. It is felt that such communication abilities are of more relevance to human based systems. Communications between machine based agents will be defined by certain protocols and 'humanesque' communication strategies will not be required, although they should clearly inform any machine based methods used.

11.7.1 The Conceptual modelling language (CML)

CML is a highly structured, semi-formal notation for representing the contents of a *Common*KADS expertise model in a pseudo-code format. The CML notation allows the complete expertise model, consisting of domain, inference and task knowledge to be represented. For more details of CML, see Schreiber et al [1994b].

11.8 *Common*KADS COMMUNICATION MODEL

This describes how different communication tasks occur between the different agents comprising the KBS (Waern et al [1993]). Effectively this governs all events where communication transactions cross agent boundaries. The communication model specifies protocols for how different communication events should occur, in particular where the initiative for the communication will come from.

In the Expertise model, each interaction with the environment is represented as a transfer task. Every transfer task must be represented as a transaction within the communication model. The communication model provides the concepts and mechanisms for a communication based on the competence of the communicating agents. The main components of the communication model are described in table 11.16.

Transaction	A collection of ingredients exchanged between agents.
Information item	The specification in an admissible vocabulary and syntax of the ingredients
Capabilities	The skills and knowledge required for the agent to participate in the transaction.
Transaction plan	This deals with sequencing issues and is part of a larger plan of how to arrange an effective information exchange between two agents.
Initiative	Any transaction must be initiated by a particular agent.

Table 11.16: The entities comprising the *Common*KADS communication model.
This illustrates the characteristics of the main components comprising the Communication model.

There are clear relationships between other models and the communication model. In particular, the expertise and agent models may impose constraints on the communication model while the task model defines the ingredients that are exchanged between agents. Hence the landmark state 'ingredients described' in the task model must be complete before the state 'information items specified' can be reached in the communication model.

The main ingredients exchanged between agents are design proposals and corresponding critiques and any communications generated by negotiation tasks. As such, in the generic template I have modelled transactions for design proposals and critiquing (in terms of the ingredients of a proposal and a critiquing communication). In addition, I have outlined a skeletal model for negotiation transactions, although the features of this transaction are more instantiation dependent than for the communications implied by design proposal and critiquing. From my case studies, I have defined two different types of negotiation, appeasement and unilateral decision. Hence I define transaction plans for both these approaches.

Many of the features of the communication model are felt to be generic, with the exception of the nature of discourses and the initiators of transactions. In addition, there is extensive knowledge required on the part of the design proposer (the designer) to be able to represent the proposal in a form that is meaningful to the critiquing agents. There is also a requirement for the critiquing agents to be able to form a meaningful view of the given proposal. Whether these represent transfer or problem-solving knowledge is debatable. I have effectively included them in the expertise task models for propose and critique respectively. However, there is also communication specific knowledge relevant to the communication tasks, necessary for the propose, critique and negotiate tasks. These are discussed in an earlier section (11.4.4) where I describe the expertise model.

11.8.1 Entities in the generic communication model

The entities comprising my generic communication model are outlined in table 11.17.

name	**description**
context	Describes the communications implied by design proposal, critiquing and negotiation.
all transactions	
trans_Proposal	The transaction whereby design proposals are made available to critiquing agents
trans_Critique	The transaction whereby critiques are made available to designers or other critiquing agents
trans_Negotiate_Unilateral	The transaction implied by a unilateral negotiation approach
trans_Negotiate_appease	The transaction implied by an appeasement form of negotiation
all discourses	
disc_Propose	The transaction whereby design proposals are made available to critiquing agents
disc_Proposal_rationale	The discourse whereby a designer explains the rationale behind a design proposal to critiquing agents
disc_Distance_measure	The discourse whereby a distance measure is explained to a design proposer by a design critiquer.
disc_Functional_problems	The discourse whereby functional problems in a design from a given life-cycle perspective are explained to a design proposer by a design critiquer.
disc_Assessment	The discourse whereby an assessment from a given life-cycle perspective are explained to a design proposer by a design critiquer.
disc_Alternative_proposal	The discourse whereby an alternative proposal from a given life-cycle perspective are explained to a design proposer by a design critiquer.
disc_Additional_constraints	The discourse whereby any additional constraints from a given life-cycle perspective are explained to a design proposer by a design critiquer.
all info items	
ing_Design_specification	The design specification in terms of functions the design must achieve together with any constraints on the way in which the design may be achieved
ing_Design_proposal	A proposal which advances the state of the current design at a certain level of granularity
all capabilities	
cap_Form_view	A communication relevant capability is that the different agents must be able to form a meaningful view of design proposals and corresponding ingredients
cap_Time_to_present	The capability of knowing when to present a design proposal to critiquing agents. Too often and the design time will be excessive, too late and the design may have advanced too far.

Table 11.17: Entities of my generic communication model for concurrent design.

11.8.2 Transactions

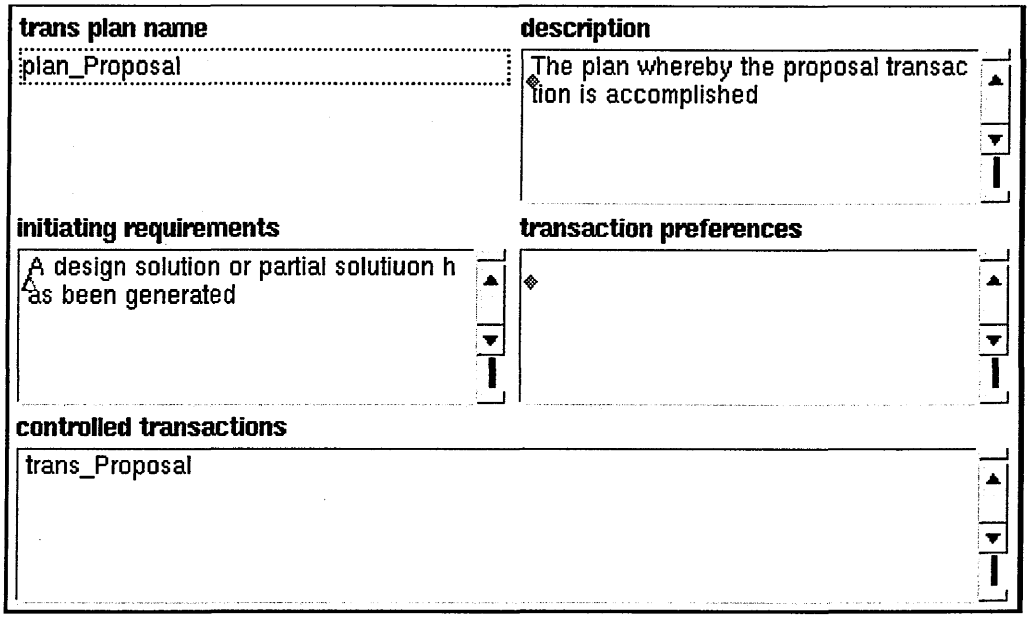
Each transaction entity is characterised by different properties. The ‘transaction’ entity for trans_Proposal is outlined in Table 11.18.

name	**description**
trans_Proposal	The transaction whereby design proposals are made available to critiquing agents.
Major characteristics	
Communication_type	Sensitive
Extra ingredients	None
Transaction structure	Undefined
realising discourses	disc_Propose, disc_Propose_rationale
part of plans	plan_Proposal
has ingredients	ing_Design_solution
Transaction agents	This outlines the participating agents in the transaction and links to the agent model are defined.
initiating_agents	**participating_agents**
age_Designer	age_Designer
	age_Critique_Assembly
	age_Critique_Manufacture
	age_Critique_Environment
	age_Critique_Materials
	age_Critique_Electronics
	age_Critique_Cost
Expertise and design modelling	Undefined
describes transfer tasks	transfer_proposal
implemented by subsystems	Undefined

Table 11.18: The transaction 'Trans_proposal'.
This is defined by a number of different properties and links to other models.

11.8.3 Transaction plans

The transaction plan defines the ordering of different transactions. The plan_Proposal, detailed in Table 11.19, defines how the trans_Proposal transaction is accomplished.



trans plan name
plan_Proposal

description
The plan whereby the proposal transaction is accomplished

initiating requirements
A design solution or partial solutiuon has been generated

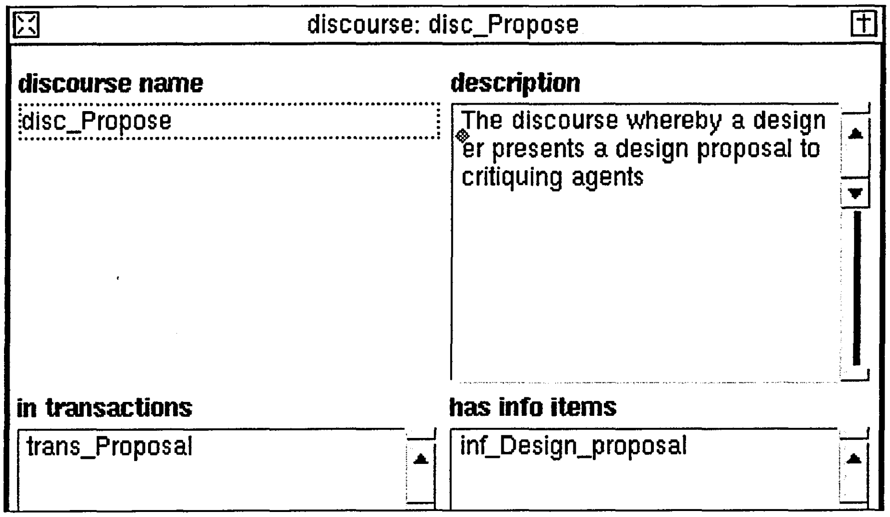
transaction preferences

controlled transactions
trans_Proposal

Table 11.19: The 'plan_Proposal' transaction plan.

11.8.4 Discourses

The discourse entity disc_Propose outlines in more detail how part of the trans_Proposal transaction occurs.



discourse: disc_Propose

discourse name

disc_Propose

description

The discourse whereby a designer presents a design proposal to critiquing agents

in transactions

trans_Proposal

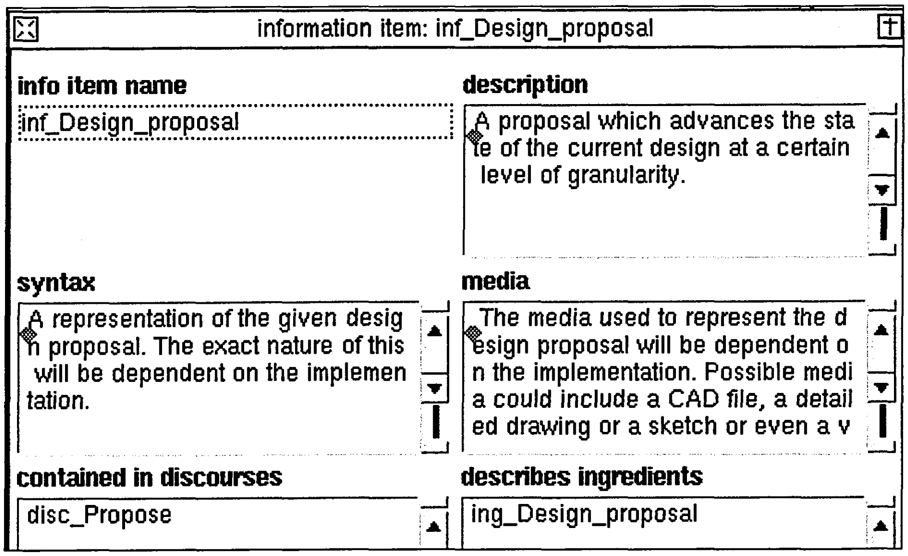
has info items

inf_Design_proposal

Table 11.20: The 'disc_Propose' discourse entity.

11.8.5 Information items

The information item inf_Design_proposal is characterised in Table 11.21.



information item: inf_Design_proposal

info item name

inf_Design_proposal

description

A proposal which advances the state of the current design at a certain level of granularity.

syntax

A representation of the given design proposal. The exact nature of this will be dependent on the implementation.

media

The media used to represent the design proposal will be dependent on the implementation. Possible media could include a CAD file, a detailed drawing or a sketch or even a v

contained in discourses

disc_Propose

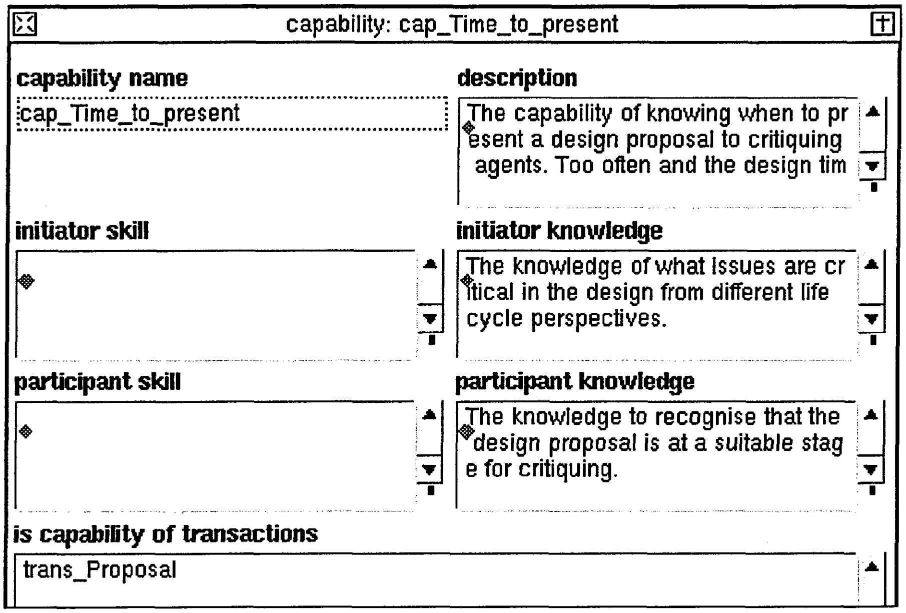
describes ingredients

ing_Design_proposal

Table 11.21: The 'inf_Design_proposal' information item.

11.8.6 Capabilities

The capability cap_Time_to_present is characterised in Table 11.22.



capability: cap_Time_to_present

capability name

cap_Time_to_present

description

The capability of knowing when to present a design proposal to critiquing agents. Too often and the design tim

initiator skill

initiator knowledge

The knowledge of what issues are critical in the design from different life cycle perspectives.

participant skill

participant knowledge

The knowledge to recognise that the design proposal is at a suitable stage for critiquing.

is capability of transactions

trans_Proposal

Table 11.22: The 'cap_Time_to_present' capability.

The initiatives for the different transactions will depend on the agents comprising a particular implementation. For example, the trans_Proposal transaction could conceivably be initiated either by the designer or a critiquing agent who wanted to see what the designer is doing. In addition, a number of different agents can participate in a transaction based on the implementation. The knowledge required to participate in the transaction is defined by the transfer task transfer_proposal in the generic expertise model.

11.9 *Common*KADS DESIGN MODEL

Because the *Common*KADS design model is essentially the link between the model templates and a potential implementation, this model is dependent on a number of other models being developed and is by far the most implementation dependent of the model set. Hence the development of the design model will by necessity be done towards the end of the complete *Common*KADS model set development. The design model operationalises the entities of other models developed and is decomposed into the application model, architectural and platform design models (Van de Welde et al [1994]).

11.9.1 Architectural and platform design

The architectural model effectively defines the computational architecture of any solution. As an example, a possible implementational architecture would be some form of rule-based system. The platform design model defines issues such as the programming language that will be used to implement the given computational architecture and what hardware platform will be used.

As such, these two aspects of the design model are very implementation dependent and I have not defined any generic aspects for these models. As with other models, where I have defined the generic models in terms of a super-set of components from which possible instantiations may be developed, it would be possible to define all possible programming languages and hardware platforms. However, I did not think this would help to inform potential developers of KBS. For the implementation scenarios (see Chapter 12) I have discussed how the architectural and platform models are instantiated for the scenarios.

11.9.2 Application design

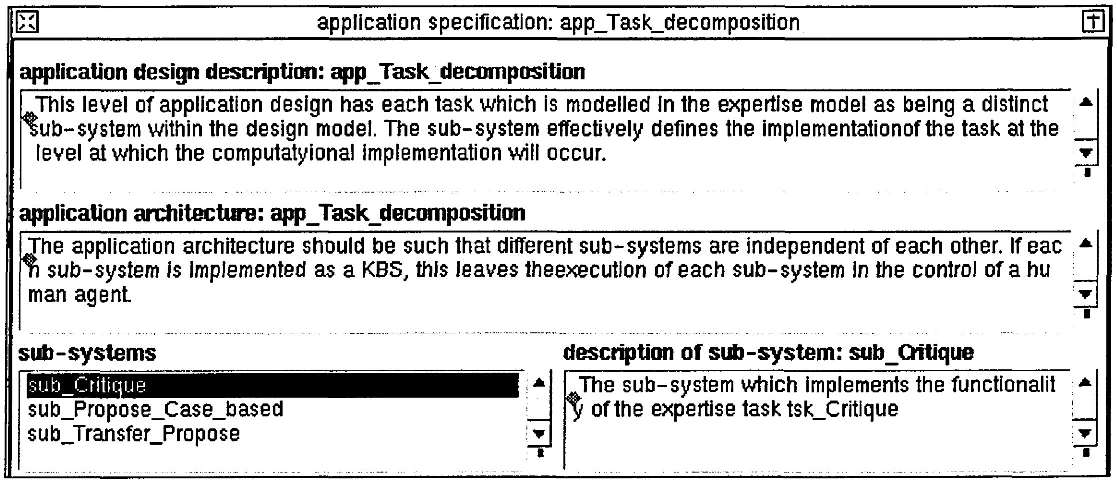
Although I have not attempted to define generic architectural and platform models, I have developed application models which operationalise the expertise task models I have defined. Table 11.23 summarises aspects of my generic design model

name	**description**
design_model	The planned implementational model for the generic *Common*KADS model set for concurrent design
all applications	
app_Task_decomposition	The level of application design has each task which is defined as an expertise task model as being a distinct sub-system at the computational level at which the implementation will be operationalised.
all architectures	
Undefined	
all platforms	
Undefined	

Table 11.23: The generic high-level design model.
This is decomposed into application, architecture and platform design. As outlined in the text, I have only considered those application design aspects which operationalise tasks defined in the expertise task model.

11.9.2.1 Application specification

The application specification for the application 'app_Task_decomposition' defines the sub-systems which will operationalise this application aspect. The elements of this application are defined in Table 11.24.



application design description: app_Task_decomposition

This level of application design has each task which is modelled in the expertise model as being a distinct sub-system within the design model. The sub-system effectively defines the implementationof the task at the level at which the computatyional implementation will occur.

application architecture: app_Task_decomposition

The application architecture should be such that different sub-systems are independent of each other. If each sub-system is implemented as a KBS, this leaves theexecution of each sub-system in the control of a human agent.

sub-systems

sub_Critique
sub_Propose_Case_based
sub_Transfer_Propose

description of sub-system: sub_Critique

The sub-system which implements the functionality of the expertise task tsk_Critique

Table 11.24: The application architecture of the design model.
This defines the sub-systems that will operationalise aspects of different models.

The sub system conceptual entity operationalises or implements the expertise task model. Each sub-system has a number of properties which are defined in three different views - major characteristics, detailed design and decomposition.

11.9.2.2 Properties of sub-system 'sub_Propose_case_based'

The properties of the sub-system 'sub_Propose_case_based' are detailed in Table 11.25.

name	**description**
Major characteristics	
type	KBS
functionality	The sub-system module uses a case-based reasoning mechanism to generate a design solution or partial solution from an input design specification and a library of previous design cases or solutions.
implements_task	tsk_Propose
Decomposition	Each entity defined will fulfil the partial functionality of the complete sub-system.
has decomposition into the following	
ent_Select_previous_design	This code module will select a suitable case from a library of previous design solutions
ent_Split_up_previous_design	This code module will split up the previous case into sub-components
ent_Generate_new_design_solution	This code module will generate a design solution for each sub-component.
ent_Combine_sub_solutions	This code module will combine the solutions for each sub-component into a complete solution
Detailed design	
program description	The program will generate design solutions or partial solutions based on an input design specification and a library of previous design cases
realised within architecture design	*none*
instantiates interface media	Undefined
calls computational methods	Undefined
calls interface activities	Undefined

Table 11.25: The major characteristics, decomposition and detailed design of the 'sub_Propose_case_based' sub-system.

Each entity defined in the sub-system decomposition can be linked to the entity in the expertise model, which this sub-system entity operationalises. As an example, the entity 'ent_Select_previous_design' links to the expertise model sub-task select_case. The detailed design is where elements of the application design are linked to the architectural design ('call computational methods' etc). As I have not defined any generic elements of the architectural design, I have not made these links in my generic design model.

11.10 Chapter Summary

In this chapter I have described the different model templates comprising the full *Common*KADS model set. In addition, I have outlined how a number of software tools can be used to support the further development and refinement of these model templates. In particular, the *Common*KADS workbench has been developed in order to aid knowledge engineers and teams in the development of knowledge-based systems, using the *Common*KADS methodology.

This chapter then describes, with examples, how the different model templates comprising the *Common*KADS model set have been refined and extended as generic model templates to aid in the development of computer tools to support the concurrent design process, using the *Common*KADS workbench. The elements of the different models I have taken to be generic are based on my findings from different case studies, outlined in earlier chapters. As such, my generic model templates offer a super-set of entities from which different possible scenarios may be instantiated. It must also be emphasised that my generic models have been developed from my experiences with different case studies of organisations practicing concurrent design. As a result, I do not presume that my templates are in any way definitive in terms of offering support for the concurrent design process. Rather, I present them as generic models to be developed, extended and refined by system developers.

The extensive process of developing the models on the workbench ensures the generic templates I have developed adhere to the formalisms of the *Common*KADS methodology while also showing the interactions and dependencies between different models. Some of the models comprising the *Common*KADS model set have been instantiated more fully than others. In particular, the expertise task models have been developed in some detail, as these are the main focus of my research. The design model has been less fully instantiated. A characteristic of the *Common*KADS model set is that different models show significant interaction and overlap. In particular, the expertise and task models have many aspects in common.

The expertise task models are clearly based on the generic characteristics of the given case studies. In practice, expertise models at a greater level of detail (the inference and domain layers) in a KBS will be developed in an iterative manner. The initial models will be used to compare the output from a KBS with the human expert or experts the system's expertise is based on. See Duursma et al [1994] for how this might occur.

The level of abstraction of my model templates is critical in defining how useful they can be to support knowledge-based systems in these ways. If my generic model templates are too abstract, they are limited in the real support they can provide. In contrast, if the models are too tightly linked with a particular implementation scenario, the level of detail of the model templates is too highly specified and their scope of application is limited.

Hence, the next chapter looks at how my developed generic model templates for concurrent design have been instantiated using two different implementation scenarios.

12 Instantiating my generic model templates

12.1 Introduction

The previous chapter described how I have developed a number of generic model templates for the different models comprising the complete *Common*KADS model set, in order to facilitate knowledge-based support for the concurrent design process.

In order to support potential developers of KBS, my model templates must be at a suitable level of abstraction. If the models are too abstract, they will be limited in the useful support they can provide. However, if they are too detailed, the scope of application of the model templates will be limited by the level of detail.

Hence in this chapter, I reflect on the experience of instantiating my model templates with different implementation scenarios and the insights this gave me into the level of abstraction required for the templates with a view to evaluating the chosen level of abstraction. The technique of re-engineering previously documented systems has been used consistently throughout the evolution of the KADS and *Common*KADS methodologies. Essentially the case studies are benchmark systems, which can be used to validate refinements in the methodology. A well-documented example where this technique has been used is described in Kingston [1993].

I begin by discussing the two different implementation scenarios I used to refine my model templates. I then go on to describe how my generic templates were instantiated for the scenarios, the particular details of the instantiations and necessary refinements and modifications to the generic templates. I finish by discussing the implications of instantiating these scenarios and also outline some limitations I believe currently exist in the *Common*KADS methodology and the level of support it can currently provide for the concurrent design process.

The workbench assumes that a particular system's development is contained in a project. As an aid to model development, the workbench allows my generic templates for the task, communication, agent and design models to be imported from other projects and then adapted to fit the desired scenario. For the two scenarios, this involved importing the generic template and deleting any entities that were not required for the particular scenario. The remaining entities were then instantiated to suit the particular scenario. This made instantiation of the models for the scenarios a much quicker and more efficient process than would have been the case if I were developing the different models from nothing.

12.2 Two different implementation scenarios

I wished to instantiate my model templates with what I would consider to be typical, or at least representative aspects of applications where the model templates would be of use to a knowledge engineer, attempting to implement knowledge-based support for the concurrent design process. Hence I have instantiated two contrasting scenarios, with differing requirements, in order to evaluate and validate my generic model templates. In my descriptions of these scenarios, I have concentrated on areas where my instantiations of the model templates either expand on, or differ from, the generic templates.

12.2.1 Scenario 1

Scenario 1 is based on a situation that was encountered through work done with a company as part of case study 3. The company currently employs a designer (Peter) who produces designs, using a 3D CAD system, for electric vehicles. He is assisted in this by two other people, one a manufacturing and assembly expert (Jeremy) and the other an electrical and electronics expert. Both these experts assist the designer by critiquing his designs whenever asked. These three personnel are all under the management of Paul, the production controller. However, the electronics expert is due to leave the company and so the company will no longer have access to his expertise. The proposed scenario is to have a knowledge-based system as an extension of the CAD system, encapsulating his relevant electronics expertise as it relates to the design of electric vehicles. This would then be used to critique evolving designs from an electronic engineering perspective at the designer's request. I envisage a negotiation strategy of appeasement, in that unfavourable critiques from the critiquing agents will result in a revised design proposal from the designer followed by revised critiques and alternative proposals from critiquing agents. However, a clear problem with this form of negotiation is that concensus may never be reached or may be approached very slowly. Hence, as a pragmatic modification of this approach, I propose that the human production director, Paul, can intervene as an arbitrator to unilaterally determine any negotiation that may occur. This approach also has implications for the agents' autonomy and the goals for tasks, as outlined when I discuss these models.

12.2.2 Scenario 2

Scenario 2 is based on an existing system called I3D, described in detail by Victor et al [1993]. The system uses discrete, intelligent expert systems to aid in the design of powder ceramic components. Findings from this research have influenced later work on Single Function Agents (Brown et al [1995]). The system interacts with a designer sitting at a workstation in a similar way to that envisaged for scenario 1. As the designer makes design decisions, the system provides feedback about the system from several different points of view. It is interesting to note the 'roles' that the different experts systems comprising the complete I3D system play. These are:

- Advisor: Suggests a portion of the design.
- Critic: Comments on possible problems with a given design.
- Planner: Can provide information on processing sequences.
- Selector: Picks items from a list such as material or process.
- Estimator: Estimates derived values, such as cost.

As such, these combined roles can be seen as analogous to the more encompassing critiquing role I have developed up to now in this thesis. The 'aspects' or 'perspectives' the expert systems can comment on are material, manufacturing process, inspection, cost and reliability. These roles complement and expand on the perspectives I have described as being generic from my case studies in Chapter 11.

12.3 Instantiating Scenario 1

I will now describe how the generic model templates have been instantiated for the scenario 1. This models critiquing knowledge (as a design critic on Peter the designer's workstation) of the electronics life-cycle perspective.

12.3.1 Organisation model for scenario 1

The problems and opportunities, together with the organisational context of scenario 1 are outlined in table 12.1. The principal opportunity that can be instantiated from my generic template is the fact that expertise is to be lost to the organisation. The development of a computer-based critic is an opportunity to alleviate this problem.

name	description
Opportunities	
Loss_of_expertise	The loss of electronics expertise to the organisation can be alleviated by the opportunity of developing a knowledge-based system, in the form of a design critic.
Organisational context	
Risks of changing dynamics	The development of a computer-based design critic will have an affect on the existing dynamics of the design process within the organisatiion.

Table 12.1: Problems, opportunities and the organisational context of the organisation model for scenario 1.

12.3.1.1 Variant constituents for Scenario 1

The different constituents comprising the proposed solution are now discussed.

