Design Synthesis For Multi-X

- A 'Life-Cycle Consequence Knowledge' Approach

by

Jonathan C. Borg

Thesis submitted to the University of Strathclyde, Glasgow, for the degree of
Doctor of Philosophy

CAD Centre
Department of Design, Manufacture and Engineering Management
University of Strathclyde
Glasgow, Scotland, UK

11th June 1999

Acknowledgements

Anyone, who has gone through a Ph.D., is more than aware that such an achievement is not attained alone. I am thus very grateful to my Ph.D. supervisory team, composed over the years of Dr.Xiu-Tian Yan, Prof.Neal P.Juster, and Prof. Ken. J. MacCallum. They have all, in their very own individual way, provided me with professional supervision and research training at different stages of my registration period. Moreover, I would like to thank them for being patient with me in moments I was perhaps a bit impatient. My sincere thanks also go to Prof.Mogens.M.Andreasen, who following a recommendation by the CAD Centre Director, Dr.Alex H.B.Duffy, made it possible for me to pay research visits to the Technical University of Denmark, this allowing me to greatly benefit from his very own professional and technical expertise during lengthy discussions about my research work. Thanks thus also go to Dr.Alex H.B.Duffy for providing me with a helping hand in the difficult time following Prof.MacCallum's departure from the University of Strathclyde and also for the penetrating questions he raised during internal research seminars, that uniquely made me really 'think'. Thanks also go to the CAD Centre staff and research students for the many academic and non-academic discussions from which I tremendously learnt. A special thanks goes to Mr.Frank J.O'Donell, not only for raising valuable research discussions and being a true friend in time of need, but also for teaching me how to 'chill off' in an Irish way. An extended thanks goes to all the participants involved in the research evaluation exercise and to Mrs.Doreen Sehgal, Departmental Secretary, for providing valuable help and tips on living in Glasgow.

I would also like to thank the University of Malta for the financial support provided, allowing me to carry out Ph.D. research visits to the University of Strathclyde, the Technical University of Denmark and to participate in a number of international conferences/workshops. In particular, I would like to thank Eur.Ing.F.E.Farrugia, Head, Department of Manufacturing Engineering, for encouraging me to embark on this lengthy postgraduate training process and for making it possible to *'mix n match'* my lecturing and research commitments during my Ph.D.

From a different but equally important perspective, I would like to heartily thank, *both* my parents, who lovingly and with personal sacrifices, *initialized* and *continuously* supported my learning process, in their very own *unique* ways. Sincere thanks also go to my in-laws and relatives, for the routine support provided to my wife and daughter, during my long and frequent research periods away from home.

An *infinite* 'thanks' goes to my wife Jacqueline for her *tremendous* love, patience and morale support *throughout* this Ph.D., for carrying the major portion of family duties and for her encouragement in the difficult times. I also want to specifically thank her for being such a *wonderful* and *caring* mother to our daughter Iona, especially during the many times I was away from home.

Finally, I would like to thank God Almighty, Who guided me throughout this period of my life and Who blessed me with the *faith* [Mark, Chp. 11, 23-24] required to go through and complete this Ph.D.

Abstract

Product design decisions can result in unintended consequences that *propagate* across *multiple* life-phases such as manufacturing, use and disposal. If designers are to *generate* 'life-oriented' solutions, handling this phenomena is a necessity. Due to the sequence of life-phases, knowledge of such 'life-cycle consequences' (LCCs) is generated late, *after* decisions have been committed. Thus, designers have difficulties in foreseeing LCCs co-evolving with their solution. Further, a literature review established that, designers currently lack adequate support to foresee and explore LCCs during synthesis. To address this *'Design Synthesis for Multi-X ($D_sF\Sigma X$)'* problem, this thesis proposes, implements and evaluates a computational 'Knowledge of life-cycle Consequences (KC)' approach.

The establishment of a phenomena model disclosing *how* LCCs are generated from two different conditions has highlighted the necessity of concurrent 'artefact' and 'life-phase system' synthesis. This provided a foundation of how to model and timely utilize LCC knowledge for revealing LCCs co-evolving with a solution description. This resulted in a framework for the 'KC' approach consisting of: the *'LCC knowledge modelling frame'* which presents a formalism of 'what' elements to acquire and model for an application domain, together with how to structure the established relationships into 'LCC inference' and 'LCC action' knowledge; an *'artefact life modelling'* frame which provides a formalism for describing 'artefact life' compositional models that support the inference of LCCs; and the *'operational frame'* which discloses principles of how a LCC knowledge model can be utilized to amplify the human designer's capabilities. By identifying system requirements, an architecture and knowledge codification schemes, the framework was realized as a *Knowledge Intensive CAD* prototype, 'FORESEE', for the thermoplastic component domain.