12.3.1.2 People

I have outlined three distinct human stakeholders who play a part in the proposed scenario. The role and links to the agent models for the stakeholders are outlined in Table 12.2. It must be noted that each stakeholder may play the part of more than one agent, as illustrated by the assembly and manufacture critiquing agent who plays the part of a critiquing agent from a number of life-cycle perspectives and also that of a negotiating agent.

name	Paul
description	Controlling agent – production director
links to agent	age_Control
requires_knowledge	Control_knowledge, negotiate_knowledge
role	Control
name	Peter
description	Agent with the ability to propose designs
links to agent	age_Propose
requires_knowledge	Propose_knowledge, negotiate_knowledge
role	Propose, Negotiate
name	Jeremy
description	Agent with the ability to critique designs from an assembly and manufacture perspective
links to agent	age_Critique_Assembly, age_Critique_manufacture
requires_knowledge	Assembly_critiquing_knowledge, Manufacture_critiquing_knowledge, negotiate_knowledge
role	Critique

Table 12.2: The properties of the different stakeholders in the instantiation of scenario 1.

12.3.1.3 Structures

In terms of structure, organisation 1 attempts to group resources around design projects rather than have distinct functionally-oriented departments. As a result, the organisational structure is derived from my generic template for a project-oriented organisational structure, as illustrated in figure 12.1.

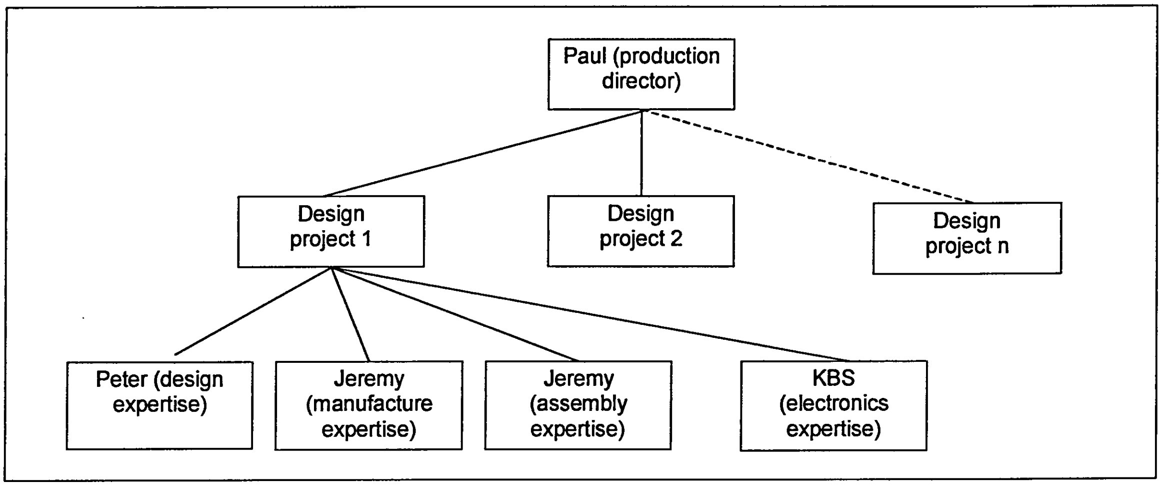

Paul (production director)
Design project 1
Design project 2
Design project n
Peter (design expertise)
Jeremy (manufacture expertise)
Jeremy (assembly expertise)
KBS (electronics expertise)

Figure 12.1: The organisational structure of organisation 1.

This shows the proposed KBS solution with personnel and resources grouped around different design projects.

12.3.1.4 Additional constituents of the organisation model

The additional constituents of the organisation model for scenario 1 are outlined in table 12.3.

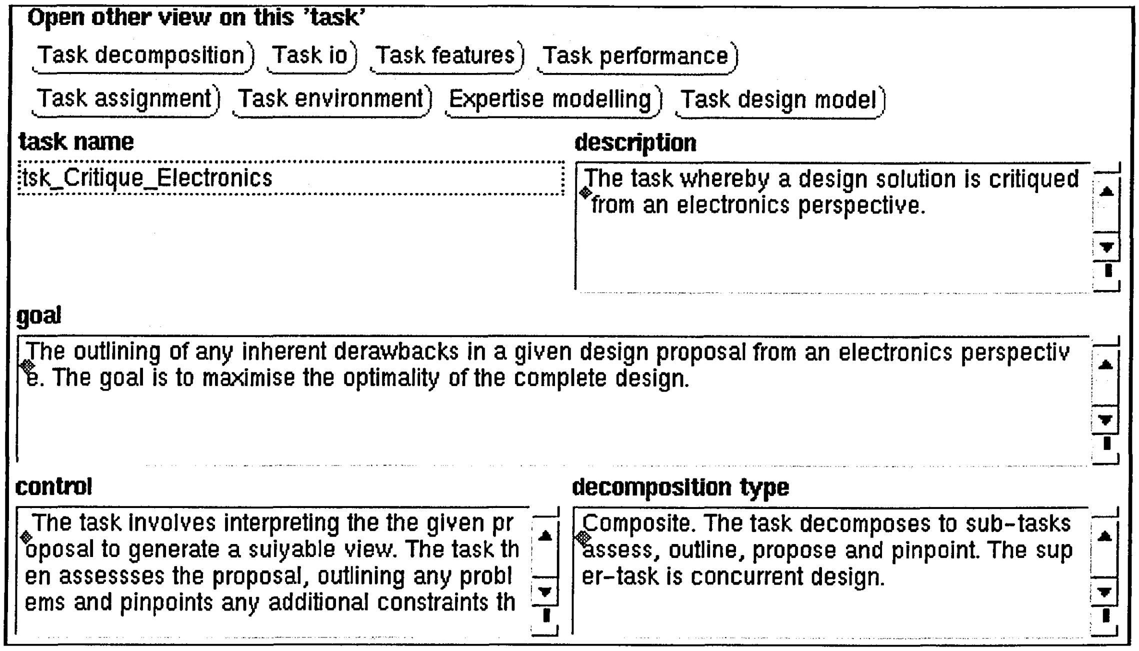
Knowledge items
The knowledge item to be computationally modelled is electronics_critiquing_knowledge.
Computing resources
The computing resources on which the system is to be implemented features the server-based network currently in place in the organisation and specifically, the CAD workstation on which the designer generates designs.
Electronics_critiquing_agent. This is the machine-based agent implemented on the workstation.
Functions
The functions present in the organisation are fnc_Negotiate, fnc_Propose, fnc_Assembly_Critique, fnc_Manufacture_Critique, fnc_Electronics_critique and fnc_Control.
Processes
Although they are considering ISO 9000 accreditation, the organisational procedures and methods have not yet been formalised according to this standard.
Power authority
The power authority links are relatively simple. The production director influences both the designer and the assembly and manufacture critiquing expert. The designer has effective control over the KBS agent, in that the designer can choose to accept or reject the recommendations of the agent.

Table 12.3: Additional constituents of the organisation model for scenario 1.

12.3.2 Task model for scenario 1

The main task to be performed by the integrated design solution is the task of concurrent design. The identified sub-tasks implied by this and instantiated from my generic task model template are tsk_propose, tsk_critique_assembly, tsk_critique_manufacture, tsk_critique_electronics and tsk_negotiate. Most of the attributes for each task have been defined in the generic model. The main task during instantiation is task assignment - I.e. which agent will perform which tasks.

For this scenario, the ‘propose’, ‘negotiate’ and some ‘critiquing’ tasks are performed by human agents, so the reasoning capabilities required for these tasks are defined in the task model. However, the electronics critiquing task is performed by a machine based agent and so the reasoning capabilities for this agent are defined in the critiquing and negotiation expertise models referenced by the electronics-critiquing agent. The major characteristics of the task tsk_critique_electronics are outlined in table 12.4. I have defined the task goal of the tsk_Critique_Electronics as being to maximise the optimality of the complete product life-cycle. In an implementation, this goal would be variable during a typical design process, from the maximisation of the critiquing agents' own position to the maximisation of the optimality of the product life-cycle. The limitations of rigidly defining task goals in this way are discussed in a later section.



task name

tsk_Critique_Electronics

description

The task whereby a design solution is critiqued from an electronics perspective.

goal

The outlining of any inherent derawbacks in a given design proposal from an electronics perspectiv e. The goal is to maximise the optimality of the complete design.

control

The task involves interpreting the the given pr oposal to generate a suiyable view. The task th en assessses the proposal, outlining any probl ems and pinpoints any additional constraints th

decomposition type

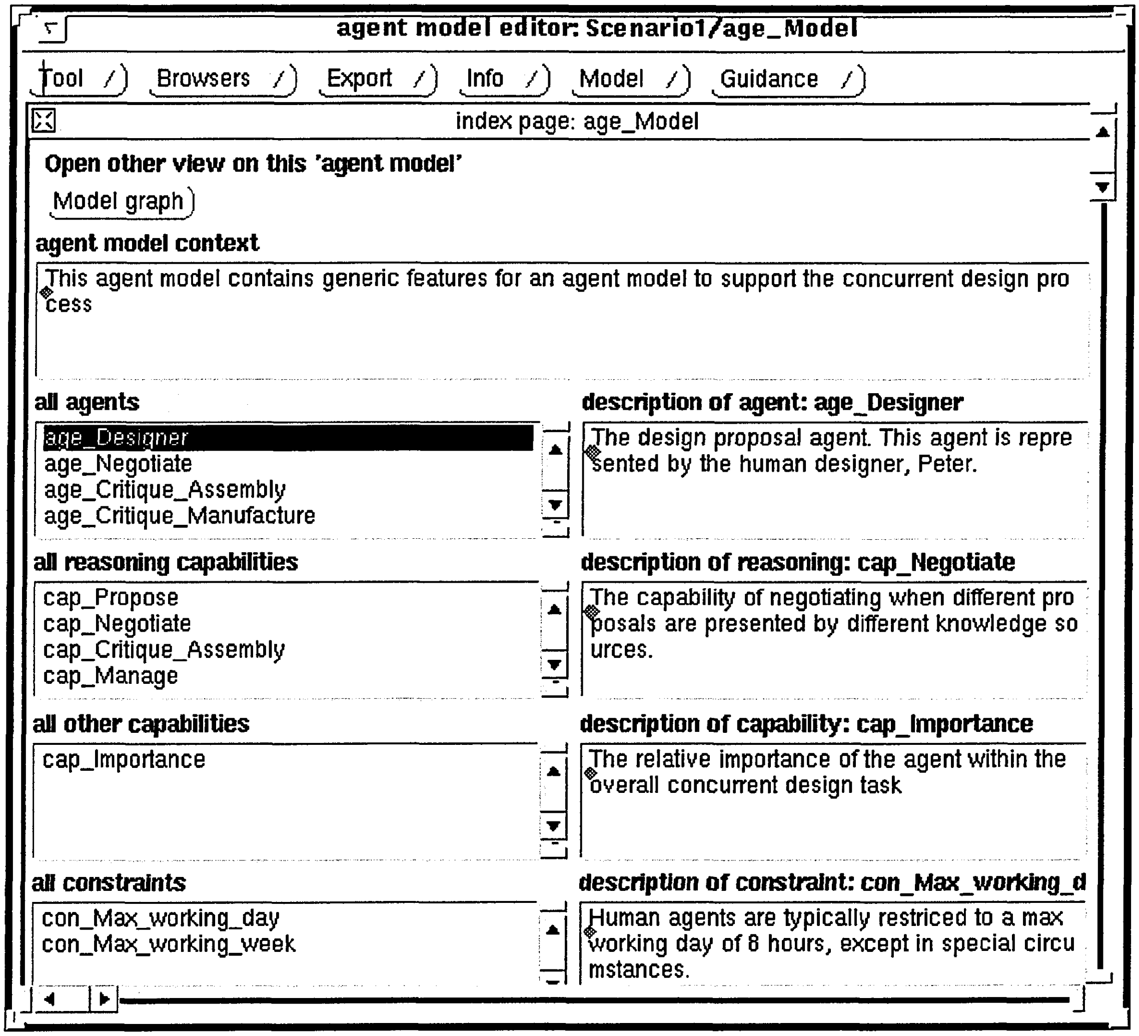
Composite. The task decomposes to sub-tasks assess, outline, propose and pinpoint. The sup er-task is concurrent design.

Table 12.4: The major characteristics for the task tsk_Critique_electronics in scenario 1.

The goal of the task will be defined by how autonomous the critiquing agent is. Because the agent autonomy needs to be adjustable, based on the negotiation model employed, this will affect the task goal, which therefore also needs to be adjustable based on a particular negotiation stance.

12.3.3 Agent model for scenario 1

For this scenario, the human agents are the production controller, designer and manufacture and assembly expert. The only machine based agent is the electronics critiquing agent. Hence the agents required from my generic model template (and the corresponding agents they relate to) are age_Designer (Peter), age_Critique_assembly (Jeremy), age_Critique_manufacture (Jeremy), age_Critique_Electronics (machine based agent) and age_Control (Paul). These are illustrated in Table 12.5.


agent model editor: Scenario1/age_Model
Tool
Browsers
Export
Info
Model
Guidance
index page: age_Model
Open other view on this 'agent model'
Model graph

agent model context

This agent model contains generic features for an agent model to support the concurrent design process

all agents

- age_Designer
- age_Negotiate
- age_Critique_Assembly
- age_Critique_Manufacture

description of agent: age_Designer

The design proposal agent. This agent is represented by the human designer, Peter.

all reasoning capabilities

- cap_Propose
- cap_Negotiate
- cap_Critique_Assembly
- cap_Manage

description of reasoning: cap_Negotiate

The capability of negotiating when different proposals are presented by different knowledge sources.

all other capabilities

- cap_Importance

description of capability: cap_Importance

The relative importance of the agent within the overall concurrent design task

all constraints

- con_Max_working_day
- con_Max_working_week

description of constraint: con_Max_working_d

Human agents are typically restriced to a max working day of 8 hours, except in special circumstances.

Table 12.5: Scenario 1 agent model.
This outlines the entities comprising the agent model for scenario 1. Each required agent from my generic model template is defined as being either human or machine-based in nature.

The main issues that are relevant from the agent point of view are the fact that the human critiquing agent is not always available to the designer in order to critique designs while the machine-based agent is theoretically available all the time. The relative importance of the critiquing agents is modelled as being the same (i.e. the designer will consider the critiques from the two separate sources as being equally important). Table 12.6 illustrates how the major characteristics from my generic model template for the electronics critiquing have been instantiated for this particular scenario.

name	description
type	Machine based
role	Critiquing
organisational position	KBS Critiquing agent
performs tasks	tsk_Critique_electronics
needs reasoning capabilities	cap_Critique_assembly
needs other capabilities	None
agent as comp resource	KBS agent
agent as stakeholder	None
Autonomy	0
Importance	5

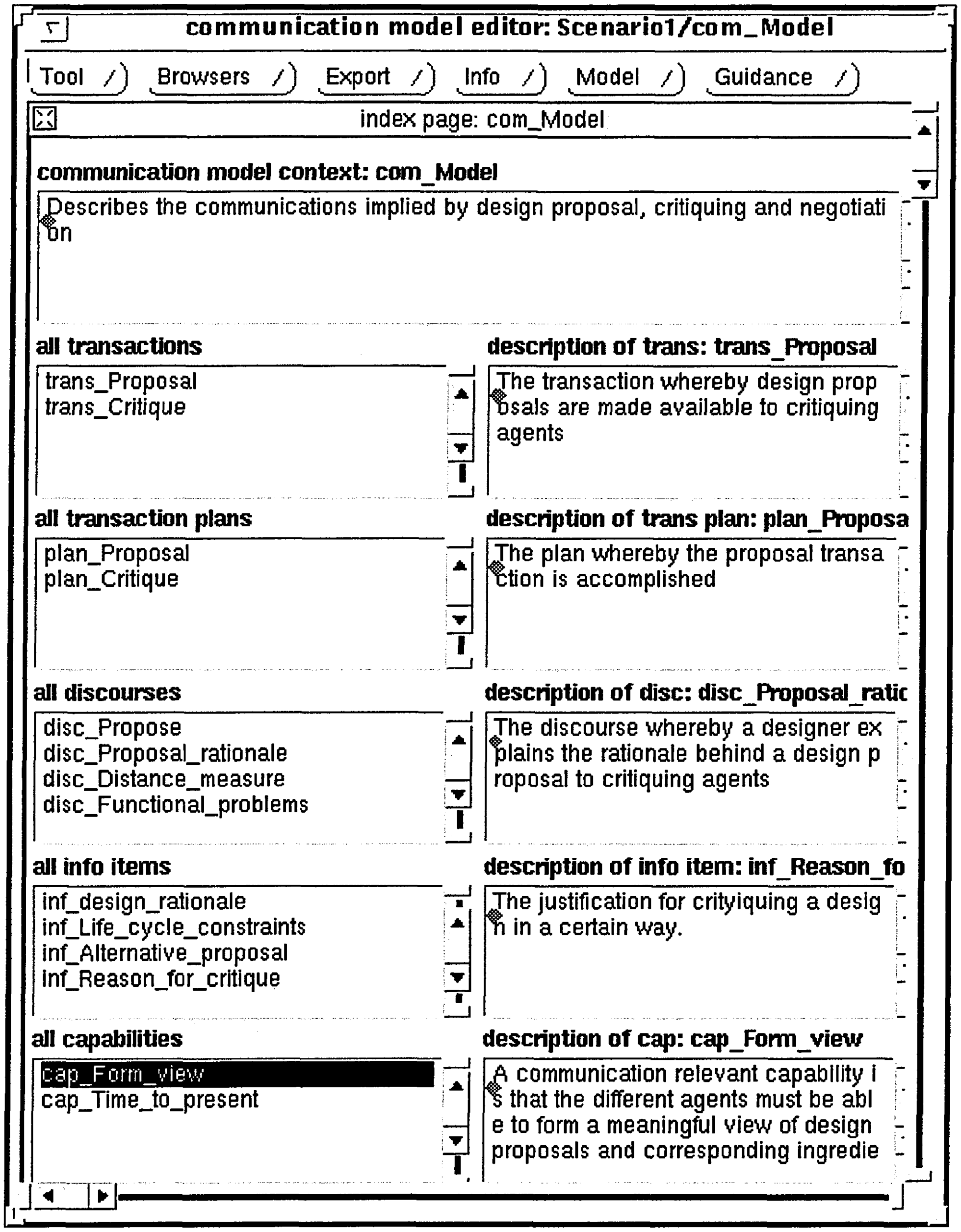
Table 12.6: Important properties of the electronics_critiquing_agent, modelled as a KBS within the given solution.
This shows how the electronics critiquing agent is to be implemented as a machine based agent to perform the electronics critiquing task.

The capabilities for the designer are the ability to generate design proposals together with a limited ability to be able to reflect on the implications of the design proposals from given perspectives. The manufacture and assembly critiquing agent must have the capability to critique evolving designs. The expert critiquing capability of the machine-based electronics agent to critique designs is defined in the expertise model for the agent.

A critical issue for this scenario is the autonomy of the agent, which dictates how 'self interested' the agent is. Essentially the agent autonomy, which has an influence on the goal of the task performed by the agent, needs to be adjustable to account for any negotiation process that may occur.

12.3.4 Communication Model for scenario 1

The main features of this communication model are the transfer of design proposals to the proposed system and the corresponding output of critiques from the system. The form of negotiation envisaged has the designer and critiquing agents making concessionary proposals to each other. However, in terms of the communication involved, this does not involve any additional transactions. A modified design proposal is input to the system (by the designer) followed by a modified critique (from the critiquing agents). Hence there are no additional communication transfers defined by this mode of negotiation. The communication transactions defined for scenario 1, instantiated from my generic communication model are outlined in Table 12.7.



communication model context: com_Model

Describes the communications implied by design proposal, critiquing and negotiati
on

all transactions

trans_Proposal
trans_Critique

description of trans: trans_Proposal

The transaction whereby design prop
osals are made available to critiquing
agents

all transaction plans

plan_Proposal
plan_Critique

description of trans plan: plan_Proposa

The plan whereby the proposal transa
ction is accomplished

all discourses

disc_Propose
disc_Proposal_rationale
disc_Distance_measure
disc_Functional_problems

description of disc: disc_Proposal_ratic

The discourse whereby a designer ex
plains the rationale behind a design p
roposal to critiquing agents

all info items

inf_design_rationale
inf_Life_cycle_constraints
inf_Alternative_proposal
inf_Reason_for_critique

description of info item: inf_Reason_fo

The justification for crityiquing a desig
n in a certain way.

all capabilities

cap_Form_view
cap_Time_to_present

description of cap: cap_Form_view

A communication relevant capability i
s that the different agents must be abl
e to form a meaningful view of design
proposals and corresponding ingredie

Table 12.7: The communication model transactions for scenario 1
The communication mode defined by this scenario is based on a negotiation model of concession. As a result, the communication transactions and corresponding plans are based around transferring design proposals and critiques between agents.

The initiatives for the different transactions are defined by the agents concerned. The transfer of design proposals to the proposed system is under the control of the designer, while the corresponding return of critiques is controlled by the critiquing agents.

The expertise model tasks defining how the transactions occur are defined by the transfer_proposal and transfer_critique tasks. These are knowledge-intensive tasks in their own right and would be implemented by separate sub-systems comprising an implemented solution. In order to illustrate this, it is necessary to consider how such sub-systems would be implemented.

The designer is required to input his or her design proposals into the system in a particular format. For the scenario, this is defined to be in the form of a CAD drawing or a textual description. The system would then need to be capable of representing the presented design proposal in a format suitable for critiquing by the internal computational agents. The resulting critiques output from the critiquing agents would need to be parsed in a similar manner.

Hence this outlines the need for complex computational sub-systems which have the capability to parse concepts such as design solutions, critiques etc into formats or views that are meaningful to the participating agents. This is far from a trivial task and implies that the design of such sub-systems is a complex and time-consuming process. The architecture and design of such sub-system will be greatly influenced by the format and structure of domain knowledge in the domain.

The transactions trans_Proposal and trans_Critique are therefore defined by knowledge intensive tasks in the expertise model for this scenario (transfer_Proposal and transfer_Critique respectively) and would be implemented by sub-systems sub_Transfer_Propose and sub_Transfer_Critique defined in the design model instantiation.

12.3.5 Expertise model for scenario 1

The expertise-intensive tasks modelled are the critiquing and negotiation ability of the electronics agent, which are based on my generic expertise task model for critiquing and the 'appeasement' form of negotiation. The propose task, together with additional negotiation tasks are performed by human agents and are outlined in the task model. The transfer tasks are for critiquing and negotiation. The expertise task model for the electronics-critiquing agent is presented in figure 12.3.

The critiquing task decomposes to sub-tasks from my generic critiquing model, which I have previously outlined in section 10.4. The negotiation task decomposes into two tasks, assess and propose (I have identified these as tasks assess_negotiate and propose_negotiate on the *Common*KADS workbench in order to differentiate them from the assess and propose sub-tasks used in other tasks). In practice, the propose_negotiate task is a specialisation of the generic propose task I have outlined, with the inclusion of an additional input roles 'additional constraints' and 'alternative proposal' to highlight the concerns of other agents (see Figure 10.11).

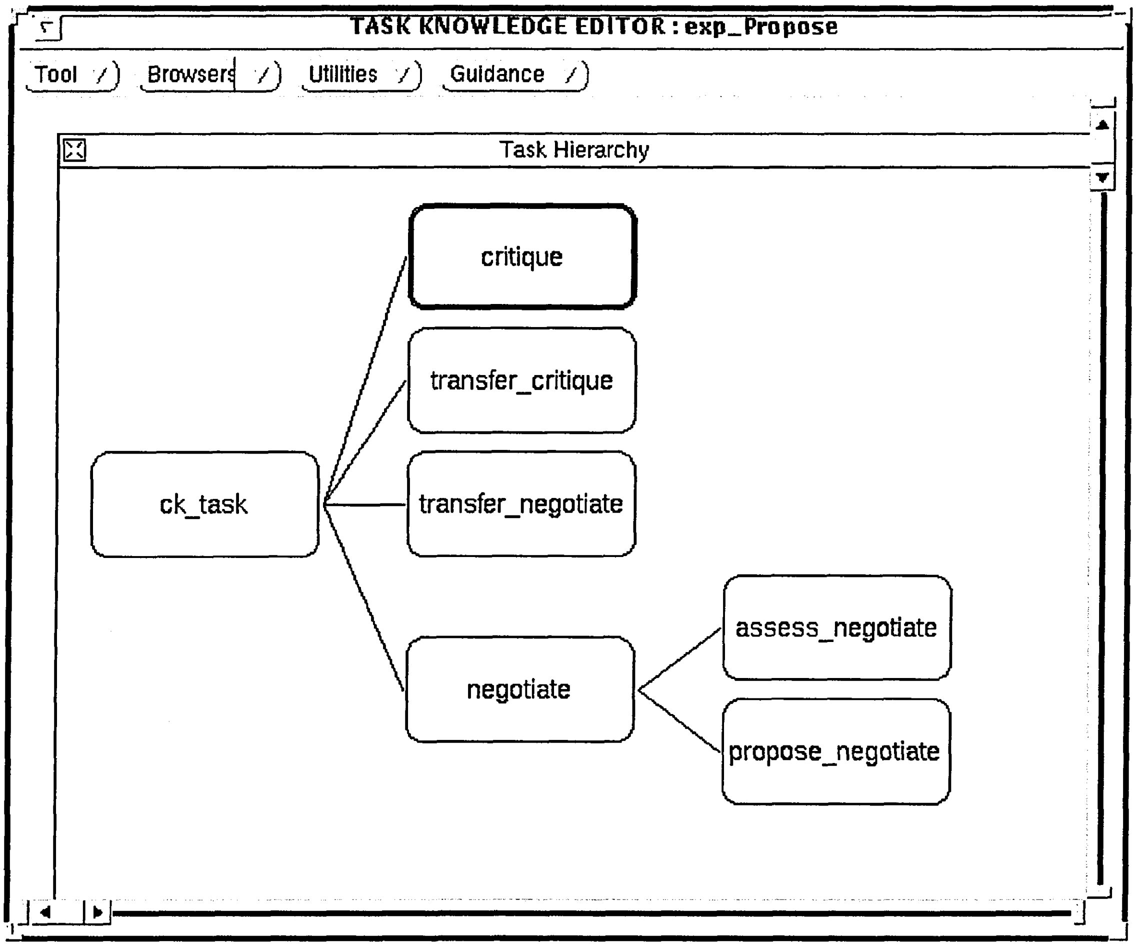


Figure 12.3: The overall expertise task model employed by the electronics critiquing agent.

This features four high-level tasks, the actual critiquing and negotiation tasks and the transfer tasks where critiques and negotiation information is transferred to the design proposer.

The control structure for this scenario can be considered as design proposal followed by critique. There will then be a round of negotiation (consisting of a series of appeasement-type counter proposals) until this is terminated by the (human) controller. The knowledge role 'Task goal' will play an important role in both the critiquing and propose_negotiate tasks as this will change based on the stage of the negotiation. In the CML format advocated by *Common*KADS, this can be represented as:

```
control structure:
propose(Database of previous designs + Design specification + Conceptual design + Current design + Designers knowledge of downstream constraints-> Design proposal)
critique(Task goal + Current design + Design proposal + Design specification + Life-cycle constraints -> Additional constraints + Functional problems + Assessment + Alternative proposal + Additional life-cycle constraints)

Do until terminated:
assess_negotiate(Additional constraints + Functional problems + Assessment + Alternative proposal + Additional life-cycle constraints + Importance of critiquer -> Implications for design)
propose_negotiate(Task goal + Additional constraints + Alternative proposal + Current design -> Revised design + Revised design specification
```

12.4 Instantiating Scenario 2

I will now describe how my generic model templates have been instantiated for scenario 2. As outlined in section 12.2, this scenario is based on an existing system, the I3D system, described in Victor et al [1993]. Throughout, I will describe the system as it fits the architecture of the *Common*KADS model set and my instantiation of my generic templates on the *Common*KADS workbench. In addition, I have not attempted to instantiate the I3D system in its entirety, I have concentrated on the knowledge-based aspects of the system. Unattributed quotes (in italics) are from Victor et al [1993].

12.4.1 Organisation model for scenario 2

The I3D system was originally developed as a research tool and as a result, there is no single organisational context that the system is intended to be used in. However, a number of general organisational issues are evident from the description of the system.

'An integrated design system is needed by the powder ceramics processing industry in order to make significant improvements in the areas of production quality and cost, customer responsiveness and reduced cycle times as well as in improved product functionality and manufacturability.'

Ceramic materials are becoming increasingly important for both defence and industrial applications. The manufacturing processes for these types of powder ceramic materials have lagged behind their industrial use. The design and planning systems developed ... will advance the manufacturing capabilities of the ceramics processing industry.'

As a result, the opportunities inherent in the development of the I3D system are instantiated from my generic descriptors.

name	description
Opportunities	
Improved_design_quality	By bringing life-cycle perspectives to bear on the ceramic design process, product functionality and manufacturability can be improved.
Reduced_design_time	By bringing life-cycle perspective knowledge to bear on the ceramic design process, the overall design time can be reduced.
Reduced_life_cycle_costs	An integrated design system in the ceramics processing industry can result in reduced life-cycle costs.

Table 12.8: The organisational context of the organisation model for scenario 2. This illustrates how a number of opportunities can be gained by implementing knowledge-based support for the ceramic design process.

The different constituents comprising the proposed solution are now discussed.

12.4.1.1 People

Only one human stakeholder plays a part in the scenario, the designer. As such, the human designer plays both a design proposal role and also a controlling role for the overall design process. The role and links to the agent model for this human stakeholder are illustrated in Table 12.9.

name	Designer
description	Controlling agent – designer
links to agent	age_Designer
requires_knowledge	Propose_knowledge, Control_knowledge
role	Propose, Control

Table 12.9: The properties of the Designer stakeholder in the organisational model for scenario 2.

12.4.1.2 Processes

The design process inherent in the I3D system is split into three main stages. These are requirements definition, where the designer enters the functional descriptors which characterise the design. In the conceptual design stage, the overall features of the design are generated. In the detailed design phase, exact criteria are specified. *'As design decisions are made, the system provides feedback about the design from several points of view'.* As a result, the overall process implemented in the system is characterised by my view of concurrent design, previously described and illustrated in section 10.3.

12.4.1.3 Power authority

The I3D system is under the control of the designer - *'to allow the user to override decisions reached by agents. I.e. the user was in control of resolving any conflicts between the agents and user.'*

12.4.1.4 Additional constituents of the organisation model for scenario 2

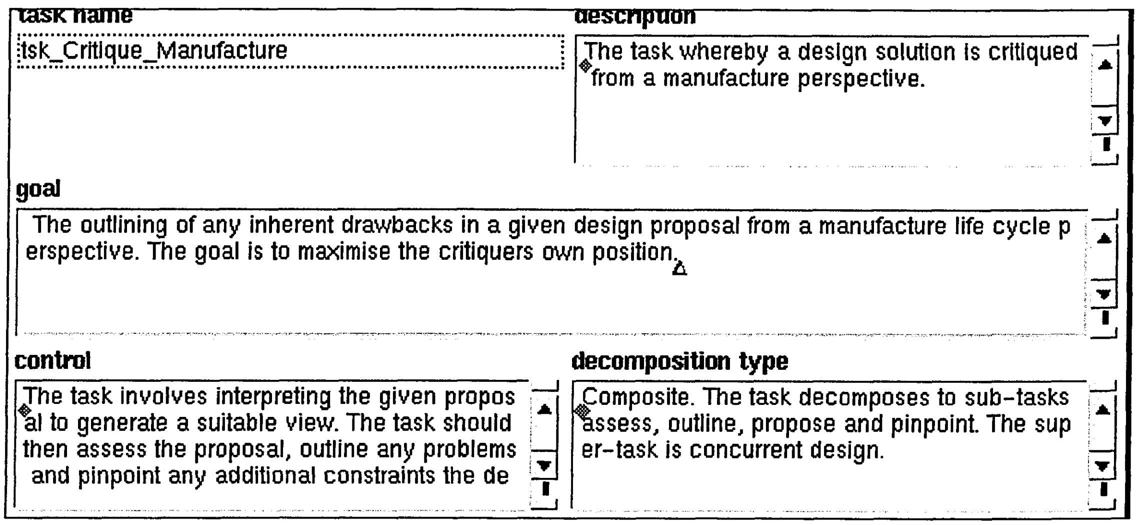
Additional constituents of the organisation model for scenario 2 have been instantiated from my generic organisational model template and are illustrated in table 12.10. A number of additional computing resources (FMEA and simulation systems) together with additional computer-based agents and their functionality (reliability and inspection experts) complement and expand on my generic model template for the organisational model.

Knowledge items
Manufacture_critiquing_knowledge, cost_critiquing_knowledge, material_critiquing_knowledge, inspection_critiquing_knowledge and reliability_critiquing_knowledge.
Computing resources
The essential computing resource on which the system is implemented features a high-level workstation on which the system runs. This features a solid modeller CAD system (CATIA) with interfaces to the different expert systems.
Manufacture _critiquing_agent: Machine based agent which critiques designs from a manufacture perspective.
Cost_critiquing_agent: Machine based agent which critiques designs from a cost perspective.
Material_critiquing_agent: Machine based agent which critiques designs from a materials perspective.
Inspection_critiquing_agent: Machine based agent which critiques designs from an inspection perspective.
Reliability_critiquing_agent: Machine based agent which critiques designs from a reliability perspective.
FMEA system
Simulation systems
Structures
No specific organisational structure is specified for the system
Functions
Fnc_Manufacture_Critique, fnc_Cost_critique, fnc_Material_critique, fnc_Inspection_critique and fnc_Reliability_critique.

Table 12.10: Additional constituents of the organisation model for scenario 2.

12.4.2 Task Model for scenario 2

The main tasks performed by the I3D system are manufacture critiquing, cost critiquing, material critiquing, inspection critiquing and reliability critiquing. These are represented by tasks tsk_Manufacture_critique, tsk_Cost_critique, and tsk_Material_critique from my generic model template. In addition, two extra tasks, tsk_Inspection_critique and tsk_Reliability_critique are instantiated. The negotiation mode is conflicting in that the goal of each critiquing agent's task is to maximise it's own position, however the designer effectively has control. All task capabilities within this scenario are realised by the relevant expertise model for the agent. As an example, the major characteristics of the task tsk_critique_manufacture are outlined in table 12.11.



task name

tsk_Critique_Manufacture

description

The task whereby a design solution is critiqued from a manufacture perspective.

goal

The outlining of any inherent drawbacks in a given design proposal from a manufacture life cycle perspective. The goal is to maximise the critiquers own position.

control

The task involves interpreting the given proposal to generate a suitable view. The task should then assess the proposal, outline any problems and pinpoint any additional constraints the de

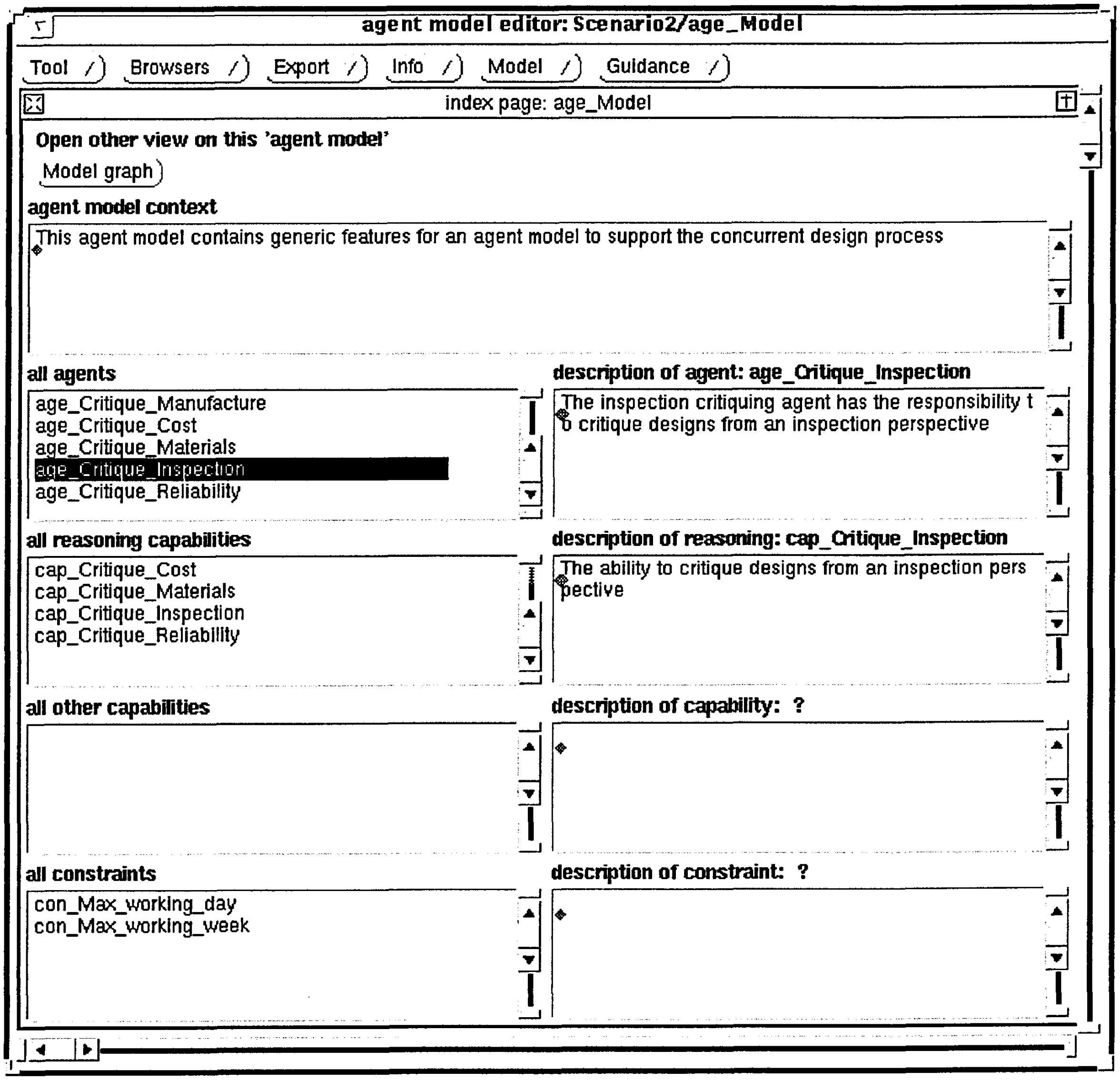
decomposition type

Composite. The task decomposes to sub-tasks assess, outline, propose and pinpoint. The super-task is concurrent design.

Table 12.11: The major characteristics for the task tsk_manufacture_critique.
The control and decomposition for the task show how the given task is decomposed into sub-tasks and the sequencing of these sub-tasks.

12.4.3 Agent model for scenario 2

For this scenario, the only human agent is the senior designer. The machine based agents are the manufacture expert, cost expert, materials expert, inspection expert and reliability expert. As a result, the constituents of the agent model are instantiated from the generic constituents of my model template for the agent model but also expand on my generic model with the inspection and reliability experts. The designer agent has propose and negotiation capabilities while the critiquing agents have the capability to critique. The agents instantiated for this scenario from my generic model template are illustrated in Table 12.12 (the constraints outlined in the table clearly only apply to ‘human’ agents).



Open other view on this 'agent model'

agent model context

This agent model contains generic features for an agent model to support the concurrent design process

all agents

- age_Critique_Manufacture
- age_Critique_Cost
- age_Critique_Materials
- age_Critique_Inspection
- age_Critique_Reliability

description of agent: age_Critique_Inspection

The inspection critiquing agent has the responsibility t o critique designs from an inspection perspective

all reasoning capabilities

- cap_Critique_Cost
- cap_Critique_Materials
- cap_Critique_Inspection
- cap_Critique_Reliability

description of reasoning: cap_Critique_Inspection

The ability to critique designs from an inspection pers pective

all other capabilities

description of capability: ?

all constraints

- con_Max_working_day
- con_Max_working_week

description of constraint: ?

Table 12.12: Scenario 2 agent model.
This outlines the entities comprising the agent model for scenario 2.

The properties of the manufacture-critiquing agent are outlined in table 12.13.

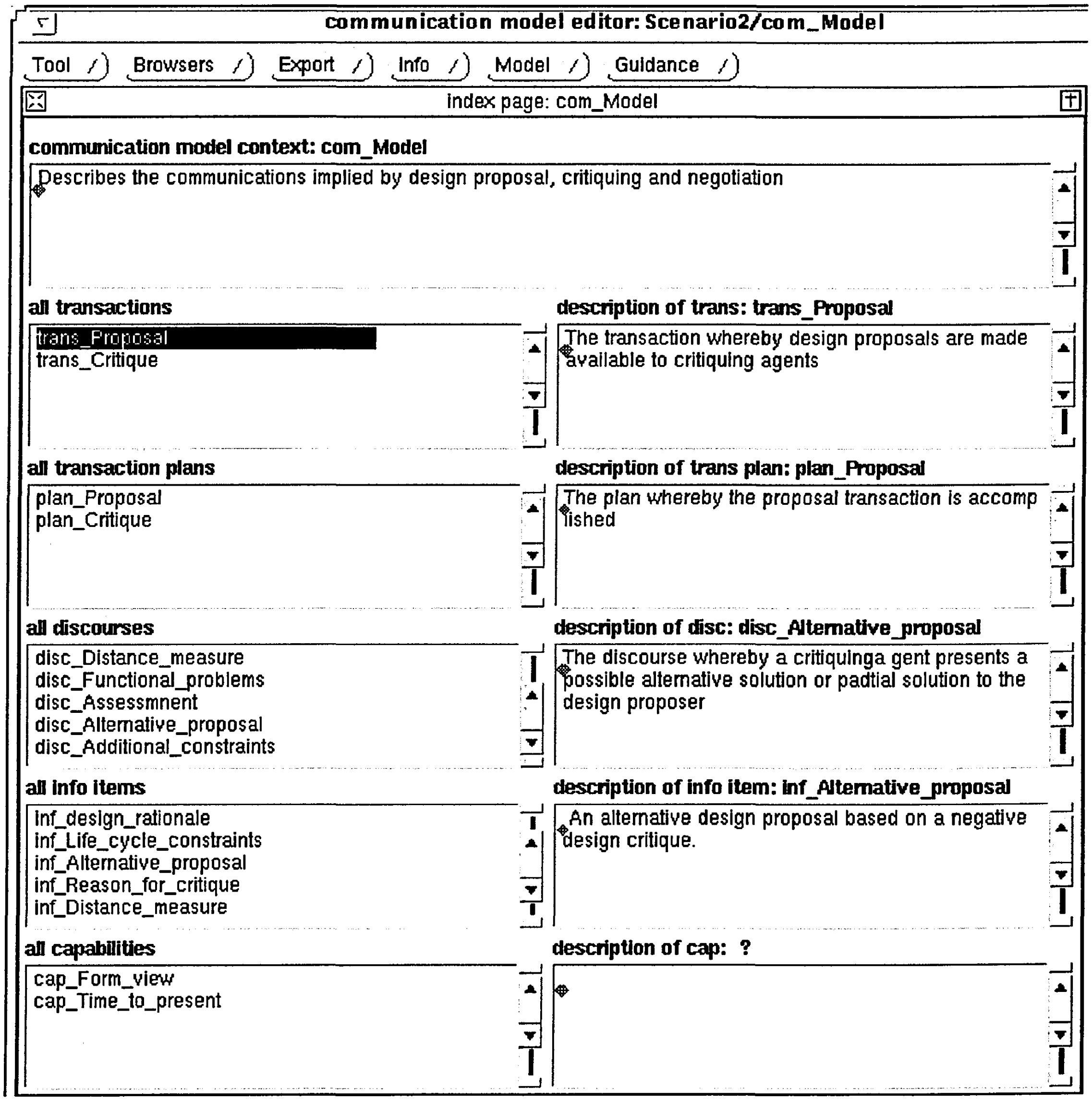
name	description
type	Machine based
role	Critiquing
organisational position	KBS Critiquing agent
performs tasks	tsk_Critique_manufacture
needs reasoning capabilities	cap_Critique_manufacture
needs other capabilities	None
agent as comp resource	KBS agent
agent as stakeholder	None
Autonomy	10
Importance	5

Table 12.13: Important properties of the manufacture_critiquing_agent.
This is modelled as a KBS within scenario 2 and shows how particular properties of the generic agent model have been instantiated for the manufacture critiquing agent.

An important consideration for this scenario are the relative importance and autonomy of each agent. All agents are modelled as being of equal importance, hence all agents have an importance rating of 5. The autonomy of each agent dictates how free the agent is to pursue their own goals. As I have defined the agents as being self-interested, the autonomy of each agent is defined as 10 on my numerical scale.

12.4.4 Communication Model for scenario 2

The main ingredients exchanged between agents are design proposals and corresponding critiques. The initiative for the communication between agents is controlled by the designer agent. The transactions defined for the communication model of scenario 2 are outlined in table 12.14.



communication model context: com_Model

Describes the communications implied by design proposal, critiquing and negotiation

all transactions	**description of trans: trans_Proposal**
trans_Proposal trans_Critique	The transaction whereby design proposals are made available to critiquing agents

all transaction plans	**description of trans plan: plan_Proposal**
plan_Proposal plan_Critique	The plan whereby the proposal transaction is accomp lished

all discourses	**description of disc: disc_Alternative_proposal**
disc_Distance_measure disc_Functional_problems disc_Assessmnent disc_Alternative_proposal disc_Additional_constraints	The discourse whereby a critiquinga gent presents a possible alternative solution or padtial solution to the design proposer

all info items	**description of info item: inf_Alternative_proposal**
inf_design_rationale inf_Life_cycle_constraints inf_Alternative_proposal inf_Reason_for_critique inf_Distance_measure	An alternative design proposal based on a negative design critique.

all capabilities	**description of cap: ?**
cap_Form_view cap_Time_to_present	

Table 12.14: The communication model transactions for scenario 2.
For this scenario, the two essential transactions are the communication of design proposals and critiques between the agents comprising the system.

12.4.5 Expertise model for scenario 2

The I3D system encapsulates both the analysis (requirements definition) and synthesis (solution generation) phases of the complete design cycle. I have concentrated on the instantiation of the synthesis stage of the concurrent design process.

The expertise model has been instantiated down to the level of the task layer and the necessary associated knowledge roles. The expertise task models for this scenario are manufacture critiquing, cost critiquing, materials critiquing, inspection critiquing, reliability critiquing and negotiation. The generic critiquing task model has been instantiated for each agent. The negotiation model assumes a problem-solving method of unilateral decision is applied (by the designer). The transfer tasks necessary are derived from my generic model, these are transfer_propose (from designer to system) and transfer_critique (from critics to designer).

As a result, the overall task model employed in each expertise task model is essentially similar to that for the manufacture critiquing agent, illustrated in Figure 12.4.

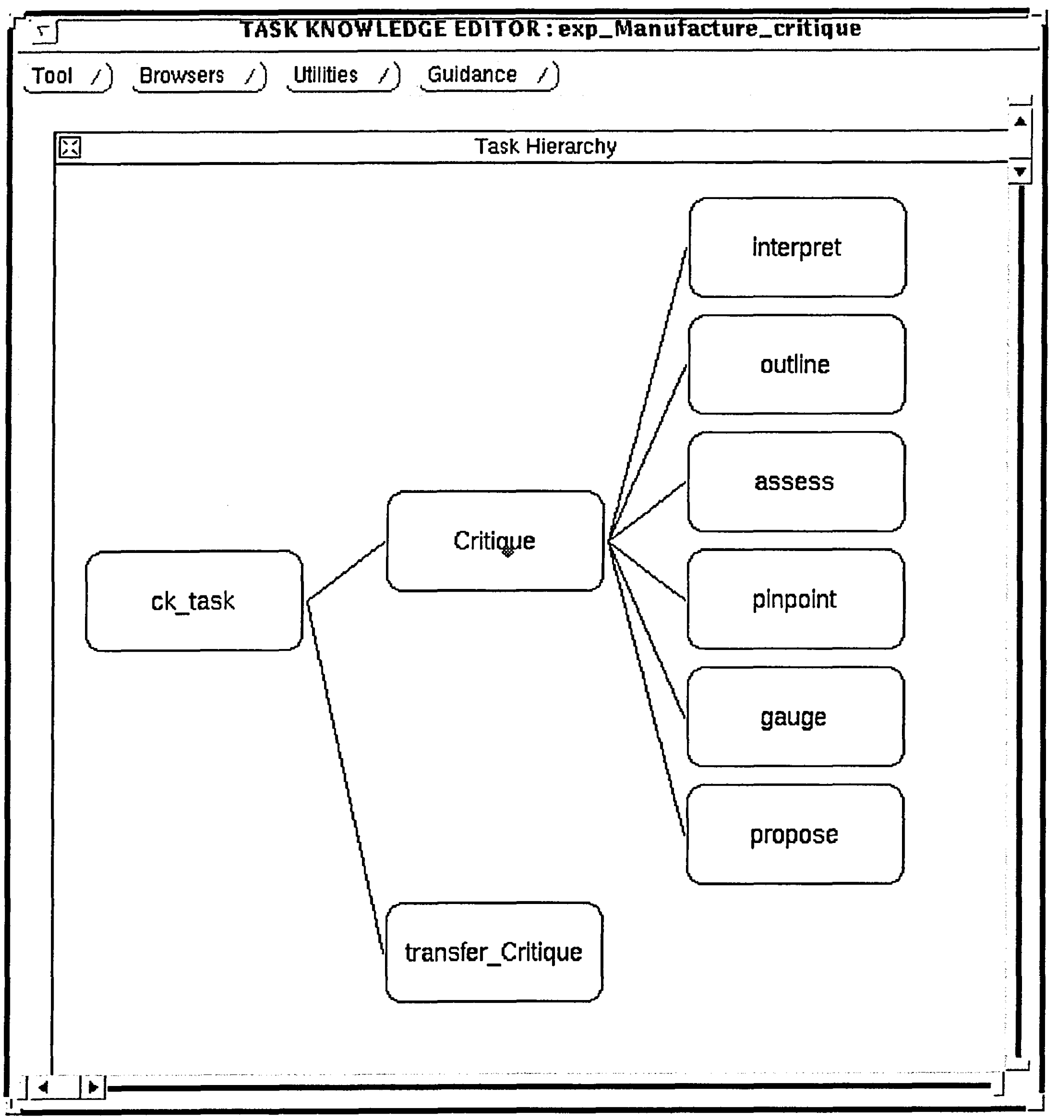


Figure 12.4: The overall expertise task model employed by the manufacture critiquing agent.

This features two high-level tasks, the actual critiquing task and the transfer task whereby the critique is transferred to the design proposer.

A critical issue in the I3D system is the control structure imposed on the concurrent design process – *'... a strict control regime, such that all agents were fired as a group (i.e. after the users request for analysis), in a pre-determined sequence, with each agent coming after those on which it depends.'* In this way, each agent is working with the current version of the evolving design and complex negotiational issues are avoided.

12.5 *Common*KADS Design models for scenario 1 and 2

Because no actual KBS are being implemented in either scenario, the design models have not been fully instantiated for the scenarios. However for both scenarios, I describe some of the features and issues involved in instantiating the design model, particularly the actual implementation of the I3D system and it's instantiation as a *Common*KADS design model.

12.5.1 Scenario 1

For scenario 1 an 'agents' based implementation would appear to offer a number of attractive features. Lander [1997] outlines a number of different systems where such architectures have been integrated, the advantages of this type of approach and looks at issues involved with implementing agents within multi agent design systems (MADS). These include the communication protocols between different agents. In my model template set, these are covered by the protocols specified in the communication model.

The application specification consists of a high-level decomposition to sub-systems, which can be KBS, interaction and any other required systems. Two decomposition paradigms suggested by *Common*KADS are function and object orientation. Because of the support most commercial software tools now offer to an object-oriented approach and because of the inherently 'agent' oriented nature of the scenario, this would appear to be the most suitable decomposition paradigm to use. An object-oriented approach also permits a looser control structure over the system than might be possible utilising a function-oriented approach.

In *Common*KADS (as in many other software projects) pragmatic considerations such as cost and the availability of programming personnel, rather than more logical reasons, usually govern the chosen platform and programming language for an implementation. Hence to attempt to specify hardware platforms and implementation languages for the scenario would be inappropriate.

12.5.2 Scenario 2

The architectural structure of the I3D system is detailed in Figure 12.5.

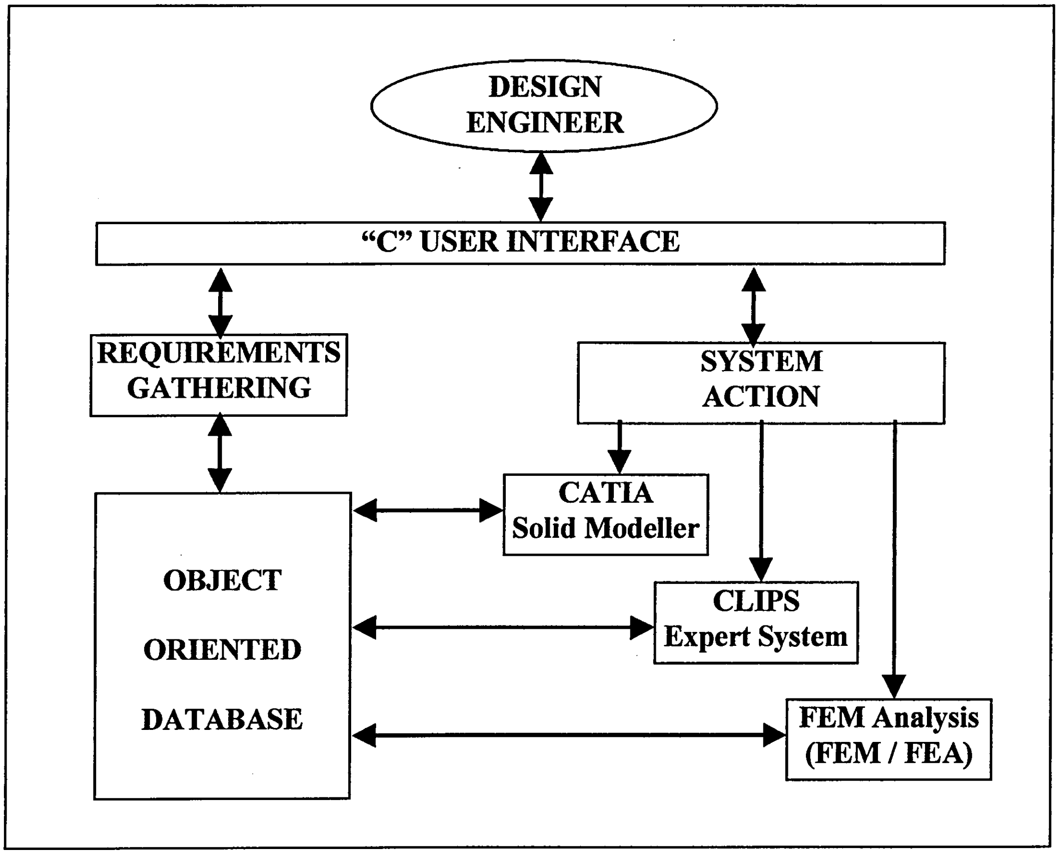


Figure 12.5: The implemented architecture of the I3D system.

Reproduced from Victor et al [1993] this shows the different modules comprising the full I3D system and their interactions.

The implemented I3D system incorporates a modular approach to system building and integration.

12.5.2.1 Architectural design

A major goal of the project was that it *'requires the use of commercial, open-architecture software systems ... this enables easy transfer of the systems developed.'*

The user interface to the system provides a number of graphical displays, which the user can use to view different aspects of the design. This features four different windows on the workstation screen. The upper left window shows the user interaction with the system. The upper right window is used by CATIA to display design information. The lower left window is used by the system to give instructions to the designer while the lower right window shows a design history.

12.5.2.2 Platform design

The Design engineer utilises a CAD system to interface with different modules via an interface layer, which is coded using the 'C' programming language. The Solid modeller used is CATIA, a high-level solid modelling program from IBM. The expert system building tool is CLIPS (C Language Integrated Production System). This is a rule-based, forward chaining system which provides an easy interface to the programmes written in 'C' as well as featuring the CLIPS Object Oriented Language (COOL). In I3D, the state of the design is stored as CLIPS facts. The entire system sits on an IBM RS/6000 workstation, running UNIX.