An evaluation of FORESEE established that the 'KC' approach *integrates* synthesis with foreseeing *multiple* LCCs. This is fundamentally different from first generating a candidate solution and afterwards analysing the solution for conflicts with artefact life issues. The 'KC' approach thus provides a step towards realizing pro-active $D_sF\Sigma X$ support. However, further work is required to the framework and FORESEE to practically exploit its utilization.

Contents

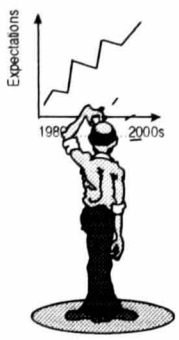
List of Figures

List of Tables

chapter 1

1.0 Introduction

Rising expectations



Besides providing the market with products that fulfill their function [Roozenburg et al. 1995] to rising customer expectations [Farish 1992], manufacturing firms are increasingly expected to deliver *'life-oriented'* products that incorporate various total life-cycle values [Ishii 1995]. For instance, artefacts are expected to be easy to produce [Lindbeck 1995], assemble [Boothroyd et al. 1991], test [Lawlor-Wright et al. 1995], service [Eubanks et al. 1993] and cater for end-of-life issues by facilitating automated dis-assembly [Boks et al. 1997] and recycling [Seliger et al. 1994]. In achieving all this, manufacturing firms are also expected to keep costs low, deliver products on time and within a quality context [Clausing 1994; Dooley 1994].

1.1 Product Development Reality

Developing products that cater for total life issues

Coping with this complex product development reality has gradually changed the focus of industry to the product *design process* [Duffy et al. 1993; Blessing 1994]. Of relevance is that as argued by Berliner & Brimson [Berliner et al. 1988], investments made to generate good design concepts give higher returns than equivalent efforts made in other areas such as manufacturing methods. Thus, in today's industrial climate, *"design is a tool for competition"* [Stenros 1997]. However, whilst designers are being expected to consider *a host of total life* issues during design, there is enormous pressure to reduce the overall *time* and *cost* of product development [Rogers 1997; Swift et al. 1997]. As a result, the effort necessary to create total life oriented design solutions is resulting in additional complexity [Andreasen et al. 1997] during the product development task. This research is concerned with this role of the design process and how designers can be supported in the context of generating mechanical artefact conceptual solutions, that cater for a host of total life issues. To introduce the design problem motivating this research, this section proceeds with an outline of the realities involved in developing 'life-oriented' design solutions.

1.1.1 Concurrent Engineering Trends

Various definitions

To deliver products that cater for a host of total life issues, industry increasingly raised an interest in *concurrent engineering* [Cleetus 1992; Jo et al. 1993; Parsaei et al. 1993], sometimes termed *simultaneous engineering* [Belson 1994; Ranky 1994; Molina et al. 1995; Wallace 1997]. Unfortunately, the term concurrent engineering has been used to mean a number of different things, both to practitioners and researchers, all this baffling potential users of such an approach. As a matter of fact, one can encounter several definitions having varying views. According to a practitioner [Dan 1992], concurrent engineering (C.E.) is focused on a team based, design approach. Others view it as related to a cultural change consisting of redefining organizational structures and breaking down barriers between disciplines that have previously worked in isolation from each other [Albano et al. 1994]. An alternative view focuses on 'what' is being decided or learned, as opposed to 'who' is deciding or 'how' it is being done [Douglas et al. 1993], the argument being that the key issue in C.E. is the accumulation of knowledge. Winner's [Winner 1988] definition:

Winner's definition

> "Concurrent Engineering is a systematic approach to the integrated, concurrent design of products and their processes, including manufacture and support. This approach is intended to cause the developers, from the outset, to consider all elements of the product life-cycle from concept through disposal, including quality, cost, schedule and user requirements."

primarily focuses on the need of concurrently designing both product and process solutions and emphasizes the need of considering *all* product life issues, early in the design process.

DICE's definition

The *US DARPA Initiative in Concurrent Engineering (DICE)* program provides a definition [Cleetus 1992] that considers 'what' is actually involved :

> "CE is a systematic approach to integrated product development that emphasizes response to customer expectations and embodies team values of cooperation, trust and sharing in such a manner that decision making proceeds with large intervals of parallel working by all life-cycle perspectives early in the process, synchronized by comparatively brief exchanges to produce consensus."

This highlights the role of team members taking a different *perspective* towards the artefact design solution. Also, it explicitly points out the role group *decision making* has in such an approach to bear on every issue being decided upon. In spite of the varying views taken by these definitions and many others found in [Prasad 1996a], they essentially reflect what this thesis considers as being two key elements of a C.E. approach, these schematically illustrated in Figure 1.1:

Concurrency elements

- the synergetic consideration of both functional and life-cycle issues, referred to in this research as *concurrent consideration (CC)*. This concurrency element is viewed as being related to the 'life-oriented' *knowledge* ('knowledge' being inclusive of data, information and experience) available and employed during the execution of a design activity;

- the *concurrent execution (CX)* of independent or semi-independent product development activities such as the design of a thermoplastic component and the design of its mould tool. This is viewed in this thesis as related to the time dimension between product development activities.

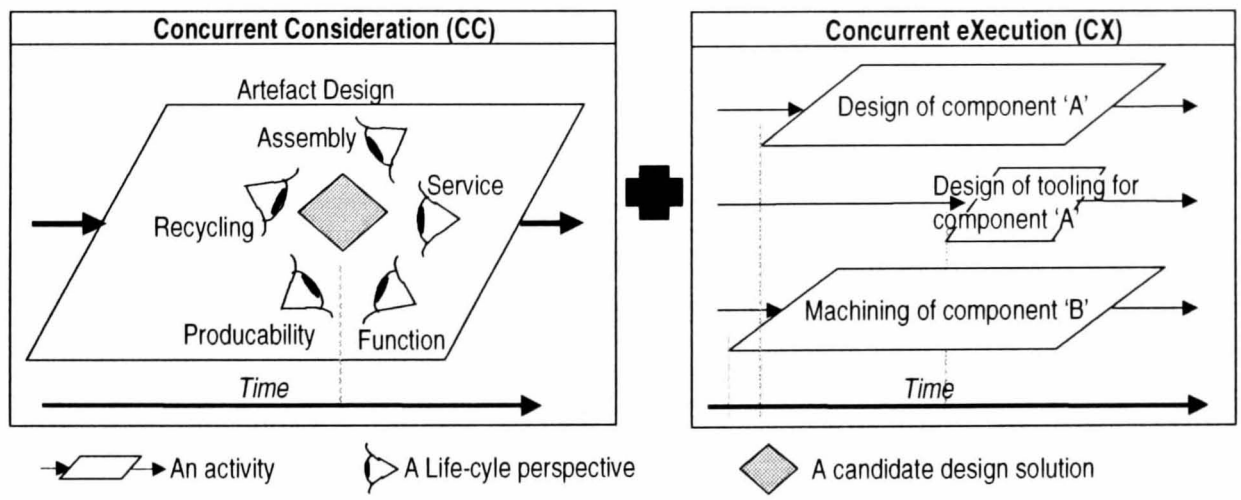


Figure 1.1 - Concurrency elements

These concurrency elements are reflected in various research efforts. Some relate to concurrent consideration (CC), examples being [Ishii 1991; MacCallum 1992; Olesen 1992; Molloy et al. 1993; Morup 1993; Chal et al. 1997; Tichem 1997]. Others focus upon enhancing the collaboration between various product development actors so as to share and communicate information [Karinthi et al. 1992; Cleetus 1993; Park et al. 1994; Kiriyama et al. 1996; Maher et al. 1997; Prasad et al. 1997] thus permitting the concurrent execution (CX) of activities in a virtual enterprise [Gadient et al. 1997; Williams et al. 1998].

Providence

Collectively, these elements reflect that C.E. requires more than the concurrent execution (CX) of product development activities or, simply an approach that focuses on considering single artefact life-phase issues, like with for instance *Design For Manufacture (DFM)* [Boothroyd et al. 1991]. Rather as defined by Winner [Winner 1988], a C.E. approach is also concerned with the consideration of *all* product life-cycle issues as from the *outset* of the design process. Olesen [Olesen 1992] thus argues that besides

simultaneity i.e. the execution of interrelated tasks at the same time and the *integration* of relevant functional areas during the development process, an important element of concurrence is *providence*, i.e. foreseeing and taking into account aspects of the total life that are fixed or determined during design.