12.5.2.3 Application design

The essential functions performed by the expert systems are the critiquing tasks for different life-cycle perspectives. Hence these can be de-composed to sub-systems sub_Critique_Manufacture etc. It is at this level that the actual implementation of the sub-system will feature input from the expertise task models and where computational means will be used to achieve 'humanesque' inferencing capabilities. As an example, I will now consider the cost critiquing sub-system of the I3D system, where design critiques from a cost perspective are generated.

I have modelled the critiquing task as being de-composed into sub-tasks interpret, gauge, outline, assess, propose and pinpoint (see Figure 12.3). I will now consider the 'propose' sub-task where an alternative to a design proposal, from a cost life-cycle perspective, is generated. As shown in figure 12.6, the primitive task 'Propose' at the task layer is linked to the inference 'Propose' in the inference layer. This implies propose is a task which cannot be further decomposed into sub-tasks and can thus be computationally implemented as a 'black box' code module where the inner workings of the propose inference do not necessarily match the cognitive means by which experts perform the task. Victor et al [1993] do not detail the computational methods that are used to implement such tasks but suggest they are based on *'text book, experimental and heuristic knowledge ... analytical and heuristic models were developed.'*

The domain 'roles' acting as input and output to the inference are linked to domain knowledge at the domain layer.

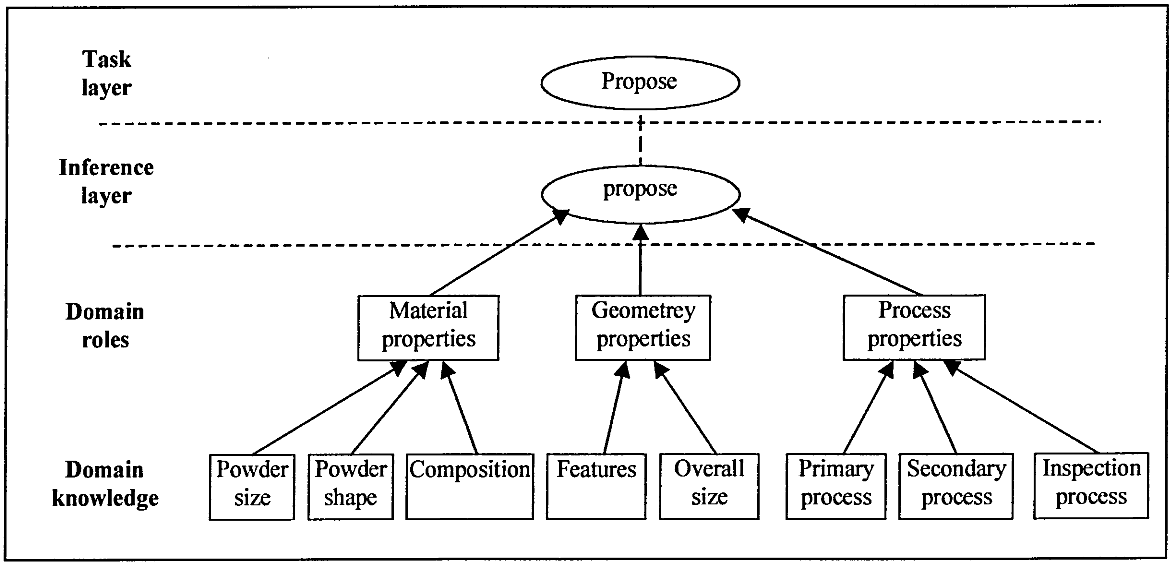


Figure 12.6: A propose inference from a cost life-cycle perspective.

This shows how knowledge at different 'layers' is linked and utilised within an implementation. The element of domain knowledge 'Powder size' is abstracted to the domain role 'Material properties'. This is used by the inference 'propose' which is linked to the super-task 'Propose'.

As another example, I have assumed a basic inference 'interpret' whereby different life-cycle perspectives can interpret a given design from their own point of view. The cognitive process by which this occurs is clearly a very complex process and the computational means to achieve this will be significant in terms of the time and effort required. Sanderson et al [1990] outline computational methods by which designs may be interpreted from an assembly perspective. These issues are central to the knowledge modelling methodology, where at some level of granularity, computational methods are used to produce similar results to those derived from the application of human cognitive processes. This issue of deriving views on a design is also important in the two scenarios, where computational means must be implemented whereby machine-based critiquing agents are able to interpret a given design (in the form of a CAD drawing) in a form suitable for 'intelligent critiquing' by these agents.

12.6 Discussion

I have detailed the instantiation of two possible scenarios using my generic model templates as a basis. I will now discuss my experiences in instantiating the scenarios and the extent to which this validated and extended my generic model templates.

12.6.1 Using the generic model templates

The ability to import models and model entities from other projects and libraries is central to the *Common*KADS ideal of re-use and proved extremely useful when instantiating my two scenarios from the generic model templates I have developed.

During instantiation of the different scenarios, it became clear that some of the models are more generic than others. The design models are very specific to a particular scenario. As a result, the generic design model contains relatively little detail. In addition, the agent model is relatively implementation-dependent in that the agents comprising a proposed system will be dictated by the requirements of the system.

However, the task and communication model templates offer more generic features. The only details needing instantiation for different scenarios in the task model were different environmental constraints and the goals of the task (the critiquing task). The communication model ingredients were similarly generic although transaction plans and initiatives needed defining for each scenario.

The expertise model is generic down to the level of task (although some implementations may make greater use of the available expertise and transfer tasks than others). However, at the inference and domain levels, there will clearly be instantiation dependent features. For example, the basic inferences required from the types of knowledge-based systems envisaged, will depart from the cognitive processes that are employed by human experts, as I have outlined in the previous section.

One of the main considerations is as to how co-operative or concurrent the intended scenario is believed to be. In an 'idealised' cooperative design scenario, all agents would be working together towards the common goal of developing the design. In a more concurrent scenario, the goals of different agents will clearly differ. Hence there needs to be some means of adjusting the autonomy of different agents within the system depending on the degree of cooperation envisaged. I have envisaged this being done by defining the goals of the tasks performed by different agents and also by defining the importance and autonomy properties, which I have introduced, for the different agents.

A useful conclusion to make is that I estimate my generic model templates were very significant in speeding model generation by a matter of weeks while developing the two different implementational scenarios. This includes all the background work to develop initial models for concurrent design and the subsequent knowledge elicitation process with my case studies. Without access to the generic model templates for the purposes of instantiating the scenarios, the instantiation would have taken considerably longer as the existing *Common*KADS templates are more distant from models for design. My generated templates acted as partially instantiated models. In addition, instantiating the two scenario also acted to extend the content of my generic model templates. As an example, scenario 2 required the instantiation of two additional critiquing agents, the reliability and inspection experts.

12.6.2 Using the *Common*KADS workbench

Using the *Common*KADS workbench provided a number of advantages during the development of the generic model templates and the instantiation of the two scenarios. In particular, because the workbench is based on the *Common*KADS methodology, the knowledge engineer is forced to comply with the formalisms of the methodology. This helps to ensure the formal consistency and correctness of the developed models.

A number of limitations also became apparent during use of the workbench. The workbench is generally slow and the reliability is not at the level of what would generally be considered acceptable for production software. In addition, there are a number of bugs in the workbench itself. As such, the quality of the workbench can be seen as being at a Beta level of testing. In particular, the organisation model module was difficult and frustrating to use. As a result, while the quality of the software is acceptable for a research project such as ours, it is unlikely that developers of real-life KBS would be prepared to accept these limitations.

Expertise model entities can be imported from the *Common*KADS library, but cannot be imported from other projects. While this made the instantiation of the expertise models for my scenarios more time-consuming, I believe this is a sensible approach for expertise models as it allows any proposed expertise models to be fully validated by the *Common*KADS Model Coordination Committee before they can be used as generic templates.

12.6.3 Limitations within *Common*KADS

I have outlined a number of pragmatic issues that make the workbench difficult to use, however it is the shortcomings within the underlying *Common*KADS methodology that are more important. A major problem with *Common*KADS is that it does not allow for the alternative specification of task decompositions. As I have identified, a propose task (for example) may have a number of possible decompositions, based on different applied PSM's, even though the task definition is the same. Unfortunately, the workbench does not allow the representation of different possible decompositions for the same task within the same expertise task model. The only way to represent different task decompositions is to specify a different expertise task model (propose1, propose 2 etc). This is felt to be a deficiency in the tool in that during the concurrent design task, a designer or critiquer may use different problem-solving methods at different stages of the concurrent design task to generate proposals or alternative proposals. The knowledge they use to select the different PSM's is their strategic knowledge of that particular domain. In essence, they need to be able to specify different task decompositions 'on the fly' and the workbench needs to be able to represent this in some way instead of rigidly specifying particular task decompositions for a task.

This limitation becomes particularly apparent when a 'super–task', such as concurrent design, is considered. With a number of possible different sub-tasks (propose1, 2, 3, critique 1,2 etc), the number of possible decompositions for concurrent design 1,2,3 etc becomes unmanageable. The negotiation task can also feature a complex flow of control in that different agents can make concessionary proposals, using an appeasement strategy or some other negotiation strategy.

It is very difficult to represent such a scenario in a satisfactory manner, using the workbench because of these limitations with the *Common*KADS methodology. The selection of different PSM's for tasks and the implied flow of control are effectively defined as the strategic reasoning of an application. *Common*KADS support for this strategic aspect of knowledge is currently not as strong as for the application aspect, a view endorsed by Schreiber et al [1999] - *'Dynamic model configuration can also be seen as strategic reasoning. This is certainly an area for further development'.*

However, work is ongoing to improve this situation, see Breuker and Boer [1998]. If KBS support for these strategic aspects of problem-solving are to be implemented, both the methodology and tools to support the methodology need to be able to incorporate these strategic reasoning aspects of knowledge. It should be possible to specify a task definition and then select from a number of different possible PSM'S to instantiate the task body. The knowledge used to select the most suitable PSM is the strategic reasoning knowledge. It could include factors such as the availability of different types of knowledge required for a particular PSM or the cognitive overhead implied by obtaining some piece of knowledge.

I believe these issues are particularly important if *Common*KADS is to be used to support concurrent design, where computational implementations can provide useful support for these discrete sub-tasks within the overall concurrent design process.

In addition, I wished to model the different agents within the system as having varying levels of autonomy. I.e. the goals and motivations of agents would be different. However, this could not be done in the agent model and had to be expressed by setting the goal of a task in the task definition. Again, this meant generating different task models for the same task in order to accommodate different levels of autonomy. I believe that extending the agent model to provide a means of expressing the autonomy and goals of the agent would be useful.

A further criticism of *Common*KADS I believe is the degree of overlap shown by the different models comprising the complete model set. I believe that the expertise and organisational models are very useful in providing informative and complementary views on issues to be considered when implementing knowledge-based systems in organisations. However, the agent, task and communication models exhibit significant degrees of overlap and use of these models can become cumbersome and time-consuming when instantiating different scenarios. While *Common*KADS does acknowledge that some models may be instantiated to a greater degree than others, I believe that the models could be stream-lined to reduce this degree of overlap. In addition I believe that the design model, where the actual implementation decisions inherent in the other models are specified might currently be best modelled in a more mainstream software development methodology such as UML (see Jacobson et al [1998], Booch et al [1999]).

12.7 Chapter Summary

Two possible implementation scenarios have been outlined and my generic template models instantiated to support the particular scenarios. This has helped to validate my generic model templates and illustrates how the templates could be used by developers of knowledge-based systems for the development of tools to support concurrent design. It is concluded that my generic model templates acted as extremely useful templates during the instantiation of the two scenarios and greatly speeded model development.

Use of the *Common*KADS workbench was advantageous in that it enforces compliance with the formalisms of the methodology. However, a number of problems inherent in the use of the tool and its level of support for *Common*KADS are also commented on. The implications of some of these issues have been discussed in this chapter. Of particular importance is the way in which *Common*KADS and the workbench fail to provide support for the dynamic allocation of problem-solving methods to tasks. For a flexible and dynamic process such as concurrent design, these issues are crucial. A rigidly defined task structure is not felt to be appropriate to support concurrent design.

13 Conclusions and future work

13.1 Introduction

In this chapter I conclude the thesis by summarising work done to develop a knowledge-level model to support the concurrent design process and the methodology used to develop the models. The term 'a knowledge-level model' is used to describe the overall model for the concurrent design process. This model consists of further sub-models or components. Hence the term 'model' and 'models' are used interchangeably in the text when describing the developed model(s).

I have defined concurrent design as the design process where the influence of downstream life-cycle constraints on the design process is of considerable importance. I then go on to re-visit the aims and objectives of the research, previously outlined in Chapter 1 and discuss the main findings from the research and identify original contributions to knowledge. Future work that could expand on this research and contemporary research that begins to address these issues is then discussed.

The motivations behind this research originally grew from a belief that existing prescriptive models for design do not address at a sufficient level of detail how the concurrent design process occurs. Such prescriptive models are typified by those described in texts such as Pahl and Beitz [1984], Pugh [1990] and Hubka [1982]. These prescriptive models give a good, general overall process model for how both design (and concurrent design) could proceed. However, they do not give at a lower level of detail, the tasks and methods that a designer or design team practicing concurrent design might utilise. If computer support for the concurrent design process is to be effectively implemented, it is my belief that more detailed knowledge-level models for concurrent design are needed.

*Common*KADS is a knowledge-based system development methodology, intended to provide support for developers of knoledge-based systems. The *Common*KADS library of task models already contains models at various levels of detail for the design process. However, these are generally formalised versions of the so-called prescriptive models.

They do not explicitly support the process of concurrent design, where the consideration of downstream constraints and the implied critiquing and negotiation that ensues, are crucial. At the beginning of the research, I believed that there were no knowledge-level models that effectively described and supported the concurrent design task, although a number of researchers, in particular Smithers [1996] had illustrated the need for such models.

Analysing and modelling design activity has been outlined in earlier chapters as a complex and difficult process. A number of researchers and methodologies, including *Common*KADS, outline different methods and techniques for analysing and eliciting expert behaviour and specifically design behaviour. On analysing the literature, it became clear that there were no definitive guidelines on how best to analyse design pracrtitioners in action, with a view to eliciting and modelling their expertise. Concurrent design, because of its specific characteristics, where multi-functional sources of knowledge contribute to the overall process, provides a number of additional and unique challenges to the knowledge engineer attempting to model the process. Therefore, I have developed and employed a novel method for eliciting and modelling concurrent design behaviour.

Hence the original aims and objectives of the research were:

- To develop a knowledge-level model of the concurrent design process to support and inform developers of tools for the concurrent design process. Such tools could range from a simple critiquing tool to critique evolving designs from the perspective of a single life-cycle to the development of a more complete concurrent design support system.

- To develop and utilise a novel method for analysing and eliciting the expertise of participants in the concurrent design process.

13.2 Achieving the aims and objectives

Before outlining the contribution to knowledge that I believe the findings of this research make, I will summarise how the original aims and objectives of the research were achieved.

13.2.1 The development of a knowledge-level model for concurrent design

A requirement of the developed methodology is that an initial top-down model of concurrent design behaviour is developed from analysis of the available literature and modifying and refining any existing models.

13.2.1.1 The development and refinement of knowledge-level models.

The work of Schon [1991] attempts to describe at a lower level of detail the way in which designers work via a process described as 'reflection-in-action'. Via this technique, the designer takes some action that advances the design and then reflects on this action in the context of the design as a whole. The work of Chandrasekaran [1990] also outlines at a lower level of detail the tasks and methods that designers use, although in a markedly different manner to that of Schon. The work of these two researchers was particularly influential in the development of initial top-down models for concurrent design.

Based on the results obtained from different case-studies, transcripts describing expert design behaviour were used to inform the development of new bottom-up knowledge-level models for critiquing and negotiation in the context of concurrent design. An iterative process was then utilised with the case-studies to further refine and develop the knowledge-level models.

In order to verify and validate my developed models and provide a useful library for developers of KBS to support the concurrent design process, I have used a software tool, the *Common*KADS workbench, to instantiate my knowledge-level models as generic templates. My research has taken a predominantly task-based view of the design process and views different problem-solving methods as being applied to solve the different sub-tasks implied by the concurrent design process. The research has also focussed on what I would consider to be the 'core design synthesis tasks'. I have not explicitly addressed other issues associated with design, such as project planning, reporting etc even though these activities are also clearly critical to the successful outcome of any real-life design scenario. The development of the initial top-down model to support concurrent design is detailed in Chapter 7. The generic models resulting from the refinement of these initial models using my analysis method are outlined in Chapter 10.

13.2.1.2 Concurrent design as propose, critique and negotiate

My research has resulted in models for concurrent design which suggests that designers work in a mode based on propose, propose–critique and propose–critique–negotiate sub-tasks. Each propose step, which seeks to advance the design in some way, is then subject to critiquing and possible negotiation to determine if the design proposal is acceptable from different life-cycle perspectives. These sub-tasks can be used at various levels of granularity within the concurrent design task. For instance, the 'propose' task could result in the generation of a complete conceptual design for some artefact followed by subsequent critiquing and negotiation. It could also be used for the detailed designing of some small sub-component comprising the complete artefact. This process is outlined in Figures 10.7, 10.9 and 10.10, which illustrate the different tasks of propose, critique and negotiate. Figure 10.12 then gives an overall model for the concurrent design task with important items of knowledge that play key roles in these processes.

However, while a large amount of research has studied these design 'proposal' methods, relatively little research has been done on the critiquing and negotiation tasks and corresponding methods. I believe my work has helped to expand on these areas.

13.2.1.3 My knowledge-level models and the *Common*KADS suite of problem types

I have outlined the development of a series of generic knowledge-level models to support the process of concurrent design. The models have been developed along guidelines specified by the *Common*KADS methodology and were based on a number of case studies, which have analysed designers and design teams operating in a concurrent manner.

These models are believed to be an important contribution to the *Common*KADS suite of problem-solving types. Based on the dependencies between different problem types involved in the complete design task (presented in Chapter 2) my developed models for different sub-tasks relevant to the concurrent design task effectively 'slot' into the space between design and assignment as shown in Figure 13.1.

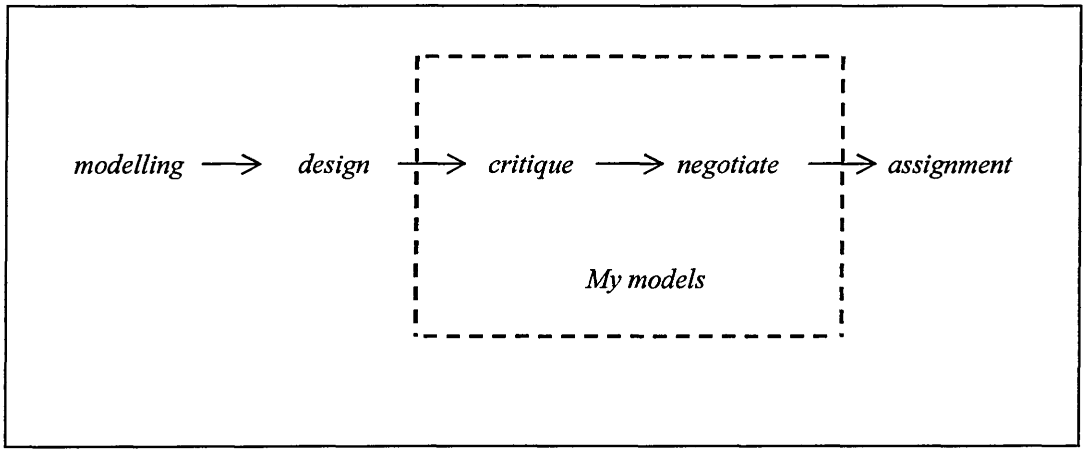


Figure 13.1: The *Common*KADS design task problem types and my models (adapted from Breuker [1994]).

Currently *Common*KADS views the design problem type, which is viewed as a synthetic task, as being followed by an assignment task. I believe there is an iterative process of critique and negotiate between these two tasks.

I believe my developed knowledge-level models to support concurrent design are complementary to the models for design previously developed as part of the *Common*KADS methodology. I view the design process as a dynamic, iterative process of propose-critique-negotiate. A number of the existing *Common*KADS design task models deal with the synthesis stage of design (including case-based, hierarchical decomposition, transformation based, generic model, parametric and configurational design). I would consider these models as being the result of the application of different problem-solving methods to what I term the design proposal task.

A number of other *Common*KADS models for design deal with the development of a suitable design specification. Only the *Common*KADS model for original design and Kingston's model for exploratory design (see Appendix A) begin to consider how some form of critiquing and negotiation may play important roles in the design process. I believe my developed models for concurrent design expand and refine on the way in which the existing *Common*KADS models provide support for the critiquing and negotiation processes, which are vital to the concurrent design process.

I see the concurrent design process as being distinct from other more co-operative design processes (termed variously collaborative, co-operative design etc). This is because of the inherently conflicting nature of the life-cycle perspectives, due to the differing goals and assumptions, that are brought to bear on the process. In particular, the negotiation aspects of the concurrent design process are significantly different from more co-operative modes of design. Hence my models developed for concurrent design are believed to be subtly but significantly different from other models developed for more co-operative modes of group design.

I also have reservations about using task structures to represent procedural design knowledge. Because of the spontaneous and dynamic nature of the concurrent design process, a rigid task-type hierarchy can be too restrictive a format to represent such knowledge. My feeling is that support for the design process can currently most effectively be implemented by providing support for the different problem-solving methods designers' use, with the sequencing and control of the problem-solving methods being left to the designer and design team. I believe that the *Common*KADS methodology and its associated software support tools need refining in order to support such dynamic allocation of methods. These issues have been further discussed and expanded on in Chapter 12.

13.2.1.4 Scope of application of the developed models

The concept of knowledge-level modelling has been introduced as a more pragmatic means (than cognitive modelling) for implementing computer-based support for complex problem-solving tasks such as design. Knowledge modelling does not require a full cognitive understanding of a particular task in order to support the task. The *Common*KADS development methodology supports the development of knowledge-level models of problem-solving expertise for subsequent implementation in computer-based systems.

It could be argued that systems such as the ACDS system (described in Appendix B, also see Darr and Birmingham [1994]) are knowledge-level models but at a relatively coarse level of granularity. To a certain extent this is true. At some level of granularity, a knowledge-level model will always depart from its cognitive counterpart because of the current lack of understanding of cognitive processes. Hence, at the given level of granularity, the knowledge-level model will become a 'black box', where the defined inputs and outputs of knowledge match those used in cognitive processes but the means used to derive the outputs from the inputs will differ. However, I believe that the finer the granularity that this eventually happens, the more the knowledge-level model approaches the corresponding cognitive processes. Hence the more likely human experts are to be able to work with systems based on such knowledge-level models and have confidence in their results. It is at this sort of level that I believe my developed models can be of use to system developers.

In addition, such tools should be as flexible as possible in supporting designers so that the designer has control over the design process in terms of what tasks and sub–tasks to pursue and the appropriate choice of PSM's for solution of tasks and sub–tasks. It is felt that these strategic aspects of design are currently not well enough understood to permit their implementation in computer-based design support tools.

13.2.2 A novel knowledge analysis and elicitation method for concurrent design

During the course of my research, I have developed and utilised what I believe is a new method of eliciting and modelling knowledge of designers using the concurrent design process. The thinking behind the justification, development and testing of this novel methodology have been described and discussed in Chapters 5 and 6.

Actually deriving models from observed data is an area fraught with difficulty. The way in which researchers analyse design activity and derive models and hypotheses from observed data is one of the critical areas related to the development of accurate models for design and is currently a very active and fruitful area of research.

The development of the methodology was influenced by analysing a number of different methods and techniques that have been used to analyse expert behaviour in a variety of domains, including design. The applicability of these techniques to the analysis of the domain of concurrent design was then investigated. It became clear that elements from the different techniques could be usefully combined to provide a novel method for analysing concurrent design activity.

A key intention of the research was to analyse concurrent design activity in a realistic, as opposed to a more laboratory type setting. Chapter 6 goes on to describe the different industrial case studies and how the novel methodology was used with these case studies.

The technique of protocol analysis has been widely used to analyse and model the design process. The usual way that protocols are analysed is to split the discourse into discrete 'chunks' and then use some pre-defined coding schema to classify the different chunks. In this way, the designers are assumed to be 'doing something' relating to a particular classification at different stages within the protocol. The major disadvantage of this approach is that the pre-defined coding schema has a pre-determining effect on the analysis of the data and may miss significant features of the subject's behaviour.

The approach I have taken is different in that both retrospective and real–time studies of designers were used to generate audio and video transcripts. These were based on discussions and analysis of designers, where an initially developed model for concurrent design was used to stimulate and drive the initial discussions. These transcripts were then analysed to determine the key tasks and knowledge roles that the designers used when performing the task of concurrent design. The resulting expertise task models were used to drive further knowledge elicitation sessions with the designers in order to verify that the models concur with the way designers believe they design.

The retrospective nature of some of the sessions also allowed some critical issues regarding concurrent design to come to light which would not have been unearthed with more conventional ethnographic type studies. These involved issues such as different perspectives lying about their true goals and aims. During the course of the research, a number of problems also became apparent in attempting to analyse designers and design teams in real–life industrial settings (issues such as secrecy and client confidentiality made organisations very reluctant to engage in such studies).

13.3 Contributions of this research

Based on this discussion, I believe the main contributions to knowledge of my research are two-fold:

- The development of knowledge-level models to support the process of concurrent design, which expand on the models that have been developed as part of the *Common*KADS methodology. These have been instantiated as generic model templates, using the formalisms of *Common*KADS on a *Common*KADS workbench.

- The use of a novel method of knowledge acquisition to elicit and model concurrent design activity in order to facilitate the development of the knowledge-level model.

This contribution is reflected in a number of conference and journal publications (see Appendix E) detailing the research at different stages. In particular, Barker et al [1998] outlines the thinking behind the development of an initial model for concurrent design. The submission of this paper resulted in an invitation to submit a more comprehensive and detailed Journal paper describing the knowledge analysis method and the knowledge-level model at the time, see Barker et al [1999].

I will now go on to discuss contemporary research that continues to expand on these areas.

13.4 Contemporary research and recommendations for further work

As can be seen from my conclusions regarding different aspects of the development of knowledge-level models for concurrent design and associated contemporary work, the development of models for design and concurrent design and their subsequent implementation in computer-based systems is a very active area of research. I would consider the following areas to be important areas of research in order to further the effectiveness of computer–based support for the concurrent design process:

- More work needs to be done in analysing in detail the problem-solving methods designers use and how they match problem-solving methods to tasks and sub-tasks. In addition, the *Common*KADS methodology and its' associated software tools need to be able to represent the dynamic allocation of problem-solving methods to tasks in a more flexible manner.

- The generic nature of the design process across different domains of application. This research has contrasted findings in the field of mechanically oriented design with findings from other researchers in the field of software design. However, design is a process applied in a wide number of other diverse domains. This also extends to issues about whether design knowledge is generalisable across different design situations or whether design knowledge is particular to a design situation.

- Analysing design activity is currently a very active area of research. There is no current consensus as to the most effective means of analysing design activity.

- I believe that the negotiation task within concurrent design would benefit from further research.

- It is very important to note that I have developed my generic model templates as being a generic super-set of all the different features apparent in the different case studies. As a result, I would not expect my generic models to comprise features that may be apparent in other instantiations of concurrent design. As a result, I feel further work is possible to expand on my generic model templates.

I will now expand on some of these issues and describe contemporary work that is beginning to address some of these recommendations for future work.

13.4.1 Problem-solving methods

Contemporary work on the design process has outlined further work on the 'propose' task. Wielinga and Schreiber [1997] have studied the ‘configurational design’ problem (where an assembly of components is generated from a set of predefined components) and see this as a distinct type of designing. They see layout, parametric and skeletal design as specialisations of this form of design. They go on to outline the different knowledge roles and PSM’s utilised for the configurational design task. The types of PSM they see as being utilised within configurational design correspond with those outlined in my research and also by other researchers.