1.1.2 Shift Towards 'Design Synthesis For Multi-X'

Life-oriented design synthesis

Due to the CC element, designers are being prescribed to adopt *'life-oriented' design approaches* [Olesen 1995; Chal et al. 1997]. Such a prescription reflects the significance of an approach promoting the *'generation'* of a solution catering for a *host* of artefact life issues, termed in this research as *'Design Synthesis for Multi-X' ($D_sF\Sigma X$).*

Not only 'function'

'Function' was traditionally the prime consideration of designers in many types of industry [Duffy et al. 1993]. Focusing on 'function' results in products that can be time consuming and costly to produce as they can involve a large number of parts and assembly operations [Lindbeck 1995]. Olesen [Olesen 1995] argues that treating solution concepts from a functional point of view is *only one* aspect of one *meeting* [Mortensen et al. 1996] taking place during the total life of a product i.e. the meeting between the product and the user.

'X-ability' values

Focusing on function therefore leads to products that are "not being designed for..." [Duffy et al. 1993] a number of 'X-abilities', where as stated by Prasad [Prasad 1996], X-abilities reflect a number of total life values:

> "X-ability ⇔ {manufactur*ability*, maintain*ability*, assembl*ability*, service*ability*,...}"

Multiple 'X'

Designing and delivering products that cater for a *host* of *'X-ability'* values is thus the foundation of the competition game [Prasad 1996]. Hence, as reported in [Duffy et al. 1993], the early eighties emphasis in industry on shop floor automation, shifted in the nineties to the *design process*, dominated by *"Design-For-Almost-Everything."* Similarly, Brown [Brown 1996] reports *"the number of 'ilities' are growing."* Further, Farbricius [Fabricius 1994] argues that a design solution must be found by using a *wide angle* focus by considering all the *performance measures* termed universal virtues [Olesen 1992] - cost, lead time, quality, flexibility, risk, efficiency and environmental effects. As a result of this shift, there is an increasing interest in total life-cycle issues, as evident from examples of academic research thrusts [Bowen et al. 1990; Alting et al. 1995; Spath et al. 1996; Vajna et al. 1997; Wallace 1997] and industrial practice [Franze 1997; Timperi 1997].

1.1.3 Decision Intensive Design Process

Solution space of alternatives makes design decision intensive

The alternatives encountered in the solution space make the design process, decision intensive [Joshi 1991; Olesen 1992; Starvey 1992; Mistree et al. 1993; Jeang et al. 1995; Willemse et al. 1995; Medland 1997]. Typical decisions concern, for instance, the selection of appropriate function means, type of materials to employ, configuration layout, and the connection technology to be employed between different parts. Clausing [Clausing 1994] even attempts to quantify the large number of decisions involved for large, complex products. The reality therefore is that as from the early design stages, decisions are being made on many aspects of the artefact being designed, as evident by the survey found in [Duckworth et al. 1998].

1.1.4 A Phenomena: Decisions Result in Life-Cycle Consequences

Dispositions

Decisions which seem good for one life cycle requirement can lead to problems with other requirements. Thus, design decisions are associated with consequences [Andreasen et al. 1990; Duffy et al. 1993; Dym 1994; Swift et al. 1997; Tichem 1997] that can be intended/unintended and good/problematic [Borg et al. 1998]. Hubka & Eder [Hubka et al. 1988] argue that every design decision, including *early* design decisions, has an influence on the following life-phases. This natural phenomena of consequences is termed the *concept of dispositions* [Olesen 1992], where a *disposition* means that part of a decision taken within one functional area (i.e. product design) which effects the type, content, efficiency and progress of activities within other functional areas (e.g. assembly). Design decisions thus influence the performance of other life-cycle phases in terms of measures such as cost and time. For example, a decision to use four screws for the assembly of a product, influences a host of product development areas – the design department, the purchasing department, stores, the production department, the quality control department and the sales department [Andreasen et al. 1987]. Thus, as disclosed in Chapter 2, design phase decisions can have consequences that *propagate* across *multiple* artefact life phases. The reality therefore is that design decision making is integrated with the phenomena of such *life-cycle consequences (LCCs)*, irrespective of whether a consequence is *intended* or *unintended* from a designer's viewpoint.

1.1.5 Limited Designer's Knowledge Of Life-Cycle Consequences

Lack of knowledge

Though Tomiyama [Tomiyama 1996] highlights the need of employing product life knowledge as from the early design stages, Olesen [Olesen 1995] argues

that designers have a lack of life-cycle phase knowledge. As argued in this thesis, due to the traditional formal training received and their personal experience of artefact life issues, designers do not generally possess a *breadth* and *depth* [MacCallum et al. 1987] of knowledge related to the different product life-cycle phases. As a result, designers lack knowledge about a host of LCCs resulting from design decisions. Thus, designers frequently engage in decision-making under ignorance of LCCs arising from their decisions.

Influences on design decision making

Technical information available during design is known to play a significant role in the quality of the design solution [Hubka 1985]. Therefore, a lack of appropriate LCC knowledge, can influence the quality of design decision making. For instance as argued in [Duffy et al. 1995] :

> "when carried out by individuals, decision making can be subject to individuals' limited knowledge, experience etc. Consequently such decision making can be based upon limited insight into the problem at hand and thus result in low quality decision making."

Need to amplify designer's LCC knowledge

Foreseeing consequences resulting in the different life phases has long been acknowledged as an important designer ability. For example Hubka & Eder [Hubka et al. 1988] state that :

> "This design process includes human beings and their social context (in the widest sense), and consists of partial processes such as planning, predicting a market, financing, designing, evaluating, etc., and anticipating the needs and problems of developing, manufacturing, assembling, assuring quality, testing, marketing, repairing, maintaining, using, disposing, impacting the environment etc."

However, how these consequences can be foreseen and appropriately handled during early design is still a bottleneck, this evident from industrial examples encountered in this research and therefore a worthwhile research thrust. For example deciding on tolerances is one of the important tasks of the designer, but [Meerkamm 1997]

> "Very often he [the designer] cannot foresee the consequences of his choice in accordance to function, production process, inspection and cost."

Consequences impinging on artefact life phases frequently influence performance measures. For example, a mould tool maker [Smith 1997] argues that:

> "There are often times when simplifying the basis of the [artefact] design can save time in the creation of the tool and mouldings, drastically reducing costs."

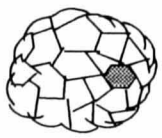
Thus, the lack of relevant knowledge in the early design stages is a major cause of many downstream life-cycle problems [Salzberg et al. 1990]. These circumstances reflect that for the concurrent consideration element of C.E., there is great scope of amplifying the designers' knowledge *during* design decision making, with 'knowledge of LCCs' to help them *generate* early, life-oriented design solutions.

1.1.6 Inadequate Means Supporting 'Life-Oriented Design'

Inadequate providence support

The literature review in Chapter 4 reveals that designers lack adequate support during early design allowing them to *foresee* unintended consequences influencing *multiple* life phases. Awareness provided is *narrow* and *segmented* (single 'X'). Further, knowledge of LCCs is provided *late*, mostly during solution analysis rather than synthesis, thus not supporting 'artefact life solution' exploration. That is, designers currently have inadequate *means* to handle the phenomena of LCCs solution synthesis.

1.1.7 Design Problem Outline



Reality

In summary, the reality with which this research is therefore concerned, is that:

- designers are being expected to consider *more* issues concerning the *total* artefact life when generating design solutions. This transition to $D_sF\Sigma X$ therefore increases the demands being made of designers;
- design decisions have a consequence, good or bad. Moreover, decisions made can *propagate* consequences across multiple artefact life phases;
- knowledge of such LCCs is thus distributed amongst various artefact life actors. At the same time, practice shows that life cycle actors are too busy to continuously and effectively form part of a design team;
- due to a lack of 'life cycle consequence' knowledge, designers are *frequently unaware* of the many unintended life cycle consequences being generated when they commit design decisions.
- current means are deficient in addressing this problem as they allow design decision making to take a *narrow* focus. Further, awareness of such LCCs, if any, is revealed too *late*, thus making 'life-oriented' solution generation and exploration difficult to achieve.

Problem summary

Therefore, handling the phenomena of propagating *life-cycle consequences* as from early design is a necessity if designers are to generate solutions satisfying a host of total life issues.

1.2 Research Project Description

Overall motivation

Based on the design problem reality outlined in section 1.1, the overall motivation behind this research is to pro-actively support mechanical artefact designers in generating early, 'life-oriented' design solutions. To conduct Ph.D. research [Phillips et al. 1996] in a planned way, the research methodology employed in this Ph.D. (recently published in [Duffy et al. 1998]) is based on a mixture of theoretical foundations and experimental activities. This is discussed next and outlined in Figure 1.2.

1.2.1 Research Methodology

Case-studies encountered with product design and development practice in industry (1a) together with a literature search (1b) provided the foundation for characterizing the design problem (2) with which this thesis is concerned. A basic hypothesis (3) for a solution to the identified problem was generated, this giving rise to a research problem (4). On tackling the research problem, a solution (5) was arrived at, accompanied by the co-evolution of research sub-problems. To test the effectiveness of the solution, the research project embarked on experimentation.