Interestingly, Wielinga and Schreiber [1997] suggest that "*getting it right the first time, using a significant amount of domain-specific knowledge is the most efficient route towards efficient and competent configuration design systems*". While I believe this approach is a computationally effective strategy, this conflicts with my view of concurrent design. One of the key tenets of concurrent design is that any one perspective does not possess full domain knowledge and the cycles of propose–critique–negotiate are necessary steps in thinking about the design and its’ proposed application.

Brown and Birmingham [1997], Maher and Gomez da Silva [1997] and Umeda and Tomiyama [1997] have also done work on what my research terms the ‘propose’ methods utilised in concurrent design.

While I have been developing my ideas on knowledge-level support for concurrent design, Matta (Matta et al [1998], Matta [2000]) has also been doing independent and complementary work in this field. However, while I have concentrated on the critiquing aspects of concurrent design (in particular at the influence of different life-cycle constraints on the design process) and the negotiation strategies that can occur as a result, Matta has focussed on conflict management and avoidance strategies within the concurrent design process. Matta's work has resulted in *Common*KADS models of expertise formalised using the CML formalism.

The complete verification and validation of PSM's is complex and involves research and issues outside the scope of my research. Breuker and Boer [1998] have recently outlined a framework that attempts to do this. They have taken some of the developed *Common*KADS PSM's and postulate on how to validate the correctness of these PSM's using a knowledge-base of domain knowledge. They are attempting to do this using a software modelling system called 'Cokace'.

Their initial thoughts are that the same domain knowledge can be used to validate a number of different PSM's. Via the use of 'roles', *Common*KADS can use domain knowledge in different ways depending on how it is to be applied in a PSM. However, they then question the very basis of this assumption. The interaction hypothesis (Chandrasekaran [1987]) suggests that domain knowledge cannot necessarily be represented in a manner that is independent of the PSM with which is to be used. This raises questions as to the reusability of knowledge-bases of 'domain' knowledge. Breuker and Boer [1998] also go on to question whether PSM's are in fact generic across different domains and conclude that this is still an area of debate.

The findings from my research would tend to suggest that the task that knowledge is to be used for, and also the representational format used, will have a great influence on the knowledge elicitation process. It would be very difficult to attempt to elicit knowledge from an expert without some idea of the use the knowledge is to be put.

13.4.2 Generality of design expertise

The reflective practitioner approach (Schon [1991]) would tend to suggest that each design event is a unique situation and attempts to ascribe any form of generality for the design process will by definition weaken the applicability of the process to specific design situations. Liddament [1999] also feels that attempts to represent the design process computationally are reductionist in that they attempt to generalise the design process. As Liddamment states, it is unlikely that working designers utilise this approach when working.

My belief is that problem-solving knowledge derived from some unique design situation has its' clearest and most effective application within that unique situation. However my analysis of different case studies and also the work of other researchers suggest that design as a task does have a number of generic features across a number of different design situations and indeed across a number of diverse domains of application.

My models for concurrent design have been mainly developed with input from the mechanically-oriented field of design. However, concurrent design as a process is utilised in a wide-ranging number of fields or domains. I have also developed models for critiquing and negotiation as they occur within the concurrent design process. However, these tasks are also found in other processes and domains. I believe that the models I have developed for critiquing and negotiation within the specified domain are likely to have applications in other domains.

Gero and McNeill [1998] outline a coding technique based on protocol analysis for testing different hypotheses about how different designers' design. The goal of the research is to obtain a better understanding of how human designers actually design. They acknowledge that the research is currently at an early stage but one of the hypotheses that they wish to test is whether there are fundamental differences between designers from different domains.

13.4.3 Analysing design activity

Valkenberg and Dorst [1998] have attempted to extend the theories of Schon to develop a 'reflection-in-action' based theory for group design. They outline some of the problems inherent in analysing design activity in a real–life setting. They make use of multi-functional student design teams which neatly side-steps some of these issues. However, a key issue regarding this approach is whether student designers utilise similar or different approaches to the design process compared to more experienced designers. A study by Lloyd and Scott [1994] suggests that they are indeed different. They also make use of a novel protocol analysis method, which splits transcripts up into what they term episodes, rather than using the more common method of using time-slices to analyse protocols.

Gavrilova and Voinov [1998] outline describe ongoing work on the visual specification of knowledge-bases. The evolving software system they describe enables a user to develop visual representations of an expert's knowledge using a variety of original knowledge sources (interview transcripts, lecture notes, audio speech records etc). However, the system they describe would appear to be aimed more at developing, in the *Common*KADS viewpoint, a domain layer of a knowledge-based system. However a number of the issues they discuss are pertinent to the novel knowledge analysis method utilised in my research, although they make no reference as to the developed visual models being presented back to experts for review.

13.5 Final thoughts

Design and particularly concurrent design as processes are still far from understood. I have developed a number of knowledge-level models of the concurrent design process. These models represent the different problem-solving methods that designers are believed to utilise in the different tasks and sub-tasks comprising concurrent design. These models are intended to support potential developers of KBS in the domain of concurrent design.

In order to develop such models, I have utilised a novel method for eliciting and modelling concurrent design behaviour. Analysing and modelling design activity is believed to be an active area of research where greater understanding of the design and concurrent design processes would result from more effective methods of knowledge elicitation and analysis.

A further issue that is not well understood is how designers match different problem methods to a particular task and how different sub–tasks are sequenced within the overall task of concurrent design. Hence, while support for the different sub–tasks associated with concurrent design can be provided by the models, it is felt that some of the more strategic and creative decisions made by human designers are still not sufficiently well understood. Because of this it is believed that computer support for design is currently limited to this supporting role within the synthesis stage until the way in which designers work is more clearly understood.

This allows some of the criticisms aimed at so called ‘prescriptive’ models for design to be overcome, by the use of a more flexible framework whereby the designer is supported in the execution of some task but is left free to determine the sequencing and control over these tasks.

My work has helped to identify areas in which the *Common*KADS methodology and its associated workbench are deficient in their support for strategic aspects of knowledge modelling. For example, where more than one problem-solving method may be applied to the task, a system or user would be required to dynamically allocate a problem-solving method to the solution of that particular task.

References

Aamodt, A; Plaza, E. 1994. 'Case-based Reasoning: Foundational Issues, Methodological Variations And System Approaches'. AICOM Vol. 7, No. 1, March 1994.

Aben, M. 1994. 'Canonical functions: *Common*KADS inferences'. Chapter 5, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds.: Breuker, J; Van de Velde, W. Publishers: IOS Press, 1994.

Akin, O 1995. Editorial. Design studies 16. 1995.

Akkermans, H; Wielinga, B; Schreiber, G. 1994. 'Approaches to expertise model construction'. Chapter 5, 'Expertise model definition document'. Ed: Wielinga, B. ESPRIT Project P5248 report, Pub. University of Amsterdam.

Akkermans, H; Speel, P-H; Ratcliffe, A. 1999. 'Problem, Opportunity and Feasibility Analysis for Knowledge Management: An Industrial Case Study'.
KAW 99, Banff, Ca. http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html

Andreasan, M. M; Kohler, S; Lund, T. 1983. 'Design for assembly'. IFS Publications, London.

Anjewierden, A; Wielemaker, J; Toussaint, C. 1990. 'Shelley - computer aided knowledge engineering'. In 'Current trends in knowledge acquisition', IOS Press, Eds.: Wielinga, Boose, Gaines, Scrieber, Van Someren.

Bahler, Dupont, C; Bowen, J. 1994. 'Mediating conflict in concurrent engineering with a protocol based on utility'. Concurrent Engineering, Research and Applications, Vol. 2, Issue 3, pages 197-207.

Bainbridge, 1979. 'Verbal reports as evidence of the process operator's knowledge'. International journal of man machine studies, Vol. 11, 1979.

Baker, M;J. 1993. 'Dialogic Learning: Negotiation and Argumentation as Mediating Mechanisms', invited talk, The Proceedings of World Conference on Artificial Intelligence in Education, Edinburgh, Scotland, 4-11.

Bakyan, C. 1996. 'Design Strategies'. Chapter 6 In 'Analysing Design Activity', Eds.: Cross, N; Christiaans, H; Dorst, K. John Wiley and Sons, 1996.

Balachandarn, M; Gero, J. S. 1988 'A model for knowledge-based graphical interfaces'. Artificial intelligence Developments and Applications, Ed: J. S. Gero, R. Stanton, Pub. Elsevier Science Publishers.

Ball, N.R; Murdoch, T.N.S; Wallace, K.M. 'A framework for design object evolution'. Artificial intelligence in design, Eds. J.S. Gero, F. Sudweeks 1996. Pub. Kluwer academic publishers.

Barker, R; Short, C; Tranter, I; Parkinson, B. 1995. 'An expert teaching system for concurrent engineering'. Second Int. Conf. on Design to manufacture in modern industry (dmi '95), 29-30 May, 1995, Bled, Slovenia.

Barker, R; Short, C; Tranter, I; Meehan, A; Parkinson, B. 1996(a). 'Critiquing and the concurrent design process'. 3rd International conference on concurrent engineering and electronic design automation - CEEDA 96, Poole, UK. Edited by Saad Meehat, Pub. EPIC EM.

Barker, R; Holloway, L; Tranter, I; Short, C; Parkinson B. 1996(b). 'A learning environment for concurrent engineering'. The First Annual Conference on Applied Concurrent Engineering (ACE '96) Seattle, Washington, USA, November 5-7 1996.

Barker, R; Meehan; A; Tranter, I. 1998. 'Towards a Knowledge-level Model for Concurrent Design'. 11th International conference on industrial and engineering applications of artificial intelligence and expert systems. Publishers: Springer Verlag 1998.

Barker, R; Meehan; A. 1999. 'Supporting Concurrent Design Teams with Adjustably Autonomous Agents.' AAAI Spring Symposium 1999, Agents with Adjustable Autonomy, Stanford University, Ca., USA; AAAI Technical Report SS-99-06, AAAI Press, Menlo Park, Ca.; ISBN 1-57735-102-9.

Barker, R; Meehan, A; Tranter, I. 1999. 'A knowledge-level model for concurrent design'. Journal: International Journal of Applied Intelligence 10, pages 113-122. Kluwer Academic Publishers

Barker, R; Holloway, L.P; Mardell, J; Meehan, A. 2000. 'Supporting Knowledge-based Processes Using Flexible Intelligent Agents.' AAAI Spring Symposium 2000, Bringing Knowledge to Business Processes, Stanford University, Ca., USA; AAAI Technical Report SS-00-03, AAAI Press, Menlo Park, Ca., USA; ISBN 1-57735-109-6

Bañares-Alcántara, R. 1995. 'Design Support Systems for Process Engineering - 1: Requirements and Proposed Solutions for a Design Process Representation'. Computers & Chemical Engineering 19, pages 267-277, 1995.

Bayliss, D; Akueson, R; Knight, J. 1994. 'Implementing concurrent engineering using intelligent systems'. Proceedings of the international conference on concurrent engineering and electronic design automation (CEEDA), Bournemouth, UK, 7-8 April, 1994. Edited by Sa'ad Medhat. Society for computer simulation international.

Benjamins, R; De Barros, L. N; Valente, A. 1996. 'Constructing planners through problem-solving methods'. In Gaines, B; & Musen, M. (Eds.). Proceedings of the 10th Knowledge acquisition for knowledge-based systems workshop.

Bernaras, A; Van de Velde, W. 1994. 'Design'. Chapter 8, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds.: Breuker, J; Van de Velde, W. Publishers IOS Press, 1994.

Blessing, L.T.M. 1994. 'A process based approach to computer-supported engineering design'. Published PhD-thesis, University of Twente.

Boehm, B. 1988. 'A spiral model of software development and enhancement'. IEEE Computer, pages 61-72, May 1988.

Bond, A. H; Gasser, L. 1988 'Readings in Distributed Artificial Intelligence'. San Mateo, California, Morgan Kaufman Publishers 1988.

Booch, G; Jacobson, I; Rumbaugh, J. 1999. '.'The Unified Modelling Language User Guide Addison Wesley 1999

Breuker, J. A; Wielinga, B. J. 1985. 'KADS: Structured knowledge acquisition for expert systems'. Proc. of the fifth international workshop on expert systems and their applications, Avignon 1985.

Breuker, J; Boer, A. 1998. 'So you want to validate your PSM's?' Proceedings of the 1998 knowledge acquisition workshop, Calgary, 1998.

Breuker, J. 1994. 'A Suite Of Problem Types'. Chapter 4, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds.: Breuker, J; Van de Velde, W. Publishers IOS Press, 1994.

Breuker, J; Van de Velde, W. 1994. '*Common*KADS library for expertise modelling: reusable problem-solving components'. Publishers IOS Press, 1994.

Brown, D. C; Dunskus, B. V; Grecu, D. L; Berker, I. 1995. SINE: Support for single function agents'. Applications Of Artificial Intelligence In Engineering 10 - Intl. Conf. 1995. Pages 525-532. Edited By Rzevski, Adey and Tasso. Pub. Computational Mechanics Publications, Southampton, Boston.

Brown, D. C; Birmingham, W. P. 1997. 'Understanding the nature of design'. IEEE Expert - Expert Intelligent Systems, Vol. 12, No 2, Special edition on AI in design. March – April 1997.

Bucciarelli, L. L. 1988. 'An ethnographic perspective on engineering design'. Design Studies Vol. 9, No 3, July 1988.

Buchanan, B. G; Shortliffe, E. H. (eds.) 1984. 'Rule-Based Expert Systems'. Addison Wesley 1984.

Bull, J. 1993. 'Life-cycle costings for construction'. Blackie Academic and Professional. Glasgow 1993.

Burton, R. R. 1982. 'Diagnosing Bugs In A Simple Procedural Skill'. In Sleeman, D. And Brown, J. S. (eds.) 'Intelligent Tutoring Systems' pages 157-183. Academic Press 1982.

Bussmann, S; Muller, J. 1993. 'A communication architecture for cooperating'. Journal of Computer and artificial intelligence, Vol. 12, Issue 1, pages 37-53.

Bylander, T; Chandrasekaran, B. 1987. 'Generic tasks for knowledge-based reasoning. The 'right' level of abstraction for knowledge acquisition'. International Journal of man machine studies, 26, pages 231-243, 1987.

Candy, L; Edmonds, E. A. 1997. 'Supporting the creative user: a criteria based approach to interaction based design'. Design Studies, April 1997, Vol. 18, No. 2.

Canter, D; Brown, J; Groat, L. 1985. 'A multiple sorting procedure for studying conceptual systems'. From 'The Research Interview: Uses and Approaches'. Publisher: Academic Press, 1985. Eds.: Brenner, M; Brown, J; Canter, D.

Carter, D. E. 1994. 'Concurrent engineering'. Proceedings of the international conference on concurrent engineering and electronic design automation (CEEDA), Bournemouth, UK, 7-8 April, 1994. Edited by Sa'ad Medhat, Society for computer simulation international.

Cha, J; Guo, W. 1993. 'The Methodology And Environment For Modelling And Implementation In Concurrent Engineering'. Advances In design Automation, ASME, Design Engineering Division, Vol. 65, part 1, pages 51-56. ASME, New York, 1993.

Chakrabarti, A; Tang, M. X. 1996. 'Generating conceptual solutions on FUNCSION: Evolution of a functional synthesiser'. Artificial intelligence in design, Eds. J.S. Gero, F. Sudweeks 1996. Pub. Kluwer academic publishers.

Chandrasekaran, B. 1987. 'Towards a functional architecture for intelligence based on generic information processing tasks'. Proc. of the 10th joint international conference on Artificial Intelligence, pages 1183-1192, Milan.

Chandrasekaran, B. 1990. 'Design problem-solving: A task analysis'. AI magazine, Winter 1990.

Chandrasekaran, B. 1994. 'Functional representation: A brief historical perspective'. Applied Artificial Intelligence, April-June 1994, Volume 8, Number 2, pages 173-197.

Chandrasekaran, B; Johnson, T. R; Smith, J. W. 1992. 'Task-structure analysis for knowledge modelling'. Communications of the ACM September 1992, Vol. 35, No. 9.

Charniak, E; McDermott, D. 1985. 'Introduction to Artificial Intelligence'. Addison-Wesley.

Chawla, A; Sangal, R. 1992. 'An Intelligent Design System'. Applications Of Artificial Intelligence In Engineering 7 - Intl. Conf. 1992. Pages 807-824. Edited By Grierson, Rzevski and Adey. Pub. Computational Mechanics Publications, Southampton, Boston.

Chern, J. H. 1991. 'Knowledge-based Engineering In Concurrent Engineering Automation'. Journal of Applications of Artificial Intelligence in Engineering, 1992, pp. 289-302. Pub. Computational Mechanics Publications, Southampton, England.

Churchland, P. M. 1989. 'A neurocomputational perspective: The nature of mind and the structure of science'. MIT Press, 1989.

Clancey, W. 1983. 'The epistemology of a rule-based expert system. A framework for explanation'. Artificial Intelligence 20:3, 1983, pages 215-251.

Clancey, W. J. 1985. 'Heuristic classification'. Artificial Intelligence 27, pages 289-350.

Clark, P. 1990. 'Representing knowledge as arguments: Applying expert system technology to judgmental problem-solving'. In R. Addis and R. M. Muir (eds.) 'Research and development in expert systems', 7, pages 147-159. Cambridge University Press, 1990.

Colton, J. S; Dascanio, J. L. 1991. 'An integrated intelligent design environment'. Engineering with computers, 7, pages 11-22, 1991. Pub. Springer-Verlag, New York Inc.

Condoor, S. S; Shankar, S. R; Brock, H. R; Burger, C. P; Jansson, D. G. 1992. 'A cognitive framework for the design process'. DE Vol. 42, Design theory and methodology, ASME 1992.

Coyne, R.D; Rosenman, M. A; Radford, A. D; Balachandran, M; Gero, J. S. 1990. 'Knowledge-based design systems'. Addison Wesley publishing company.

Cross, N. 1989. 'Engineering design methods: Strategies for product design. Second Edition'. Pub. John Wiley and sons, Chichester, 1994.

Cross, N; Christiaans, H; Dorst, K. 1996. 'Analysing design activity'. Workshop at Delft University of Technology, The Netherlands Pub. John Wiley and sons. 1996.

Cutkosky, M. R; Tenenbaum, J. M. 1991. 'Providing computational support for concurrent engineering'. International journal of Systems Automation: Research and applications. Volume1, Issue 3, pages 239-261.

Culley, S. J; Owen, G. W; Pugh, P. 1996. 'Current issues in design - survey'. Published by Bath University in conjunction with IMeche.

Darr, P. D; Birmingham, W. P. 1994. 'Automated design for concurrent engineering'. IEEE Expert, October 1994, pages 35-42.

David, J, M; Krivine, J, P; Simmons, R. 1993. 'Second generation expert systems'. Springer Verlag 1993.

Davies. S. P. 1995. 'Effects of concurrent verbalisation on design problem-solving'. Design Studies 16:2, pages 102-116

de Groot, 1965. 'Thought and choice in Chess', Mouton, The Hague, 1965.

de Hoog, R; Benus, B; Metselaar, C; Vogler, M. 1992. 'Applying the *Common*KADS Organisation Model'. ESPRIT Project P5248 KADS-II, KADS-II/T M6/DM6.1/UvA/036/1.0, University of Amsterdam, 1992.

de Hoog, R; Martil, R; Wielinga, B; Taylor, R; Bright, C; van de Welde, W. 1993(a). 'The *Common*KADS model set'. ESPRIT project report, project number P5248 KADS 2, Publisher: University of Amsterdam, 1992.

de Hoog, R; Benus, B; Metselaar, C; Vogler, M. 1993(b). 'The *Common*KADS Organisation Model'. ESPRIT Project P5248 KADS-II, KADS-II/TM6/DM6.2/UvA/041/1.0, University of Amsterdam, 1993.

Desa, S; Schmitz, J.M. 1991. 'The development and implementation of a comprehensive concurrent engineering method: theory and application'. SAE Transactions, 1991, V. 100, pages 1041-1049.

Dewhurst, P; Abbatiello, N. 1996. 'Design for service, Design for X - Concurrent Engineering Imperatives'. Chapman & Hall, 1996.

Dieng, R; Corby, O; Labidi, S. 1994. 'Agent-based knowledge acquisition'. From 'A future for knowledge acquisition', Proc of the 8th European Knowledge acquisition workshop, EKAW'94, Eds. Steels, L; Schreiber, G; Van De Welde, W. Springer Verlag, Belgium 1994.

Dominique, J; Motta, E; Watt, S. 1993. 'The emerging VITAL workbench'. In Knowledge acquisition for knowledge-based systems: Seventh European Workshop - EKAW '93, pages 320-339, Toulouse and Caylus, France, Springer - Verlag.

Dorst, K. 1995. 'Analysing design activity: new directions in protocol analysis.' Design Studies 16:2 139-142.

Dorst, K; Dijkhuis, J. 1995. 'Comparing paradigms for describing design activity.' Design Studies 16:2 261-274.

Druckman, D. 1973. 'Human factors in international negotiations: Social - psychological aspects of international conflict'. Beverly Hills, Sage Publications.

Duursma, C; Olsson, O; Sundin, U. 1994. 'Task model definition and Task Analysis process'. ESPRIT Project P5248 KADS 2 Report, Pub. University of Amsterdam.

Dwivedi, S. N; Kulpa, Z; Sobolewski, M. 1993. 'Graphical And Natural Language Interface With A Knowledge-based Concurrent Engineering Environment'. Computers In Industry 23 (1993), pages 175-184. Pub. Elsevier Science Publishers.

Ennis, C. W; Gyeszly, S. W. 1991. 'Protocol analysis of the engineering design process'. Research in engineering design, 3(1), 1991.

Ertas, A; Jones, J. C. 1993. 'The engineering design process'. Pub. John Wiley and sons inc. 1993.

Evans, S. 1991. 'Concurrent Engineering - Changing Your Way To A Better Business'. Computers, May 1991. Innopress Ltd., Kent, UK.

Favela, J; Imai, K; Connor, J. J. 1994. 'Hypermedia support for collaborative design'. Design Studies No.1, Vol. 15, January 1994. Pub. Butterworth-Heinemann Ltd.

Finger, S; Fox, M; Prinz, F; Rinderle, J. 1992. 'Concurrent design'. Applied artificial intelligence No. 6, 1992. pages 257-283, Hemisphere Publishing Corp.

Firlej, M; Hellens, D. 1991 'Knowledge Elicitation: A Practical Handbook', Prentice Hall.

Fischer, G. 1990. 'Communication requirement for cooperative problem-solving systems'. Journal of information systems, Vol. 15, No 1, 1990, pages 21-36.

Fischer, G; Mastaglio, T. 1991. 'A Conceptual Framework For Knowledge-Based Critic Systems'. Decision Support Systems Vol. 7, No. 4, Nov. 1991 pages 355-378.

Fischer, G, Lemke, A. C, Mastaglio, T. 1991(a). 'Critics: An Emerging Approach To Knowledge-based Human Computer Interaction'. International Journal Of Man-Machine Studies (1991) 35, pages 695-721.

Fischer, G; Lemke, A. C; Mastaglio, T; Morch, A. I. 1991(b). 'The Role Of Critiquing In Cooperative Problem-solving'. ACM Transactions On Information Systems, Vol. 9, No. 3, April 1991. Pages 123-151.

Fischer, G; Nakakoji, K; Ostwald, J; Stahl, G; Sumner, T. 1993(a). 'Embedding Computer-based Critics In The Contexts Of Design'. Conf. Proc. On Human Factors In Computing Systems 1993, pages 157-164, 1993. ACM, New York.

Fischer, G; Nakakoji, K; Ostwald, J; Stahl, G; Sumner, T. 1993(b). 'Embedding Critics In Design environments'. The Knowledge Engineering Review, Vol. 8: 4, 1993, pages 285-307.

Fraser, J. 1995. 'The Enterprise tool set - an open enterprise architecture'. Workshop on intelligent manufacturing systems, IJCAI 95, Montreal, Canada, August 1995.

Fruchter, R; Clayton, M, J; Krawinkler, H; Kunz, J; Teicholz, P. 1993. 'Interdisciplinary Communication of Design Critique In The Conceptual Design Stage'. Computing In Civil And Building Engineering. Proceedings Of The Fifth Intl. Conf. On Computing In Civil And Building Engineering (ICCCBE) 1993, pages 377-384. Pub. ASME, New York.

Gagdas, G. 1996. 'A shape grammar model for designing row houses'. Design studies, Vol. 17, No 1, Jan 1996.

Gaines, B; Lister, M. 1990. 'Development of Second Generation Knowledge Acquisition Systems'. From 'Current trends in knowledge acquisition'. Eds. Wielinga, Boose, Gaines, Schreiber, Van Someren. IOS Press, Amsterdam.

Gaines, B. R; Norrie, D. H. 1994. 'Mediator: Information and knowledge management for the virtual factory'. AIAI-94 workshop, Reasoning about the shop floor. Menlo Park. California. 1994.

Galle, P. 1996. 'Design rationalisation and the logic of design: A case study'. Design Studies, Vol. 17, No 3, July 1996.

Gavrilova, T; Voinov, A. 1998. 'Work in progress: Visual specification of knowledge-bases'. 11th International Conference on Industrial and Engineering applications of artificial intelligence and expert systems, IEA98. Benicassim, Castellon, Spain. Pub. Springer 1998.

Glaser, B. G; Strauss, A. L. 1967. 'The discovery of grounded theory'. Chicago, Aldine.

Gero, J. S. 1990. 'Design prototypes: A knowledge representation schema for design'. AI Magazine, Winter 1990.

Gero, J. S; McNeill, T. 1998. An approach to the analysis of design protocols'. Design Studies 19 (1998) pages 21-61.

Goel, A. 1989. 'Integration of Case-based reasoning and model based reasoning for adaptive design problem-solving', Ph.D. dissertation, Dept. of computer and information science, Ohio state University, US.

Goguen, J. A; Linde, C. 1993. 'Techniques for requirements elicitation'. Proceedings of the International symposium on requirements engineering, 1993, pages 152 – 164. Published by IEE.

Goldstein, I. 1982. 'The Genetic Graph: A Representation For The Evolution Of Procedural Knowledge'. In Sleeman, D. And Brown, J. S. (eds.) 'Intelligent Tutoring Systems', pages 51-79. Academic Press 1982.

Gruber, T. R; Russell, D. M. 1991. 'Design knowledge and design rationale: A framework for representation, capture and use'. Knowledge Systems Laboratory, Technical Report KSL 90-45. Knowledge systems laboratory, Computer Science Department, Stanford University, Stanford, California. August 1991.

Guan, X; McCallum, K. J. 1996. 'Adopting a minimum commitment principle for computer aided geometric design systems'. Artificial Intelligence In Design 96, Eds. Gero, J; Sudweeks, F. Publishers Kluwer Academic Press, 1996.

Hammer, M; Champy, J. 1993. 'Re-engineering the corporation'. Harper, New York.

Hart, A. 1985. 'The role of induction in knowledge elicitation'. Expert systems, Vol. 2(1) 1985.

Hagglund, S. 1993. 'Introducing Expert Critiquing Systems'. The Knowledge Engineering Review, Vol. 8: 4, 1993, pages 281-284.

Harrington, J. V; Soltan, H; Forskitt, M. 1995. 'Negotiation in a concurrent engineering design environment'. Expert systems, May 1995, Vol. 12, No 2.

Hedberg, R. S. 1994. 'Design of a lifetime'. BYTE, October 1994.

Hernandez, J; Luby, S. C; Hutchins, P. M; Leung, H. W; Gustavson, R. E; De Fazio, T. L; Whitney, D. E; Nevins, J. L; Edsall, A. C; Metzinger, R. W; Tung, K. K; Peters, T. J. 1991. 'An Integrated System For Concurrent Design Engineering'.
Proceedings of the 7th IEEE Conference on Artificial Intelligence Applications Part 2, Miami Beach, FL, USA, 24-28 Feb 1991, pages 205-211, Published by IEEE.

Hoffman, H. F. 1993. 'Requirements engineering: A survey of methods and tools''. Technical report no. 5, Institute of informatics, University of Zurich.

Huang, G. Q; Brandon, J. A. 1993. 'Agents for cooperating expert systems in concurrent engineering design'. (AI EDAM) (1993) 7(3), pages 145-158. Pub. Academic Press Limited 1993.

Hubka, V. 1982. 'Principles of engineering design'. Pub. Butterworth Scientific.