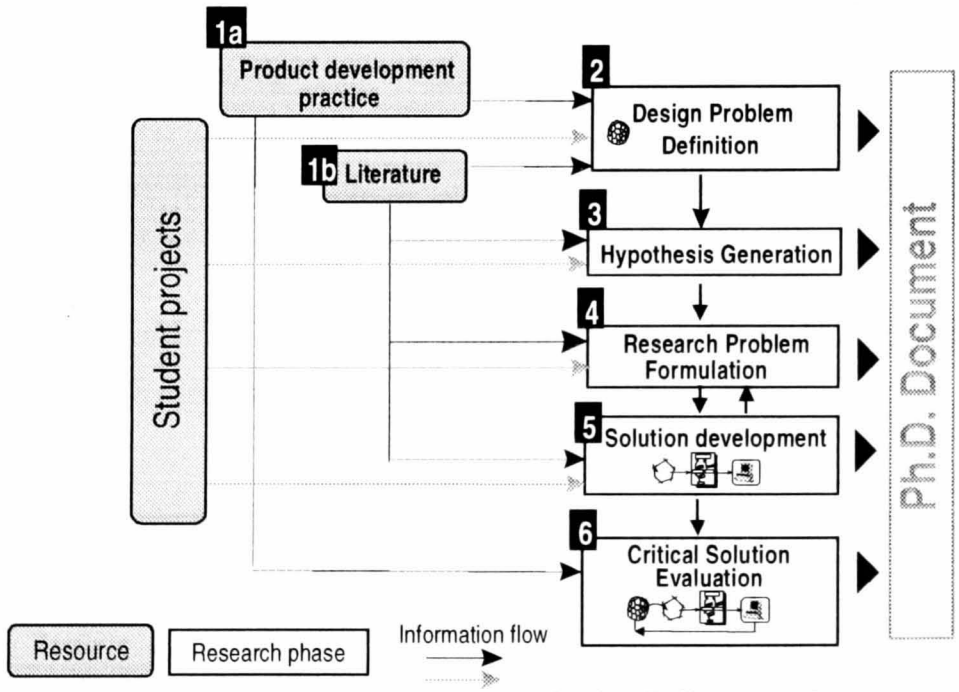


Figure 1.2 - Research methodology - modified from [Duffy et al. 1998]

This provided a basis for critical evaluation (6) of the result, enabling the identification of both strengths and weaknesses of the solution and thus the identification of future research directions. During this project, information extracted from a number of student projects [Borg 1995; Portelli 1995; Grech 1996; Bonello 1997; Cutajar 1997; Galea 1997] supervised by the author was also utilized as an additional input to different phases in the methodology

(Figure 1.2). Finally, the elements in the methodology collectively contributed to the generation of this Ph.D. document.

Solution Hypothesis

This research argues that for designers to generate life-oriented design solutions, they need to foresee during synthesis, unintended LCCs co-evolving with their early solution description. The argument being made here is that for *'concurrent consideration'*, the decision making process will be better accomplished by reasoning with knowledge providing an insight into the associated life-cycle consequences. It is natural to expect that through the explicit awareness of possible LCCs arising from their design decisions, designers would at least be motivated to consider modifying/rejecting design decisions which are likely to generate unintended LCCs. This may mean that more design decision alternatives have to be considered but the hypothesis is that the quality of the committed decisions will be better for generating life-oriented design solutions. Therefore, the hypothesis is that explicitly and systematically providing designers with knowledge of the LCCs co-evolving with a design solution, will:

Expectations

- assist designers in becoming aware of a host of solution specific *unintended* LCCs influencing *multiple* artefact life-phases;
- motivate and support designers in exploring, during synthesis, the co-evolving artefact life solution and problem space;
- guide designers to make decision commitments more consciously with respect to artefact life issues.