Jackson, P. 1990. 'Introduction To Expert Systems - Second Edition'. Addison Wesley 1990.

Jacobson, I; Booch, G; Rumbaugh, J. 1998. 'The Unified Software Development Process.' Addison Wesley 1998

Jo, H. H; Parsaei, H. R; Wong, J. P. 1991. 'Concurrent engineering - the manufacturing philosophy for the 90's'. Computers and industrial engineering, 1991, Vol. 21, No. 1 - 4, pages 35-39.

Jo, H. H; Parsaei, H. R; Wong, J. P. 1992. 'Design frameworks for concurrent engineering'. Computers and industrial engineering, 1992, Vol. 23, No. 1 - 4, pages 11-14.

Joseph, S. 1996. 'Design systems and paradigms'. Design Studies 17 pages 227-239.

Kelly, G. A. 1955. 'Personal construct theory'. New York, Norton 1955.

Kelly, V. E. 1984. 'The CRITTER System: Automated Critiquing Of Digital Circuit Designs'. Proceedings Of The 21st Design Automation Conference. ACM / IEEE 1984, pages 419-425.

Kingston, J. 1993. 'Re-engineering IMPRESS and X-MATE using *Common*KADS'. Proc. Expert Systems 93, The 13th Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge, December 1993. ISBN 1 85598 020 7. pg. 17-42.

Kingston, J. K. C. 1994. 'Design by exploration: A proposed *Common*KADS inference structure'. Publisher: Edinburgh: Artificial Intelligence Applications Institute, University of Edinburgh, 1994 Series: AIAI-TR ; 155.

Kingston, J. K; Doheny J. G; Filby, I. M. 1995. 'Evaluation of workbenches which support the *Common*KADS methodology'. The knowledge engineering review, Vol. 10:3, 1995, pages 269-300.

Klein, M; Lu, S. C -Y. 1989. 'Conflict resolution in cooperative design'. Artificial intelligence in engineering, 1989, Vol. 4, No. 4. Computational Mechanics Publications.

Klein, M. 1991. 'Supporting conflict resolution in cooperative design systems'. IEEE Transactions on Systems, Man and Cybernetics. Volume 21, Issue 6, pages 1379 – 1390.

Klein, M. 1992. 'Detecting and resolving conflicts among human and machine based design agents'. Artificial Intelligence In Engineering, Volume 7, Issue 2, pages 93 – 104.

Kolodner, J. L. 1993. 'Case-Based Reasoning'. Morgan Kaufmann.

Kolodner. JL 1996. 'Powers of observation in creative design'. Design Studies 17:4, pages 385-416.

Kott, A; Cederquist, A; Kollar, C. 1990. 'Modelling And Control Of A Multi Agent Engineering Process: An AI Approach To Concurrent Engineering'. Concurrent Engineering Of Mechanical Systems - ASME , Design Engineering Division, vol. 22. pages 47-54. Pub. ASME, New York, 1990.

Krishnamoorthy, C. S; Rajeev, S; Karimulla Raja, S; Shiva Kumar, H. 1991. 'A development environment for knowledge-based systems in engineering design'. Proceedings of the Second International conference on Application of Artificial Intelligence techniques to Civil and Structural Engineering, Oxford, England, pages 165-174.

Kruger, C; Wielinga, B. 1993. 'A KADS model for the industrial design task'. Proceedings of the third KADS meeting, Munich, Germany 1993.

Lander, S. E; Lesser, V. R. 1993. 'Understanding the role of negotiation in distributed search among heterogeneous agents'. In proceedings of the Thirteenth International Joint conference on Artificial Intelligence, pages 438-444, Chamberry, France, August 1993.

Lander, S. E. 1997. 'Issues in Multiagent Design Systems'. IEEE Expert Intelligent Systems, Vol. 12, No 2, Special edition on AI in design. March – April 1997.

Landauer, T. K. 1995. 'The trouble with computers'. Cambridge, MA., Academic Press.

Langlotz, C. P, Shortliffe, E. H. 1983. 'Adapting A Consultation System To Critique User Plans'. International Journal Of Man Machine Studies. 1983. 19, pages 479-496.

Lawson, B. 1990. 'How designers think'. Butterworth, 1990.

Liddament, T. 1999. 'The computationalist paradigm in design research'. Design Studies 20, 1999, pages 41 – 56.

Ligman, D. E. 1990. 'DEIMOS: A functional paradigm for mechanical design'. Proceedings of the 3rd International Conference on Industrial and Engineering applications of Artificial Intelligence and Expert Systems - IEA/AIE 90, Charleston, SC, USA, 15-18 Jul 1990, (Conf. code 14500) pp.773-780 Publisher: ACM, New York, NY, USA.

Linster, M. 1993. 'Explicit and operational models as a basis for second generation knowledge acquisition tools'. From 'Second generation expert systems'. Eds. David, J, M; Krivine, J, P; Simmons, R. Springer Verlag 1993.

Lloyd, P; Scott, P. 1994. 'Discovering the design problem'. Design Studies 15, 1994, pages 125-140.

Lloyd. P; Lawson. B; Scott. P 1995. 'Can concurrent verbalisation reveal design cognition?' Design Studies 16:2, pages 237-259.

Maher, M. L. 1990. 'Process models for design synthesis'. AI Magazine, Winter 1990.

Maher, M, L; Gomez de Silva, G. 1997. 'Case–Based reasoning in design'. IEEE Expert Intelligent Systems, Vol. 12, No 2, Special edition on AI in design. March – April 1997.

Major, N; Reichgelt, H. 1990. 'ALTO: An automated laddering tool'. In 'Current trends in knowledge acquisition', IOS Press, eds. Wielinga, Boose, Gaines, Scrieber, Van Someren.

March, J. L. 1984. 'The logic of design'. Developments in design methodology, Ed: Cross, N, Wiley.

Mastaglio, T. W. 1989. 'Computer-based critiquing: a foundation for learning environments'. Conference proceedings TITE '89 - Conference on technology and innovation in training and education, pages 125-136, March 6-9, 1989, Atlanta, Georgia. Editor: Linda Wiekhorst.

Matta N; Ros C; Corby O. 1998. 'A Generic Library to guide Decision Making in Concurrent Engineering'. Proceedings of TMCE'98 Tools and methods for Concurrent Engineering.

Matta, N 2000. ‘A concurrent engineering modelling library’. Ph.D. Thesis, University of Amsterdam, 2000. Published at http://www-sop.inria.fr/acacia//Cokace/Doc/cellb.html.

Mccarthy, J. 1994. ‘The state-of-the-art of cscw - cscw systems, cooperative work and organization’. Journal of information technology, 1994, vol.9, no.2, pages 73-83.

Mcdermott, J. 1982. ‘R1: A rule based configurer of computer systems.’ Artificial Intelligence, 19, pages 39-88.

Miles, B, Swift, K. 1994. ‘Developments In Computer Aided Concurrent Engineering Tools’. Conference Proceedings of ‘Design for Competitive Advantage’ - Sponsored by IMech.E No. C482, 1994.

Miller, P. L. 1984. ‘Lessons Learned: Design Parameters For A Critiquing System’. Expert Critiquing Systems - Practice Based Medical Consultation By Computer. Perry L. Miller. Springer Verlag - New York, Berlin.

Molina, A; Alashaab, A. H; Ellis, T. I. A; Young, R. I. M; Bell, R. 1995. ‘Review of computer -aided simultaneous engineering systems’. Research in engineering design - theory applications and concurrent engineering, 1995, Vol. 7, No. 1, pages 38-63.

Motta, E; Zdrahal, Z. 1995. ‘Parametric Design Problem-solving.’ Proceedings of Tenth Knowledge-Acquisition for Knowledge-Based Systems Workshop. Shareable and reusable problem-solving methods.

Musen, M, A. 1993. ‘An overview of knowledge acquisition’. Second generation expert systems, Springer Verlag, 1993.

Musen, M. A; Fagan, L; Combs, D. M; Shortliffe, E. H. 1987. ‘Use of a domain model to drive an interactive knowledge - editing tool’. International journal of man - machine studies, 26, pages 105-121.

Newell, A. 1982. ‘The knowledge-level’. Artificial Intelligence, 18, 1982, pages 87-127.

Newell, A; Simon, H. A. 1972. ‘Human problem-solving’. Prentice Hall, 1972.

Newell, A. 1990. ‘Unified theories of cognition’. Harvard University Press, 1990.

Olson, G. M; Olson, J. S; Carter, M. R; Storrosten, M. 1992. ‘Small group design meetings: An analysis of collaboration’. Human computer interaction, 1992, Vol. 7, pages 347-374.

Orady, E. A; Shareef, I. 1993. 'Expert System Design Philosophy For Application Of Simultaneous Engineering In Industry'. Design For Manufacturability 1993 ASME, Design Engineering Division DE v 52, 1993 pages 151-157. ASME, New York.

Orsvarn, K; Olsson, O; Hassan, H. A. 1994. 'Guidelines for the select - modify approach'. Chapter 7, 'Expertise model definition document'. Ed Wielinga, B. ESPRIT Project P5248 report, Pub. University of Amsterdam.

Oxman, R. 1995. 'Observing the observers: Research issues in analysing design activity'. Design studies 16 pages 275-283.

Pahl, G; Beitz, W. 1984. 'Engineering design'. Copyright The Design Council, 1984. Published by Springer Verlag.

Parkinson, B; Short, C. 1994 'The teaching of a concurrent engineering approach to design and manufacturing engineers'. Proc. of the international conference on concurrent engineering and electronic design automation (CEEDA), Bournemouth, UK. 7-8 April, 1994. Society for computer simulation international.

Petrie, C; Cutkosky, M. R. 1994. 'Design space navigation as a collaborative aid'. Third International conference on AI in Design, Lausanne, August 1994.

Petrie, C; Webster, T. A; Cutkosky, M. R. 1995. 'Using pareto optimality to coordinate distributed agents'. AIEDAM Vol. 9, 1995, pages 269-281. Special issue on conflict management.

Prasad, B. 1996. 'Concurrent engineering fundamentals. Volume 1: Integrated product and process organisation'. Pub. International Society Of Product Enhancement (ISPE). 1996.

Pugh, S. 1991. 'Total Design: Integrated Methods For Successful Product Engineering'. Addison Wesley, 1991.

Purcell, T; Gero, J. 1996. 'Design and other types of fixation'. Design Studies, Vol. 17, No 4, October 1996.

Rademakers, P. 1991. 'Task analysis of an equipment assignment problem'. AI-MEMO 91-1, VUB AI Lab, Pleinlaan 2, B-1050 Brussels, Belgium.

Rademakers, P; Van welenhuysen, J. 1993. 'Generic models and their support in modelling problem-solving behaviour'. From 'Second generation expert systems'. Eds. David, J, M; Krivine, J, P; Simmons, R. Springer Verlag 1993.

Rajeev, S; Suresh, S; Krishnamoorthy, C. S. 1993. 'A Criticism Based Model For Cooperative Problem-solving'. Computing Systems In Engineering . Vol. 4, Nos. 2-3, pages 201-210, 1993. Pub. Civil - Comp. And Pergamon Press Ltd.

Ramscar, M; Lee, J; Pain, H. 1996. 'A cognitively based approach to computer integration for design systems'. Design Studies, Vol. 17, No 4, October 1996.

Rankin, I. 1993. 'Natural Language Generation In Critiquing'. The Knowledge Engineering Review, Vol. 8: 4, 1993, pages 329-347.

Ranky, P. G. 1994. 'Concurrent / Simultaneous Engineering. (Methods, Tools And Case Studies)'. Cimware Ltd.

Rugg, G; McGeorge, P. 1997. 'The sorting techniques: a tutorial paper on card sorts, picture sorts and item sorts'. Expert systems, May 1997, Vol. 14, No 2.

Sanderson, A. C; Homen de Mello, L. S; Zhang, H. 1990 'Assembly sequence planning'. AI magazine, Spring 1990.

Sandholm, T; Lesser, V. 1995. 'Coalition formation among bounded rational agents'. 14th International Joint Conference on Artificial Intelligence (IJCAi-95), Montreal, Canada, pages 662 – 669.

Schmidt, R. F; Schmidt, M. 1996. 'Computer Aided Concurrent Integral Design' Esprit Research Project Report. Project 5168 CACID: Volume 1. Published by Springer Verlag 1996.

Schon, 1991 'The reflective practitioner: how professionals think', Avebury.

Schreiber, G; Wielinga, B; Breuker, J. 1993. 'KADS A principled approach to knowledge-based systems development'. Academic Press (Harcourt Brace Jovanovich Publishers) 1993.

Schreiber, G; Wielinga, B; de Hoog, R; Akkermans, J. M; van de Welde, W. 1994(a). '*Common*KADS: A comprehensive methodology for KBS development'. IEEE Expert, 9(6), pages 28-37.

Schreiber, G; Wielinga, B; Akkermans, H; van de Welde, W. 1994(b). 'CML; The *Common*KADS Conceptual Modelling Language'. Chapter 3, 'Expertise model definition document'. Ed Wielinga, B. ESPRIT Project P5248 report, Pub. University of Amsterdam.

Schreiber, G; Akkermans, H; Anjewierden, A; deHoog, R; Shadbolt, N; van de Welde, W; Wielinga, B. 1999. 'Knowledge engineering and management. The *Common*KADS methodology'. MIT Press, Cambridge, Massachusetts, 1999.

Sharpe, J. E. E; Bracewell, R. H. 1993. 'Application Of Bond Graph Methodology To Concurrent Conceptual Design Of Interdisciplinary Systems'. Proc. Of The IEEE Intl. Conf. on Systems, Man and Cybernetics Vol. 1. 1993. Pages 7-13. Pub. IEEE, IEEE Service Centre, Piscataway, NJ.

Shaw, M. L. G; Gaines, B. R. 1989. 'Comparing conceptual structures: consensus, conflict, correspondence and contrast'. Knowledge Acquisition, December 1989.

Shifman, M. 1990. 'Building Critiquing Expert Systems In Prolog'. Proceedings Of The 1990 Symposium On Applied Computing - SAC '90. IEEE NJ, USA. 1990.

Shiva Kumar, H; Suresh, S; Krishnamoorthy, C. S; Fenves, S. J; Rajeev, S. 1994. 'GENCRIT: A tool for knowledge-based critiquing in engineering design'. Artificial intelligence for engineering design, analysis and manufacturing (1994), 8, pages 239-259. Pub. Cambridge University Press.

Siddiqi, J; Shekaran, M. C. 1996. 'Requirements engineering: The emerging wisdom'. IEEE Software March 1996.

Silverman, B. G; Wenig, R. G. 1993. 'Engineering Expert Critics For Cooperative Systems'. The Knowledge Engineering Review, Vol. 8: 4, 1993, pages 309-328.

Silverman, B. G. 1992. 'Survey Of Expert Critiquing Systems: Practical And Theoretical Frontiers'. Communications Of The ACM. April 1992 / Vol. 35, No. 4.

Silverman, B. G. Mezher, T. M. 1992. 'Expert Critics In Engineering Design: Lessons Learned And Research Needs'. AI. Magazine Spring 1992.

Simon, H. 1969. 'The Sciences of the Artificial'. MIT Press, 1969.

Smithers, T; Conkie, A; Doheny, J; Logan, B; Millington, K; Tang, M. X. 1990. 'Design as intelligent behaviour: an AI in design research programme'. Artificial intelligence in engineering, 1990, Vol. 5, No. 2.

Smithers, T. 1996. 'On knowledge-level theories of design process'. Artificial intelligence in design, Eds. J.S. Gero, F. Sudweeks 1996. Pub. Kluwer academic publishers.

Sohlenius, G. 1992. 'Concurrent Engineering'. Annals of the CIRP Vol. 41 No.2 1992.

Sommerville, I. 1992. 'Software engineering - fourth edition'. Addison - Wesley.

Sonnenwald, D. 1996. 'Communication roles that support collaboration during the design process'. Design Studies, Vol. 17, No. 3. July 1996.

Speel, P-H; Shadbolt, N; et al. 1999. 'Knowledge Mapping for Industrial Purposes'. KAW 99, Banff, Ca.
http://sern.ucalgary.ca/KSI/KAW/KAW99/papers/Speel1/index.html

Spurr, K; Layzell, P; Jennison, L; Richards, N. 1993. 'Software assistance for business re-engineering'. 1993. Wiley.

Sriram, D; Stephanopoulos, G; Logcher, R; Gossard, D; Groleau, N; Serrano, D; Navinchandra, D. 1989. 'Knowledge-based System Applications In Design: Research At MIT'. AI Magazine Fall 1989.

Staufffer, L. A; Ullman, D. G. 1988. 'A comparison of the results of empirical studies into the mechanical design process'. Design Studies, Vol. 9, No. 2, April 1988.

Steels, L . 1990. 'Components of expertise'. AI magazine, Summer 1990.

Stevenson, I; Chappell, C. 1994. 'IT based tools for concurrent engineering'. Computer Integrated Manufacturing - Proc. of the Eighth CIM - Europe Annual conference CEC DG xiii. Telecommunications, Information industries and innovation. pages 157-166. Eds.: O' Brien, C; Macconnaill, P; Van Puymbroeck, W. Pub. Springer Verlag, Germany, 1994.

Suchman, L. 1987. 'Plans and situated actions: The problem of Human-machine communications'. Cambridge University Press, 1987.

Sycara, K. 1989. 'Multiagent compromise via negotiation', in 'Distributed Artificial Intelligence, L. Gasser and M. Huns (Eds.) Vol. 2 Pitman pages 119-137, 1989.

Takeda, H; Veerkamp, P; Tomiyama, T; Yoshikawa, H. 1990. 'Modelling design processes'. AI Magazine, Winter 1990

Taleb-Bendiab, A; Oh, V; Sommerville, I; French, M. 1992. 'Knowledge representation for engineering design product improvement'. Applications of AI in engineering. 7 Th international conference, pages 807-824.

Tansley, D. S. W; Hayball, C. C. 1993. 'Knowledge-based Systems Analysis And Design - A KADS Developers Handbook'. Prentice Hall 1993.

Thurston, D. L. 1993. 'Concurrent engineering in an expert system'. IEEE transactions on engineering management, Vol. 40, No.2 May 1993.

Top, J; Akkermans, H. 1994. 'Engineering modelling'. Chapter 12, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds. Breuker, J; Van de Velde, W. Publishers IOS Press, 1995.

Toussaint, C; Forsmark, P; Pradera, A. 1994. '*Common*KADS Workbench User Guide' . ESPRIT Project P5248 report, Pub. University of Amsterdam.

Toye, G; Cutkosky, M. R; Leifer, L. J. 1994. 'SHARE: A methodology and environment for collaborative product development'. International journal of intelligent and cooperative information systems, 1994.

Ulrich, K. T; Eppinger, S. D. 1995. 'Product design and development'. McGraw Hill 1995.

Umeda, Y; Tomiyama, T. 1997. 'Functional reasoning in design'. IEEE Expert Intelligent Systems, Vol. 12, No 2, Special edition on AI in design. March – April 1997.

Valente, A; Lockenhoff, C. 1994. 'Assessment'. Chapter 7, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds. Breuker, J; Van de Velde, W. Publishers IOS Press, 1994.

Valente, A; Breuker, J; Van de Welde, W. 1994. 'The *Common*KADS Expertise modeling library'. Chapter 3, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds. Breuker, J; Van de Velde, W. Publishers IOS Press, 1994.

Valkenberg, R; Dorst, K. 1998. 'The reflective practice of design teams'. Design Studies 19, 1998, pages 249–271.

Van de Welde, W. 1993. 'Issues in knowledge-level modelling'. In David, J, M; Krivine, J, P; Simmons, R. 1993. 'Second generation expert systems'. Springer Verlag 1993.

Van de Welde, W. 1994. 'An overview of *Common*KADS'. Chapter 2, '*Common*KADS library for expertise modelling: reusable problem-solving components'. Eds. Breuker, J; Van de Velde, W. Publishers IOS Press, 1994.

Van de Welde, W; Duursma, C; Schreiber, A. T; Terpstra, P; Schrooten, R; Golfinopolous, V; Ollson, O; Sundin, U; Gustafsson, M. 1994. 'Design model and process.' KADS 2 Report, KADS2/M7/VUB/RR/064/2.0, VUB AI Lab.

VDI (Verein Deutscher Ingenieure) 1987. 'Design Guideline 2221: Systematic approach to the design of technical systems and products'. English translation of 1985 German edition. VDI, Verlag, Dusseldorf, 1987.

Victor, S. K; Brown, D. C; Bausch, J. J; Zenger, D. C; Ludwig, R; Sisson, R. D. 1993. 'Using multiple expert systems with distinct roles in a concurrent engineering system for powder ceramic components'. Applications of artificial intelligence in engineering 8. (Conference Toulouse, France 1993). Computational Mechanics 1993.

Visser. W 1996. 'Two functions of analogical reasoning in design: a cognitive psychology approach'. Design Studies 17:4.

Waern, A; Gala, S. 1993. 'The *Common*KADS Agent Model.' ESPRIT Project P5248 KADS 2, Version 1.2, Swedish Institute of Computer Science, Stockholm, Sweden.

Waern, A; Hook, K; Gustavsson, R; Holm, P. 1993. 'The *Common*KADS Communication Model.' ESPRIT Project P5248 KADS 2, Version 1.2, Swedish Institute of Computer Science, Stockholm, Sweden.

Wells, S. 1994. 'Data-driven modelling guidelines'. Chapter 6, 'Expertise model definition document'. Ed Wielinga, B. ESPRIT Project P5248 report, Pub. University of Amsterdam.

Werkman, K. J. 1991. 'Negotiation As An Aid In Concurrent Engineering'. Issues In Design / Manufacture Integration pages 23-30, Vol. 39, 1991. ASME, New York, 1991.

Wheeler, R. 1991. 'Small Work Projects: Teamwork Counts More Than Computer-based Tools'. IEEE Spectrum, July 1991, pages 32-33.

Wick, M. R. 1992. Expert system explanation in retrospect: a case study in the evolution of expert system explanation'. Journal of systems and software. Vol. 19, Number 2, October 1992, pages 159 – 169.

Wielinga, B; Boose, J; Gaines, B; Schreiber, G; Van Someren, M. 1990. 'Current trends in knowledge acquisition'. Eds. Wielinga, Boose, Gaines, Schreiber, Van Someren. IOS Press, Amsterdam.

Wielinga, B; Schreiber, G; Van de Velde, W; Akkermans, H. 1994(a). 'Components of the expertise model.' 'Expertise model definition document'. ESPRIT Project P5248 report, Pub. University of Amsterdam, Ed: Wielinga, B.

Wielinga, B; Akkermans, H; Schreiber, G. 1994(b). 'Model Validation.' 'Expertise model definition document'. ESPRIT Project P5248 report, Pub. University of Amsterdam, Ed: Wielinga, B.

Wielinga, B; Schreiber, G. 1997. 'Configuration-design problem-solving'. IEEE Expert Intelligent Systems, Vol. 12, No 2, Special edition on AI in design. March – April 1997.

Wielinga, B; Schreiber, G. 1998. 'Knowledge technology: Moving into the next millennium'. 11th International Conference on Industrial and Engineering applications of artificial intelligence and expert systems, IEA98. Benicassim, Castellon, Spain. Pub. Springer 1998.

Winston, P.H. 1984. 'Artificial Intelligence'. Addison Wesley.

Wong, S. T. C. 1994. 'Preference-based decision making for cooperative knowledge-based systems'. ACM Trans on Information Systems, 12:4, pages 407-435, 1994.

Young, R. E; Greef, A; O'Grady, P. 1994. 'An artificial intelligence based constraint network system for concurrent engineering'. International journal of Prod Res. Vol. 30, No. 7, 1994, pages 1715 – 1735.

Xue, D; Dong, Z. 1993. 'Object Oriented Knowledge Representation And Its Applications In Concurrent Design'. Conference Proceedings Of The IEEE 1993 Pacific Rim Conference On Communications, Computers And Signal Processing, pages 638-641. Pub. IEEE, IEEE Service Centre, Piscataway, NJ, USA.

Zlotkin, G; Rosenschein, J; S. 1996. 'Mechanism design for automated negotiation and its application to task-oriented domains'. Artificial Intelligence 86, pages 195-244.

Appendix A: KADS and *Common*KADS graphical task models to support the design process

A.1 The KADS library of generic task models

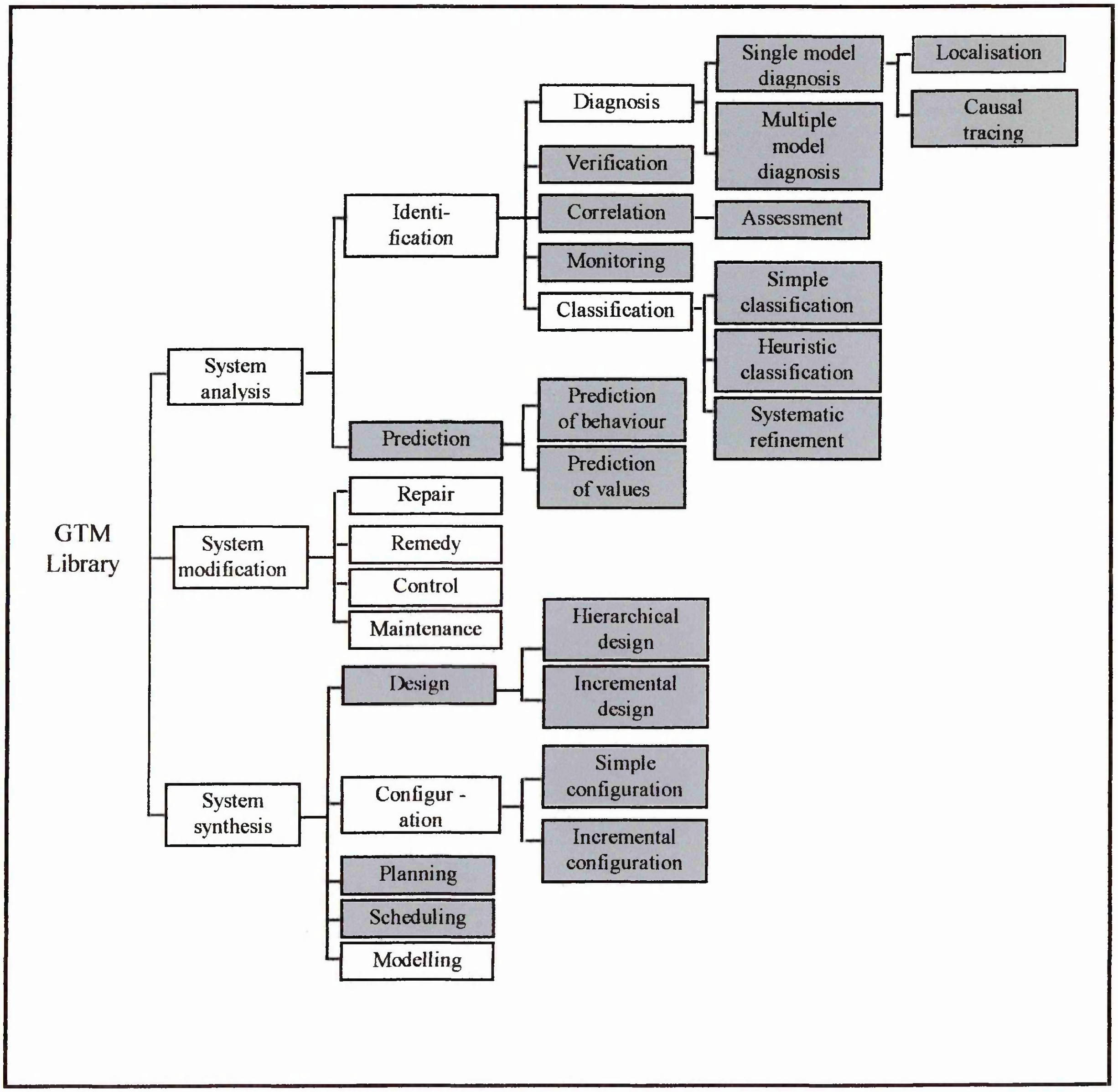


Figure A.1: The Common KADS Generic task Library

(from Tansley and Hayball [1993]). This illustrates the original KADS hierarchical structure of models to support different tasks, including different 'types' of design.