High Level Research Problem

Basic research question

The high-level research problem therefore concerns:

> *How can designers be explicitly provided with knowledge of LCCs co-evolving with their design solution description?*

Theoretical foundation

The variety, complexity and interaction [Meerkamm 1994] of artefact life issues that have to be concurrently considered during design decision making, make the *manual* exploration of a design problem and solution space difficult. Given the human being's mental storage capacity and processing limitations [Ellis et al. 1989], Computer Aided Design (CAD) support, acting as a

'knowledge amplifier' [MacCallum 1992a] is seen as a suitable solution to how designers can be provided with knowledge of LCCs co-evolving with their solution description. Relevant research by Olesen [Olesen 1992] resulted in a design approach based on *provident thinking* so as to optimize the conditions during production and the product's life. Olesen provides a theoretical foundation for providence through the *theory of dispositions.* This initial theoretical foundation can therefore be exploited to develop a suitable computational $D_sF\Sigma X$ means to support designers in the generation and exploration of 'life-oriented' design solutions.

Computational Means Development Framework

A foundation upon which to develop computational design support systems is the framework (Figure 1.3) found in [Duffy & Andreasen 1995]. It is argued that for a computational means to make an impact on the design *reality* it is trying to support, it should be based on *knowledge models* derived from descriptive models of the reality, termed *phenomena models.* The specific phenomena concerned with in this research is that design decisions give rise to life cycle consequences, irrespective of whether the designer is aware or not. A *phenomena model* aims to explain through observations made, *how* the phenomena is caused in *reality*.

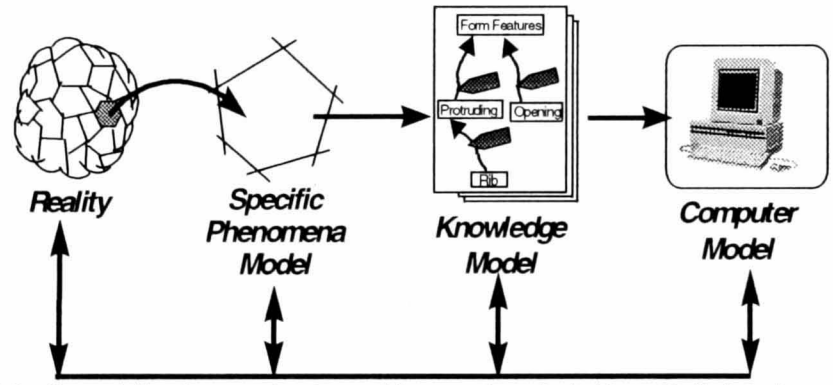


Figure 1.3 – Computational means development framework – adopted from [Duffy & Andreasen 1995]

The *knowledge model* is concerned with *what* elements should form part of the knowledge structure, and with *how* these elements should be *related* to each other and *organized* in order to result in codified [Tomiyama et al. 1995] knowledge. The *computer model* encompasses existing or new techniques to realize and manipulate the knowledge model for supporting the design reality. Thus key tasks in developing the required computational $D_sF\Sigma X$ means are LCC *phenomena* and *knowledge modelling.*

1.2.2 Aim and Research Objectives

Research aim

Based on the foregoing discussions, the research aim therefore is to:

> *Develop and evaluate a computational approach to $D_sF\Sigma X$ as a means to pro-actively guide designers in generating 'life-oriented' solutions, by supporting them during synthesis, in foreseeing and exploring, unintended life-cycle consequences co-evolving with their solution description and propagating across multiple life-phases.*

Objectives

Based on the overall research methodology and the computational means development framework, this aim will be met through the following objectives:

(a) to characterize the research problem, by:

- establishing what the design problem is, by disclosing the phenomena of propagation effects associated with a mechanical artefact's life;
- identifying the requirements for a 'life-oriented design' approach that can handle the phenomena of propagation effects;
- identifying strengths and weaknesses of existing means supporting 'life-oriented' design from the perspective of providence;

(b) to develop a computational approach to $D_sF\Sigma X$, by:

- establishing a phenomena model that provides a formal explanation of *how* life-cycle consequences are generated;
- exploiting the LCC phenomena model explanations to establish an *approach framework* describing how LCC knowledge can be modelled and explicitly utilized during synthesis to pro-actively support $D_sF\Sigma X$;
- establishing a formal LCC knowledge model for realizing the framework;
- identifying CAD system requirements and architecture for realizing the framework in a computational form;

(c) to evaluate the developed approach to $D_sF\Sigma X$ by:

- implementing the framework as a prototype CAD tool;
- applying the CAD prototype to a design scenario;
- analysing the CAD prototype evaluation results to establish the strengths and limitations of the novel approach to $D_sF\Sigma X$, this leading to the identification of future research avenues.

1.3 Thesis Structure

Founding chapter

This chapter laid the foundations for this thesis by introducing the basic problems facing designers as a motivation to the research aim and objectives presented. Building on this chapter, the rest of this thesis is split up into three parts 'A', 'B' and 'C', as illustrated in Figure 1.4.