A.2 KADS Hierarchical design

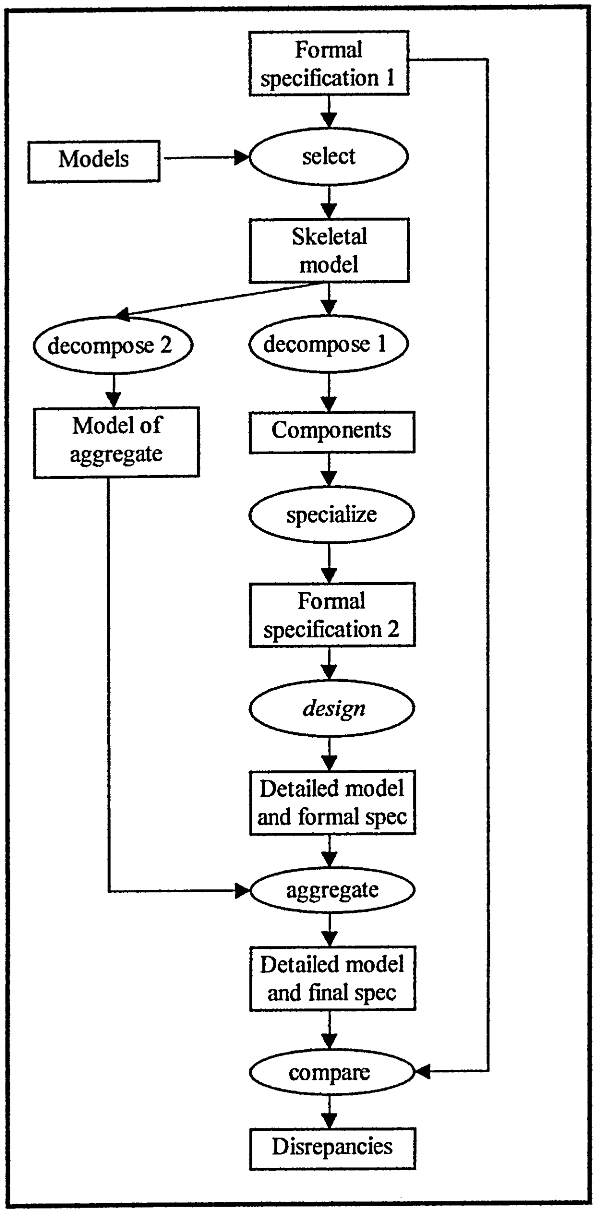


Figure A.2: Hierarchical design

(from Tansley and Hayball [1993])

A.3 KADS Incremental design

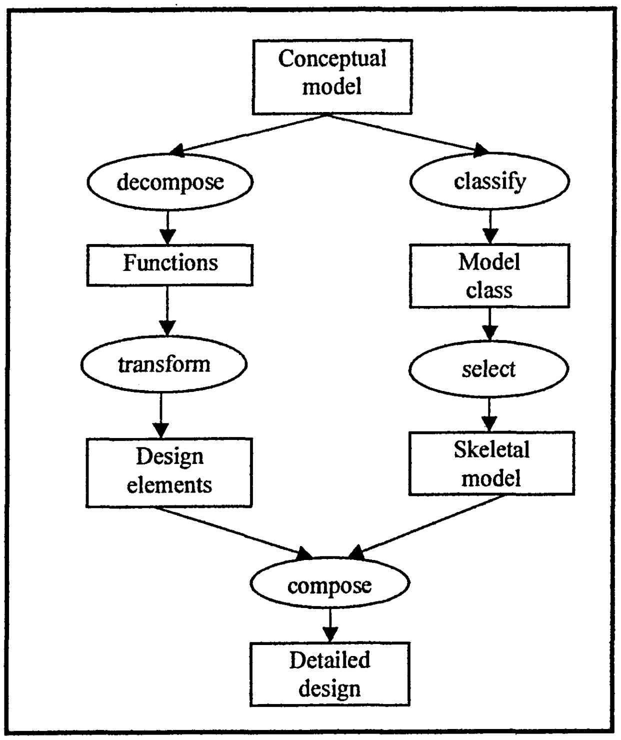


Figure A.3: Incremental design

(from Tansley and Hayball [1993])

A.4 *Common*KADS Analysis task models

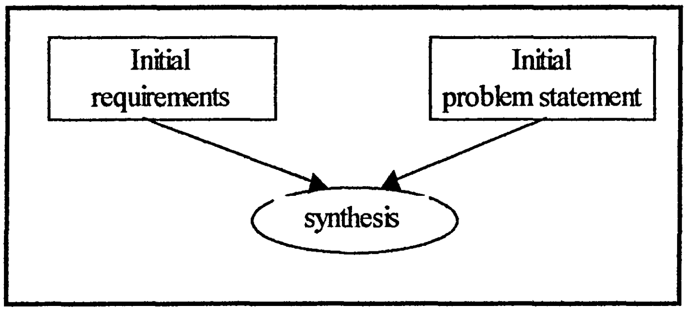


Figure A.4: *Common*KADS Analysis model 1

(From Bernaras and Van de Welde [1994[)

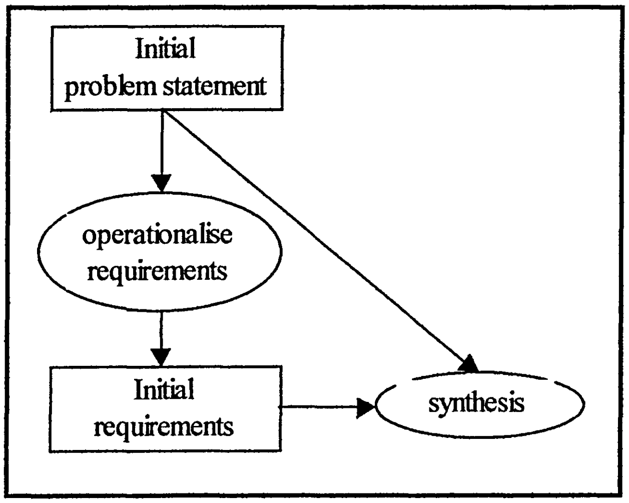


Figure A.5: *Common*KADS Analysis model 2

(From Bernaras and Van de Welde [1994[)

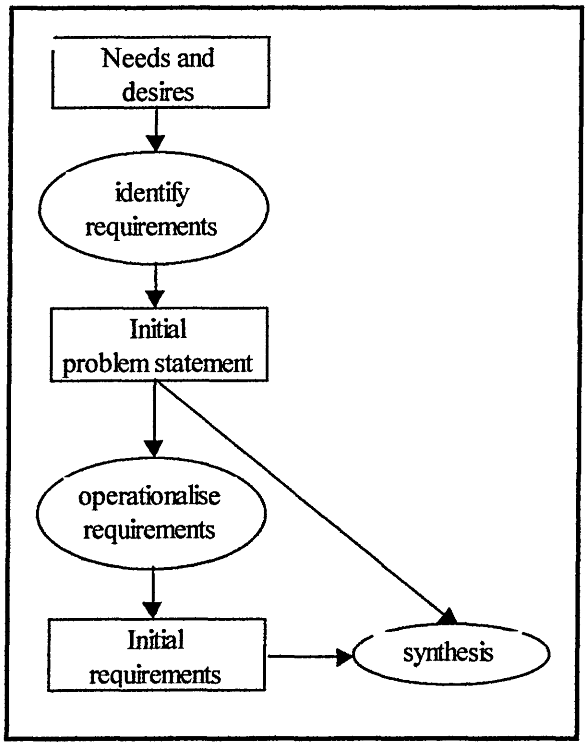


Figure A.6: *Common*KADS Analysis model 3

(From Bernaras and Van de Welde [1994[)

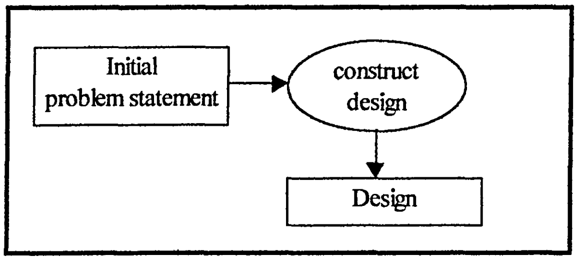


Figure A.7: *Common*KADS Routine design

(From Bernaras and Van de Welde [1994[)

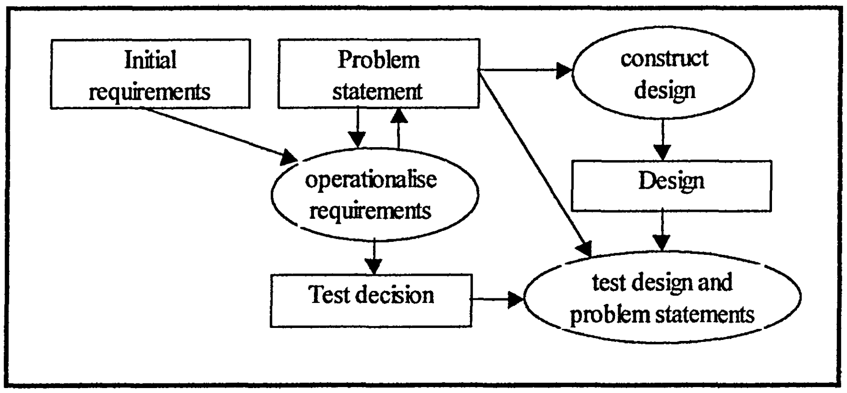


Figure A.8: *Common*KADS Innovative design

(From Bernaras and Van de Welde [1994[)

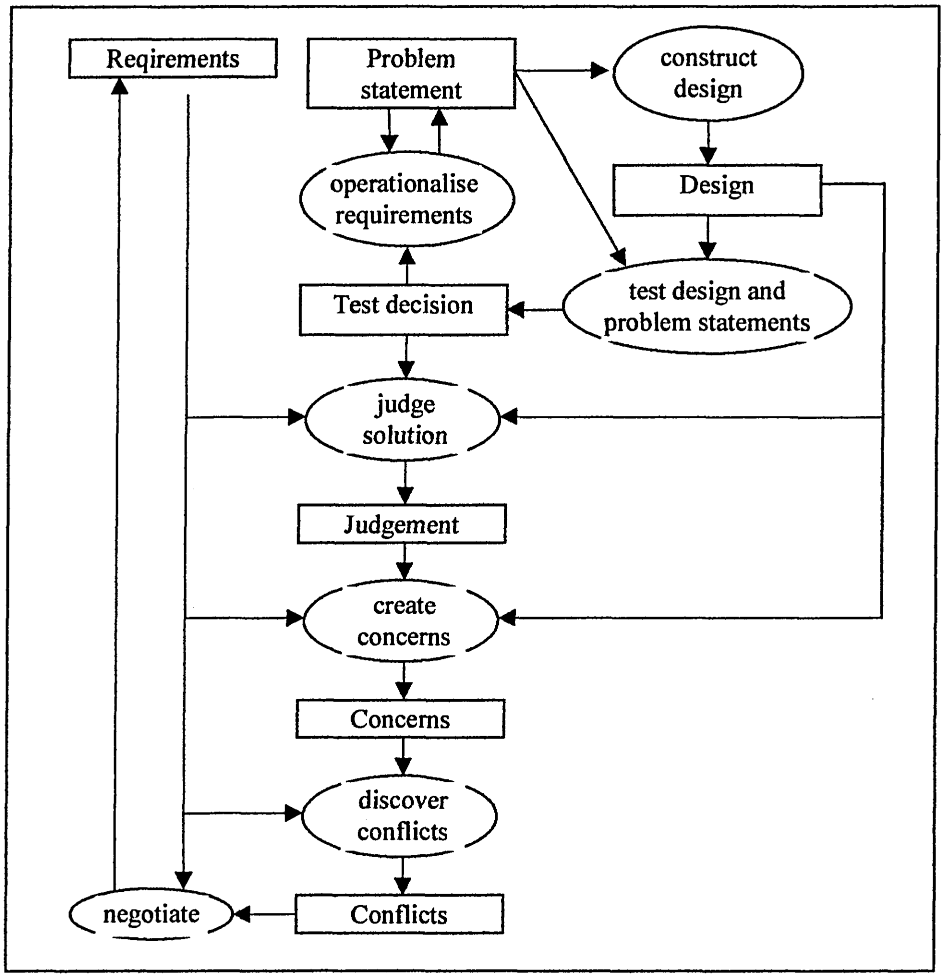


Figure A.9: ***Common*KADS Original design**

(From Bernaras and Van de Welde [1994[)

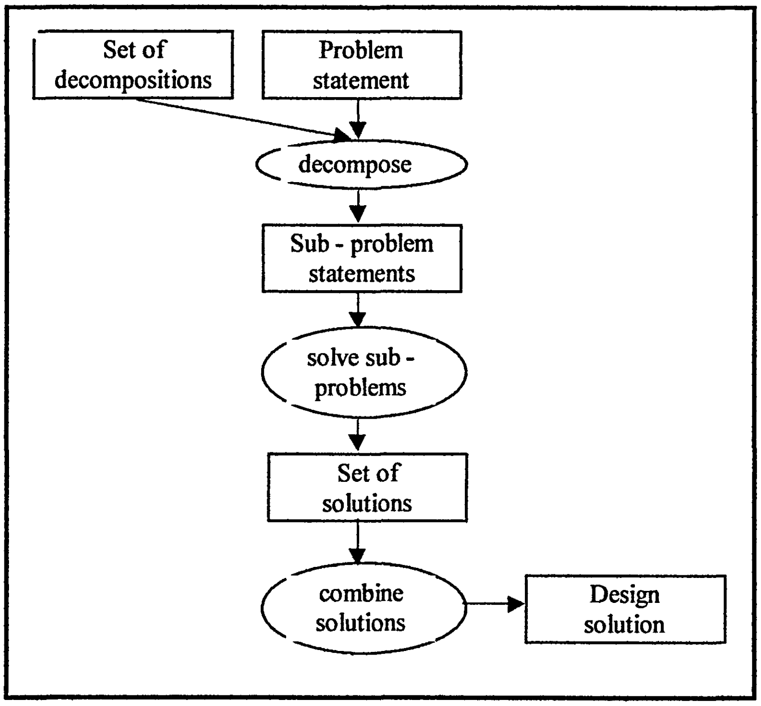


Figure A.10: ***Common*KADS Decomposition based design**

(From Bernaras and Van de Welde [1994[)

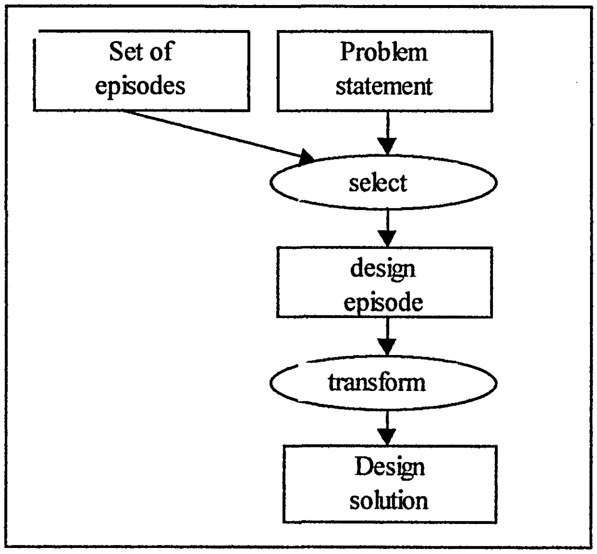


Figure A.11: ***Common*KADS Case-based design**

(From Bernaras and Van de Welde [1994[)

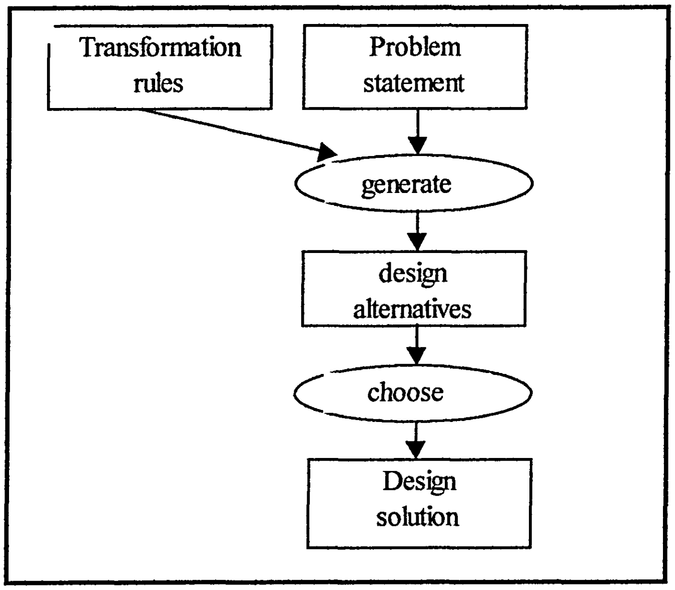


Figure A.12: ***Common*KADS Transformation based design**

(From Bernaras and Van de Welde [1994])

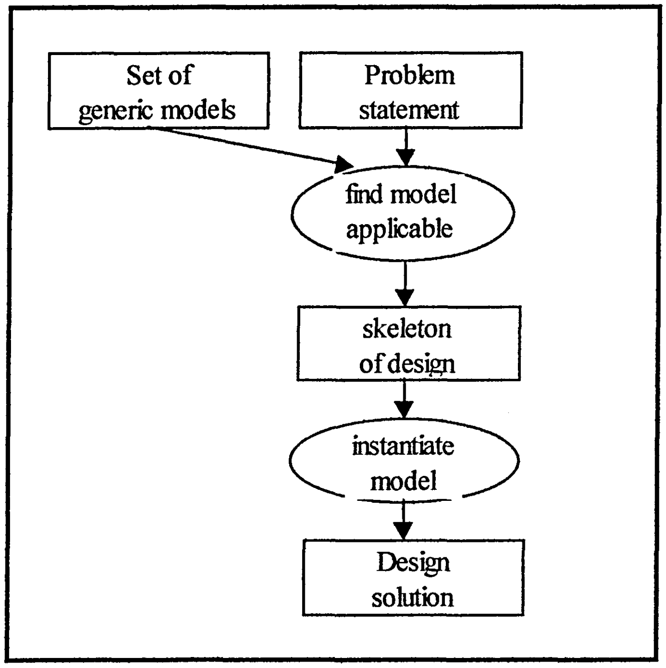


Figure A.13: ***Common*KADS Generic model design**

(From Bernaras and Van de Welde]1994])

A.5 Kingston's model for exploratory design

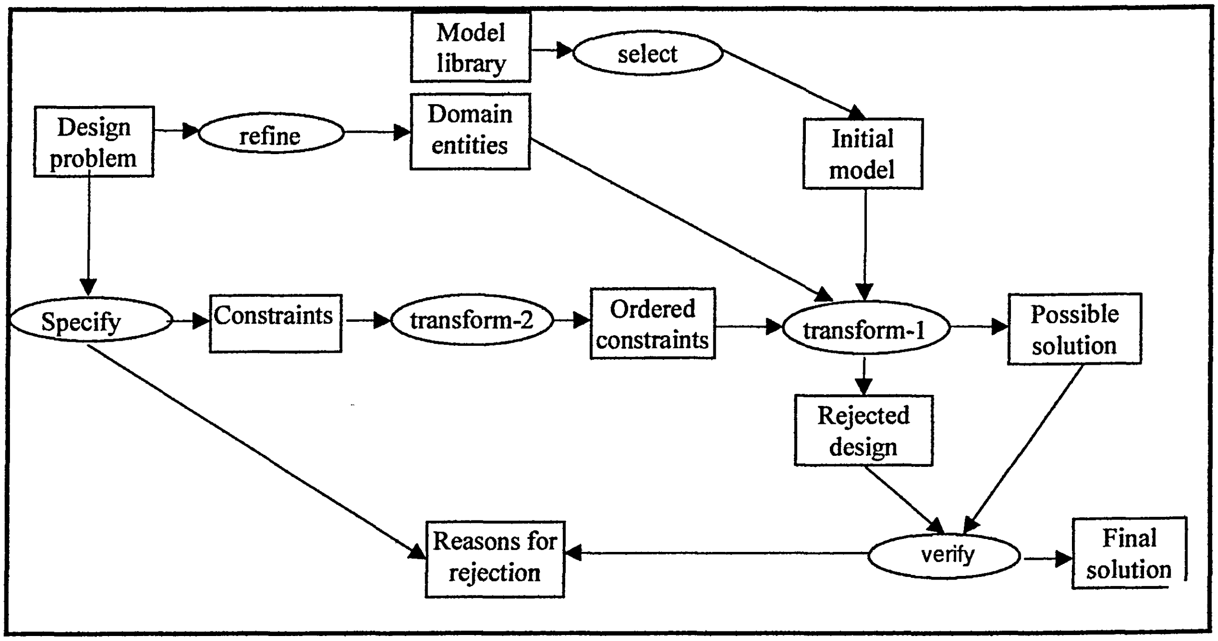


Figure A.14: Kingston's model for exploratory design

(from Kingston [1994])

Appendix B: Software tools to support the design process

B.1 Introduction

A number of different software tools have been developed to support both design and concurrent design. These range in scope from relatively simple, expert-system based tools which automate a small part of the design process to enterprise-wide software systems.

B.2 Tools to support CE

Cutkosky and Tenenbaum [1991] consider some of the characteristics of concurrent engineering and describe the 'Next-cut' system. This supports concurrent engineering design via a series of interacting software modules. They discuss how the system could be extended to support more complex design problems.

A number of tools have been developed to support the essential communication processes inherent within the design and concurrent design processes. The DICE tool - Sriram et al [1989] also allows users to communicate through a network with a 'blackboard system' controlling the interface between the different users. The DICETALK system - Dwivedi et al [1993] employs a dispatch-managing model to facilitate collaboration between team members. Favela et al [1994] look at the use of hypermedia links to facilitate communication within collaborative design. The 'Mediator' system (see Gaines and Norrie [1994) is an ambitious attempt to provide an open architecture information and knowledge management system. The different sub-systems comprising the system may be geographically dispersed and the system is intended to support the complete product life-cycle. A number of the systems make extensive use of techniques from artificial intelligence to model expertise within the resulting knowledge-based systems.

B.3 Expert Systems

The field of Expert Systems, ES, was one of the first areas where knowledge-based systems were implemented to solve real world problems. Jackson [1990] gives a good introduction to expert systems and early work in the field. Problems discovered during the development of these early expert systems led to so-called 'second generation' expert systems - see David et al [1993] for work in this area.

One of the most influential of the early expert systems was MYCIN. This diagnosed the identity of microorganisms responsible for bacterial infections. See Buchanan and Shortliffe [1984] for a fuller description. R1 / XCON was developed in the early 1980's to configure VAX computer systems and showed that they could successfully be used to solve real-life problems. See McDermott [1982] for further details.

The provision of explanations is a key facet in expert systems research. Wick [1992] outlines how explanations generated by expert systems have evolved and analyses the strengths and weaknesses of some of the techniques used.

B.4 Expert systems in design

Expert systems have also been used extensively in the field of design. A selection of systems includes the following:

Thurston [1993] outlines a framework for incorporating cost considerations in a design environment. They describe an expert system used in material selection for bumper design. This uses a rule-based approach with heuristics being used to separate more objective user actions from those that rely on assumptions from the user.

In the system presented by Colton and Dacanio [1991] an expert system is used to 'improve the designer's creativity by performing redundant and routine tasks'.

Bayliss et al. [1994] have looked at implementing the design for manufacture (DFM) philosophy - using an object oriented expert system toolkit called Kappa PC.

However, Rankin [1993] outlines some of the limitations of using expert systems in co-operative problem-solving roles. These are mainly to do with the designer losing control of the design process. Hence, while a designer may be happy to allow routine and boring aspects of design to be automated, they are generally not happy to lose control of the more important ‘creative’ aspects. As a result of these limitations, the concept of expert critiquing systems has been seen as a way of leaving the user in control of the problem-solving process with the computer system taking a more subordinate supporting role.

B.5 Expert Critiquing Systems

Rankin [1993] characterises the subtle difference between expert and critiquing systems as 'evaluating user given solutions to problems rather than presenting solutions to user given problems'.

Hence the critiquing approach can be seen as more suited to a concurrent design environment in that the designer is left in overall control of the process with individual critics able to critique an evolving design from their own perspective. Hagglund [1993] and Silverman [1992] give a general introduction to previous work in the area of critiquing.

One of the most influential of the early ECS was the ATTENDING system developed by Miller [1984]. This system analyses a physician’s plan (i.e. a solution) for a patient's treatment and based on knowledge of the patient critiques the proposed plan. Langlotz and Shortliffe [1983] also did some very influential early work on computer-based critiquing. This was implemented in their ONCOCIN system for critiquing physicians’ treatment plan.

Since these early steps work has been done on a number of aspects relating to critiquing systems. These include explanations in critic systems - Langlotz and Shortliffe [1983], Shifman [1990] and Rankin [1993], user modeling - Burton [1982], Goldstein [1982], knowledge elicitation for critiquing systems - Silverman and Wenig [1993] and methodologies for developing critiquing systems - Silverman and Mezher [1992].

B.6 Expert critiquing systems in design

Silverman and Mezher [1992] give a general introduction to expert critics in design.

In the engineering field, one of the first ECS to be developed was the CRITTER system by Kelly [1984]. This system evaluates digital circuit designs from a number of different perspectives.

Fischer and Mastaglio [1991a] and a variety of colleagues at Colorado University have developed a number of different critiquing systems and done extensive research in this area. In particular, Fischer et al [1993(b)] have looked at the possibilities of integrating critics in high functionality environments such as user interface development ('FRAMER') and design (JANUS - kitchen layouts).

B.7 Other knowledge-based systems in design

Expert systems and expert critiquing systems can be seen as specific types of tool to support the design process. Other knowledge-based systems, which are not specifically expert or critiquing systems, have also been developed. Coyne et al [1990] outline systems where knowledge-based support for design has been implemented.

The SPARK system (see Young et al [1994]) uses a constraint network with frame based inheritance to aid in the multi viewpoint design of a printing wiring board. This allows different viewpoints on the design to be represented using the same representation formalism.

The CACID system (see Schmidt and Schmidt [1996]) attempts to expand on the limitations of current CAD systems which generally represent only geometric data. This has been implemented in the C++ language using an object-oriented approach.

The FUNCSION system - Chakrabarti and Tang [1996] also uses functional analysis to generate different designs by matching the functionality of a product to a database of functional elements.

Orady and Shareef [1993] outline an approach which integrates the computer-assisted tools and databases that are needed by the CE team.

Hernandez et al [1991] describe a system which implements a feature-based design for assembly methodology. This uses complex representation techniques to allow estimates of whether different parts will match on assembly.

B.8 'Agent' based systems

Because concurrent design inherently implies that expertise from a number of different disciplines incorporated, it is inevitable that 'agent' based systems feature extensively in the literature on computer support for concurrent design.

The ACDS system, Darr and Birmingham [1998], exhaustively generates a space containing all possible designs, given a suitable specification. The space of designs is then reduced to a final design by the use of constraints and preferences.

Victor et al. [1993] have developed a system using multiple expert systems with distinct roles to enable concurrent design in the field of powder ceramic components. The different expert systems have been implemented as 'agents' in the system using an object oriented expert system shell called CLIPS.

Huang and Brandon [1993] look at issues relating to co-operating expert systems in concurrent design and go on to describe the 'AGENTS' system. This is essentially a 'shell' which allows different expert-system based agents to co-operate. The system has been implemented using an object-oriented version of the Prolog programming language - POPLOG.

Extensive research has been done at the distributed AI lab into more general issues surrounding agent-based systems. These include negotiation (see Lander and Lesser [1993], coalition formation (see Sandholm and Lesser [1995]) and communication architectures (Bussmann and Muller [1993]).

Miles and Swift [1994] in collaboration with Lucas have looked at integrating DFM and design for quality within an engineering design environment.

Chawla and Sangal [1992] describe an intelligent design system, which aids in configuration design.

Finger at al [1992] describe a system called 'Design fusion' that 'surrounds the designer with experts and advisors that provide continuous feedback based on incremental feedback as the design evolves'. This system is based on an algorithm for recognising geometric features in an evolving design and features a novel algorithm for reasoning about constraints.

The 'Schemebuilder' project, Sharpe and Bracewell [1993], is an ambitious attempt to implement a system for generating conceptual designs for mechatronic products from a multi-functional perspective. This utilises an approach based on 'bond graphs' to assemble individual components into complete systems.

Kott et al [1990] describe the 'Function advisor'. This is a prototype tool to support concurrent engineering and incorporates a hierarchical model of the design being developed. The system utilises a constraint propagation technique to ensure consistency of decisions.

B.9 Conclusion

This Appendix has given an outline of different knowledge-based systems and specific implementations of such systems that have been utilised to support the design and concurrent design processes. However, in a large proportion of the reviewed literature, the main theme of the different research was in the technical means used to implement systems and not the model(s) of design on which the systems are based.

Appendix C: The *Common*KADS workbench

C.1 Introduction

Because of the complexity and difficulties involved in KBS development, different software tools have been developed to support this process. In particular, a number of workbenches exist which support *Common*KADS model development. Kingston et al [1995] outline four different tools which support the *Common*KADS methodology (to a greater or lesser degree) and assess their relative strengths and weaknesses. They summarise by noting that all the workbenches support the development of expertise models to a reasonable degree. However, the *Common*KADS workbench (currently supplied by Integral Solutions Ltd.) is the only tool that supports the development of a complete *Common*KADS models set. Since one of the objectives of my research is to develop template models to provide documented and readily available support to developers of KBS, this was the most appropriate tool to use. However Kingston et al [1995] also outline some of the limitations involved in using the tool. These included the relative speed of the workbench compared to some of the other systems available.

C.2 The *Common*KADS workbench

The *Common*KADS workbench was originally developed as part of the KADS 2 project and is intended to support the development and refinement of all six different model templates comprising the *Common*KADS model set. Development of the workbench has been influenced by the development of an earlier tool called Shelley (see Anjewierden et al [1990]) which focussed mainly on the instantiation of the expertise model within the original KADS model.

The workbench also supports some of the *Common*KADS project management functions. Because the project is effectively being run by a 'team' of one, the project management issues are not as relevant for my research as they would be for a typical software design project.

The workbench is currently implemented in SD Prolog and runs on the UNIX operating system (currently Sun's Solaris implementation of UNIX). For a more detailed description of the features and functionality of the workbench, see Toussaint et al [1994] and Kingston et al [1995].

C.3 The *Common*KADS model set

The workbench provides a number of editors, which allow development and refinement of the different model templates comprising the *Common*KADS model set. These are the organisation, task, communication, agent, expertise and design models. Figure C.1 shows the system interface provided for accessing the different models.

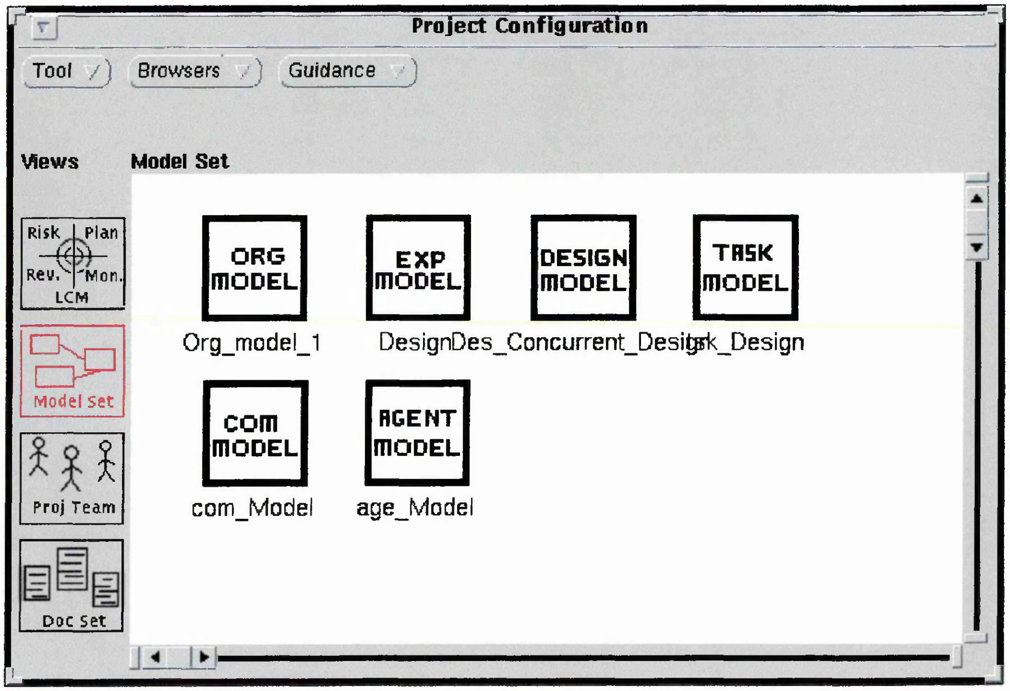
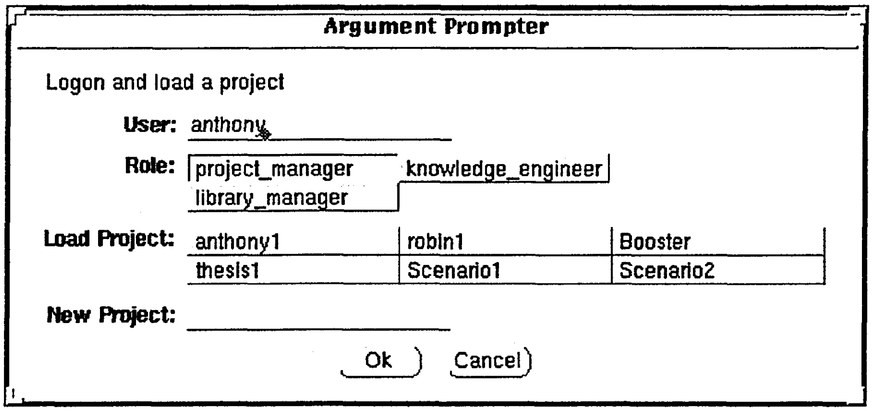


Figure C.1: The *Common*KADS workbench interface to the *Common*KADS model set.

C.4 *Common*KADS and project teams

The *Common*KADS workbench assumes the development of a complete knowledge-based system will be a team effort. As a result, the workbench provides support for the project management of system development. To this end, the workbench differentiates between three different types of user, the project manager, the knowledge engineer and the library developer. The toolbench provides a different interface and functionality depending on which type of user is using the tool. The user selects the correct user role when starting the workbench, as illustrated in Figure C.2. In addition, the workbench considers system development to be encapsulated within a particular 'project'.



Logon and load a project

User: anthony

Role: project_manager | knowledge_engineer | library_manager

Load Project: anthony1 | robin1 | Booster | thesis1 | Scenario1 | Scenario2

New Project:

Figure C.2: The *Common*KADS workbench and different user roles.

C.5 Help features

Different help and search features are provided to facilitate both the use of the tool and navigation around the different models as well as information on the methodological aspects of *Common*KADS. Figure C.3 shows the help and guidance tool being used to access information on the *Common*KADS expertise model.

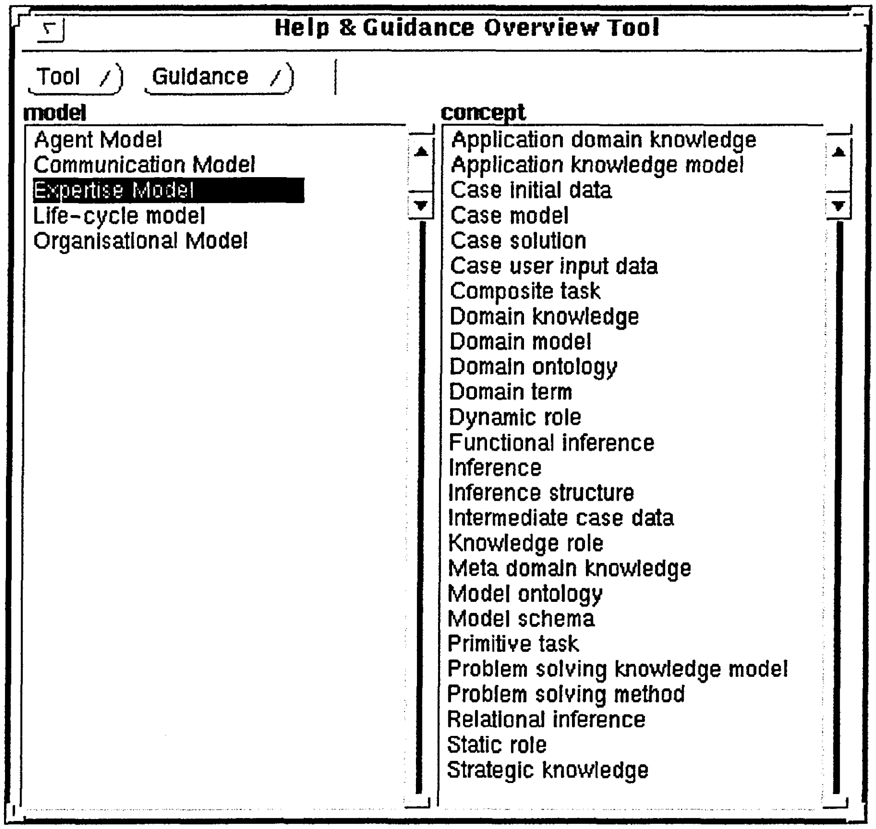
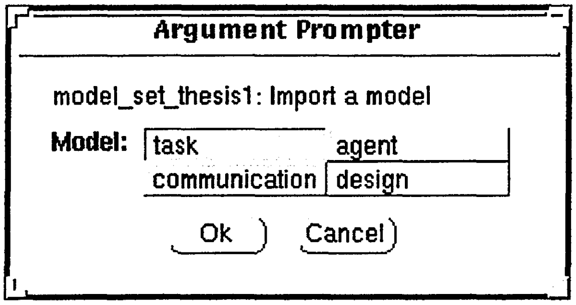


Figure C.3: The *Common*KADS workbench Help and guidance tool.

C.6 Links between different projects and models

A useful feature of the workbench is that different model templates can be imported from a previous project. This can be done for task, agent, communication and design models. As a result, model development does not have to be done from scratch. Figure C.4 shows the import tool being used to select a particular type of model for import.

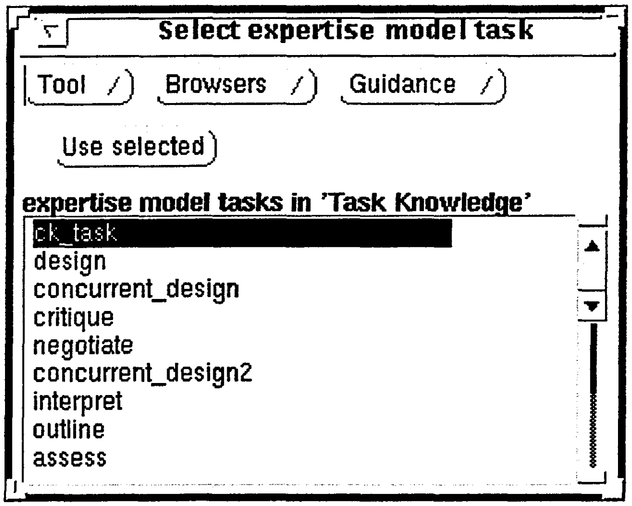


Argument Prompter

model_set_thesis1: Import a model

Figure C.4: The Import tool can be used to import model templates from previous projects.

Because of the many links between the different models comprising a complete project, the workbench provides facilities to link with entities in other models. For example, figure C.5 shows the linking tool being used to define a link between a task model and the corresponding expertise model entity relating to this task.



Select expertise model task

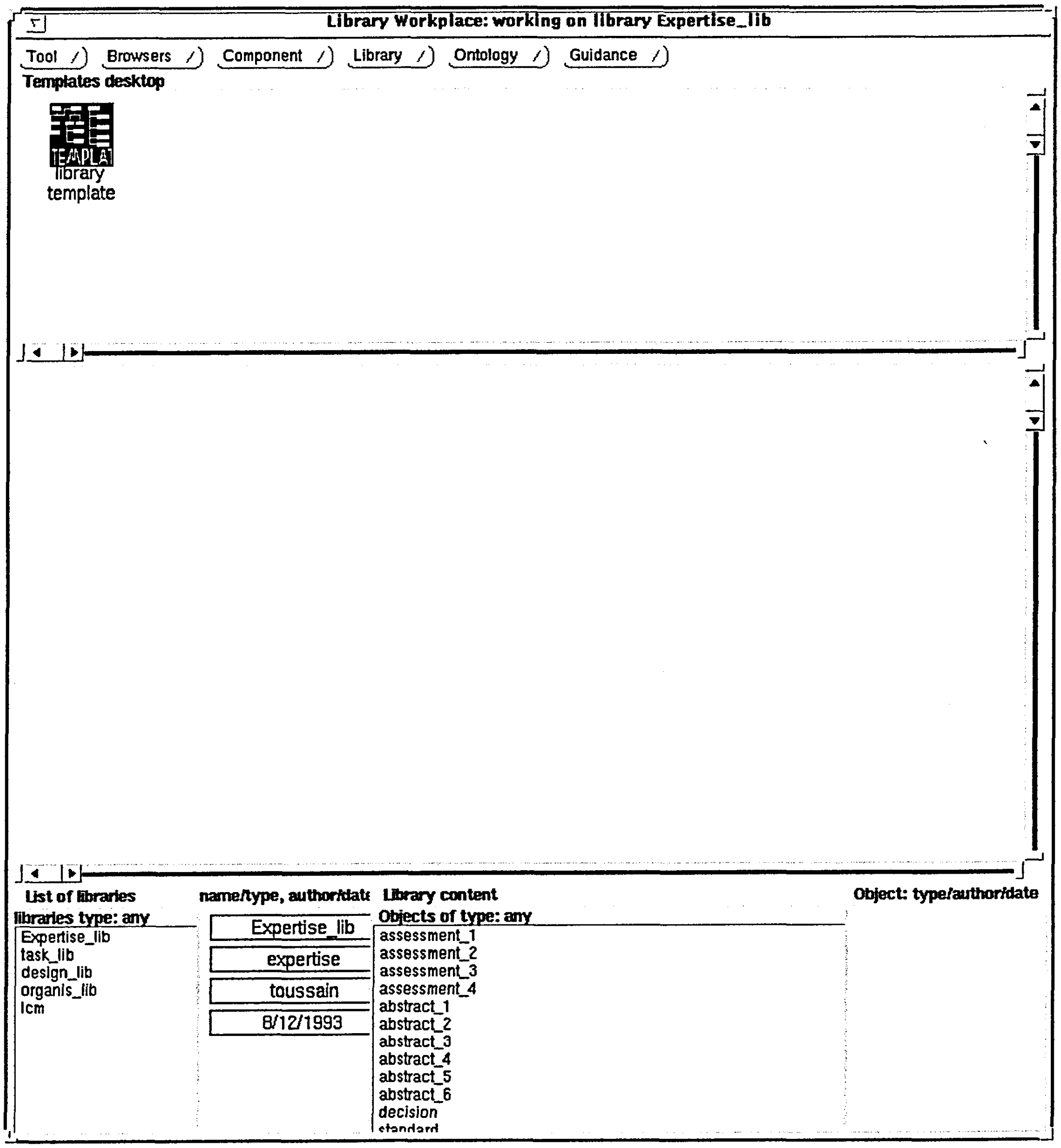
expertise model tasks in 'Task Knowledge'

Figure C.5: The linking tool can be used to link model entities within different models of the same project.

C.7 *Common*KADS Expertise model support

One of the cornerstones of the *Common*KADS methodology is that different *Common*KADS expertise task models have applications within a number of diverse fields.

I.e. the models are generic across different domains of application. As a result, the workbench also allows the import of previously defined expertise task models into a project. Unfortunately, the models available for import currently lags behind the different expertise model templates available (e.g. those defined in Breuker and Van de Welde [1994]). Current support within the workbench is mainly provided for models of the assessment task, as illustrated in Figure C.6.



List of libraries

libraries type: any

Expertise_lib
task_lib
design_lib
organis_lib
lcm

name/type, author/date

Expertise_lib
expertise
toussain
8/12/1993

Library content

Objects of type: any

assessment_1
assessment_2
assessment_3
assessment_4
abstract_1
abstract_2
abstract_3
abstract_4
abstract_5
abstract_6
decision
standard

Object: type/author/date

Figure C.6: The *Common*KADS workbench support for expertise modelling is currently mainly confined to assessment–type tasks.

Appendix D: Publications relevant to the research

Supporting Negotiation in Concurrent Design Teams

Robin Barker[1], Leigh Holloway[2] and Anthony Meehan[3]

[1] Sheffield Hallam University, Pond Street, Sheffield S1 1WB, UK.
[2] University of Sheffield, Environmental Business Network, Sheffield, UK;
[3] Sheffield Hallam University, Pond Street, Sheffield S1 1WB, UK.

Conference: Proceedings Sixth International Conference on CSCW in Design, July 12-14, 2001, London, Ontario, Canada, National Research Council of Canada, NRC Research Press, Ottawa, 2001, pg 243-248. ISBN 0-660-18493

Abstract:

This short paper reports on that part of a design support system, which assists each individual in a concurrent design team to negotiate with their colleagues during the materials selection stage of design. It identifies a difference in the way some concurrent design team members conduct negotiation compared to one of the standard economic models. It seeks to build upon this to suggest ways in which designers may be supported in identifying alternative materials, the choice of which would improve the design from their own perspective, and which are more likely to be acceptable to partners in negotiation.

Supporting Knowledge-based Processes Using Flexible Intelligent Agents

Robin Barker, Leigh Holloway, Jane Mardell, Anthony Meehan
Sheffield Hallam University, Pond Street, Sheffield S1 1WB, UK.
{R.Barker, L.Holloway, J.Mardell, A.Meehan}@shu.ac.uk

Conference: AAAI Spring Symposium 2000, Bringing Knowledge to Business Processes, Stanford University, Ca., USA. **Publishers:** AAAI Technical Report SS-00-03, AAAI Press, Menlo Park, Ca., USA. ISBN 1-57735-109-6

Abstract:

We are concerned to develop knowledge-based approaches to facilitating teams of people who must co-operate in the operation of business processes. We are particularly interested in processes characterised by high degrees of situation specificity (e.g. project-based processes such as product design) and by contexts in which the individual team members have comparable levels of authority/power and autonomy in respect of their roles in a business process. To the extent that we are interested in process design, we attempt to identify key information that needs to be made available at early stages of a project to avoid revision costs at later stages.

A knowledge-level model for concurrent design

Robin Barker, Anthony Meehan, Ian Tranter
Sheffield Hallam University, Pond Street, Sheffield S1 1WB, UK.
{R.Barker, A.Meehan, I.Tranter}@shu.ac.uk

Journal: International Journal of Applied Intelligence 10, pages 113-122 (1999) Kluwer Academic Publishers

Abstract:

This paper describes the development and validation of a knowledge-level model of concurrent design. Concurrent design is characterised by the extent to which multi-disciplinary perspectives influence all stages of the product design process. This design philosophy is being increasingly used in industry to reduce costs and improve product quality. We propose an essentially rational model for concurrent design and report on my validation of the model through studies with designers. I outline some of the limitations of current computational techniques needed to support negotiation in the design cycle and consider some of the implications of this for the development of systems to support concurrent design.

Supporting Concurrent Design Teams with Adjustably Autonomous Agents

Robin Barker, Anthony Meehan
Sheffield Hallam University, Pond Street, Sheffield S1 1WB, UK.
{R.Barker, A.Meehan}@shu.ac.uk

Conference: AAAI Spring Symposium 1999, Agents with Adjustable Autonomy, Stanford University, Ca., USA. **Publishers:** AAAI Technical Report. 1999. SS-99-06, AAAI Press, Menlo Park, Ca.; ISBN 1-57735-102-9

Abstract:

Successful deployment of knowledge-based systems in design is very limited. It has been suggested that one reason for this is that models of design have focused upon what machines can do and not on how designers actually design. The Cognitive Engineering Research Programme launched by the UK's Economic and Social Research Council in 1995 is seeking to achieve this goal by developing a better understanding of how we should design interactive systems through studying people and organisations. Oxman offers an approach to this through the development of an integrated understanding of design theory, cognitive science and computing. This paper describes work which attempts to address some of the issues above by studying concurrent engineering designers to identify (some of) the dynamics of the concurrent design process and examine ways in which these might be captured in computational approaches to support design activity.

Towards a Knowledge-level Model for Concurrent Design

Robin Barker, Anthony Meehan, Ian Tranter
Sheffield Hallam University, Pond Street, Sheffield S1 1WB, UK.
{R.Barker, A.Meehan, I.Tranter}@shu.ac.uk

Conference: 11th International conference on industrial and engineering applications of artificial intelligence and expert systems. Publishers: Springer Verlag 1998.

Abstract:

This paper describes the development and validation of a knowledge-level model of concurrent design. Concurrent design is characterised by the extent to which multidisciplinary perspectives influence all stages of the product design process. This design philosophy is being increasingly used in industry to reduce costs and improve product quality. We propose an essentially rational model for concurrent design using *Common*KADS and report on my validation of the model through studies with designers.

Critiquing and the Concurrent Design Process

Robin Barker, Chris Short, Ian Tranter, Anthony Meehan.
School Of Engineering, Sheffield Hallam University, Pond Street, Sheffield S1 **1WB.**
Brian Parkinson.
Manufacturing Systems Centre, University of Hertfordshire, College Lane, Hatfield. AL10 9AB.

Conference: Concurrent engineering and Design Automation (CEEDA96), Poole, 18-19 January 1996

Abstract:

One of the basic characteristics of concurrent engineering design is that different perspectives relating to the overall life-cycle of a product should be brought to the attention of a designer (or design team) as early as possible. Ideally a designer would be surrounded by 'experts' in different areas (e.g. manufacture, assembly, cost, materials etc.) who are able to 'critique' the evolving design from their own perspective. Unfortunately practical considerations in terms of cost, geographical distribution of expertise, communication problems etc. dictate that this ideal situation is unlikely to be realised in a typical working environment.

However by simulating the 'critiquing' behaviour of these experts in a computer-based design environment, some of these limitations can be overcome. This paper describes work that is ongoing at Sheffield Hallam University to investigate concurrent critiquing as a task and how the necessary expertise can be modelled and incorporated in such an environment.

Integrating knowledge-based systems into the concurrent design task

Robin Barker, Leigh Holloway, Ian Tranter, Anthony Meehan.
Sheffield Hallam University, Pond Street, Sheffield S1 1WB.

Conference: The First Annual Conference on Applied Concurrent Engineering (ACE '96) Seattle, Washington, USA, November 5 - 7 1996.

Abstract.

One of the basic characteristics of concurrent design (the design process in a concurrent engineering context) is that different perspectives relating to the overall life-cycle of a product should be brought to the attention of a designer (or design team) as early as possible. Ideally a designer would be surrounded by 'experts' in different areas (e.g. manufacture, assembly, cost, materials etc.) who are able to 'critique' the evolving design from their own perspective. Unfortunately practical considerations in terms of cost, geographical distribution of expertise, communication problems etc. dictate that this ideal situation is unlikely to be realised in a typical working environment.

By simulating the 'critiquing' behaviour of these experts in a computer-based design environment, some of these limitations can be overcome. This paper describes work that is ongoing at Sheffield Hallam University, using the *Common*KADS methodology to investigate concurrent critiquing as a task and how the necessary expertise can be modelled and incorporated in computer support for concurrent design.

A learning environment for concurrent engineering

Robin Barker, Leigh Holloway, Ian Tranter, and Chris Short.
Sheffield Hallam University.
Brian Parkinson.
Hertfordshire University.

Conference: The First Annual Conference on Applied Concurrent Engineering (ACE '96) Seattle, Washington, USA, November 5 - 7 1996.

Abstract:

This paper outlines the development of 'Design builder' - a computer-based system to teach design in a concurrent engineering context. This is part of the IDER (Innovative Design Engineering Research) project, which is a joint collaboration between Sheffield Hallam University and Hertfordshire University. The main objective of the project is to familiarise students with the concurrent design process by enabling them to design a product (from specification to final detail design) within a simulated concurrent engineering environment. Concurrent design can be considered as that element of concurrent engineering relating to the design process.

The *Common*KADS methodology has been used to model the processes within design and the expertise of different participants in the concurrent design process.

Extensive testing with students and design teachers has shown very positive reactions to the system and Version 1 of Design builder is now available to teachers of concurrent design.

Re-Shaping Design Teaching: A Strategy for Teaching and Learning

R.Barker, B.Parkinson and C.Short.

Conference: SEED95, University of Lancaster, 13-14 Sept 95. Published by SEED, 1995.

Abstract:

The traditional approach to the teaching of design has often mimicked the industrial 'over the wall' approach. However, industry is rapidly adopting the principles of Concurrent engineering and so it is vital that academia responds to the future requirements of industry and their own students. To teach the philosophy of concurrent engineering obviously requires a change in the curriculum content of design units but it also implies that the teaching and learning process should take place in an atmosphere of 'concurrency'. In the early years of a degree course most students (but not all) are relatively naive in their grasp of academic principles and industrial knowledge / experience and as a result are unaccustomed to working together in a concurrent way and find it extremely frustrating given their lack of knowledge. One solution to this problem is to provide students with a concurrent environment in the form of knowledge-based systems which provides information and data on a whole host of engineering subject areas and which also provides them with guidance on the application of the necessary principles and techniques. Such a computer-based system, employing artificial intelligence techniques, is being jointly developed by the University of Hertfordshire and Sheffield Hallam University as part of the Teaching and Learning Technology Programme-phase 2. This paper discusses the development of the system to date and it's application areas.

Modelling the Concurrent Engineering Environment Using Artificial Intelligence Techniques

Robin Barker, Chris Short, Ian Tranter & Anthony Meehan.
Sheffield Hallam University.
Brian Parkinson.
Hertfordshire University.

Conference: 10th International Conference on Artificial Intelligence in Engineering, International Centre For Mechanical Sciences, Udine, Italy, 4-6 July 95.

Abstract:

This paper outlines the development of a computer-based system to teach engineering design within a concurrent environment. This is part of the IDER (Innovative Design Engineering Research) project, which is a joint collaboration between Sheffield Hallam University and Hertfordshire University. The main objective of the project is to familiarise students with the concurrent design process by enabling them to design a product (from specification to final detail design) within a simulated concurrent engineering environment.

The environment is simulated using the *Common*KADS methodology to represent the expertise necessary to obtain different perspectives on the design process.

A prototype module to critique the specification development stage has been developed and is currently being assessed.

An expert teaching system for concurrent engineering

Robin Barker, Chris Short, Ian Tranter.
School Of Engineering, Sheffield Hallam University, Pond Street, Sheffield

Brian Parkinson, James Martindale.
Manufacturing Systems Centre, Hertfordshire University, College Lane, Hatfield.

Conference: 2nd International Conference on Design to Manufacture in Modern Industry. University of Maribor, Slovenia, 29-30 May 95

Abstract:

A need has been identified for engineering students and engineers currently practising in industry to learn the principles of and practices associated with concurrent engineering. One way it is believed this could be achieved is through the use of computer-based teaching systems. This paper outlines the requirements for such a computer-based teaching system and outlines work, which is progressing at Sheffield Hallam and Hertfordshire University to develop such a system.

Appendix E: Acronyms and Nomenclature

2-D	Two dimensional
3-D	Three dimensional
AI	Artificial Intelligence
BPR	Business Process Re-Engineering
CAD	Computer Aided Design
CAL	Computer Aided Learning
CBR	Case-based Reasoning
CE	Concurrent Engineering
CML	Conceptual Modeling Language
DFA	Design For Assembly
DFM	Design For Manufacture
DFX	Design For X
ERP	Enterprise Resource Planning
FEA	Finite Element Analysis
FMEA	Failure Means and Effects Analysis
KBS	Knowledge-based System
GA Drawing	General Assembly Drawing
MADS	Multi Agent Design Systems
NC	Numerically Controlled (usually in relation to machine tools)
PSM	Problem-solving method
UML	Unified Modelling Language
VDI	Verein Deutscher Ingenieure (German Institute Of Engineers)