Part 'A'	***Characterizing The Research Problem*** Chapter 2 - Mechanical Artefact Life Phenomena Characterization Chapter 3 - Characterizing 'Design Synthesis For Multi-X' Chapter 4 - A Review of Means Supporting 'Providence' Chapter 5 - Established Research Problem
Part 'B'	***Development of a 'KC' Approach to $D_sF\Sigma X$*** Chapter 6 - Life-Cycle Consequences Phenomena Model Chapter 7 - A 'Knowledge of LCCs' Approach Framework to $D_sF\Sigma X$ Chapter 8 - Formalism of A LCC Knowledge Model Chapter 9 - FORESEE - A KICAD Prototype Implementation
Part 'C'	***Evaluation, Discussion and Conclusions*** Chapter 10 - Evaluation of FORESEE To Supporting $D_sF\Sigma X$ Chapter 11 - Discussion Chapter 12 - Conclusions

Figure 1.4 - Thesis structure

As a foundation to *what* the design problem is, Chapter 2 discloses the phenomena of propagation effects associated with a mechanical artefact's life and its implications on design. Chapter 3 focuses on the transformation taking place during the design phase so as to identifying characteristics required in a 'life-oriented design' approach, for handling the phenomena of propagation effects. Chapter 4 presents a critical review of different means by which 'life-oriented design' is supported from the perspective of providence. Chapter 5 closes Part 'A' by presenting the established research problem and the Ph.D.' s research boundary. Part 'B' presents the development of the 'KC' approach to $D_sF\Sigma X$. Chapter 6 presents a LCC phenomena model that formally describes *how* LCCs are generated. Exploiting this understanding, the concept of a *'**K**nowledge of life-cycle **C**onsequences' (KC)* approach framework to $D_sF\Sigma X$ is presented in Chapter 7. Chapter 8 presents a formalization of a LCC knowledge model for supporting $D_sF\Sigma X$ at the component level. It establishes *what* artefact life elements need to be modeled and *how* they should be related to be transformed into meaningful and computationally feasible knowledge. Chapter 9 identifies the system requirements and architecture of a *Knowledge Intensive CAD (KICAD)* tool realizing the 'KC' approach framework. It presents implementation issues for a prototype, 'FORESEE', realized for evaluation purposes. Part 'C' commences with Chapter 10 that presents an evaluation of FORESEE to supporting thermoplastic component $D_sF\Sigma X$. Chapter 11 discloses the original contributions made, discusses their implications and validity, and proposes avenues for future research to develop the results further. Finally, Chapter 12 closes this thesis with conclusions resulting from the work presented.

part 1

Characterizing the Research Problem

chapter 2

2.0 Mechanical Artefact Life Phenomena Characterization

Chapter Scope

The aim of this chapter is to disclose the phenomena of propagation effects associated with a mechanical artefact's life, this providing the foundation to what the design problem is that this research work has been embarked upon. For this purpose, section 2.1 provides a theoretical background to what comprises a mechanical artefact. Section 2.2 then presents a decomposable total artefact life model to characterize what is involved in a mechanical artefact life. A number of observations based on the established artefact life model and examples are then presented in section 2.3 in order to reveal the propagation effect phenomena that occurs during an artefact's life due to design phase decisions. Chapter conclusions highlighting design phase implications as a result of this phenomena are made in section 2.4.

2.1 Mechanical Artefacts

System view

There are many artefact domains [Mills 1993] such as mechanical, electrical and electronic. This research focuses on industrially manufactured, mechanical artefacts in which, unlike with craft-based artefacts, the activity of *designing* is separate from the activity of *making* [Cross 1994]. Using the 'Theory of Technical Systems' [Hubka et al. 1988], this research considers a mechanical artefact as a *system* that can be decomposed into systems of a finer resolution [van den Kroonenberg 1987; Roozenburg et al. 1995]. As Hubka & Eder [Hubka et al. 1988] explain, a system is a finite set of *elements* collected to form a whole under certain well-defined rules, whereby certain definite *relationships* exist between the elements and its environment.

Multiple structural viewpoints

Treating a mechanical artefact as a decomposable system allows it to be studied at different levels of complexity [Hubka et al. 1988]. Many structural solutions are superimposed in an artefact [Andreasen et al. 1996]. Thus, an artefact structure can be viewed from different perspectives: *synthesis oriented view*, *functional view*, *product life view* and *product assortment view* (Figure 2.1).

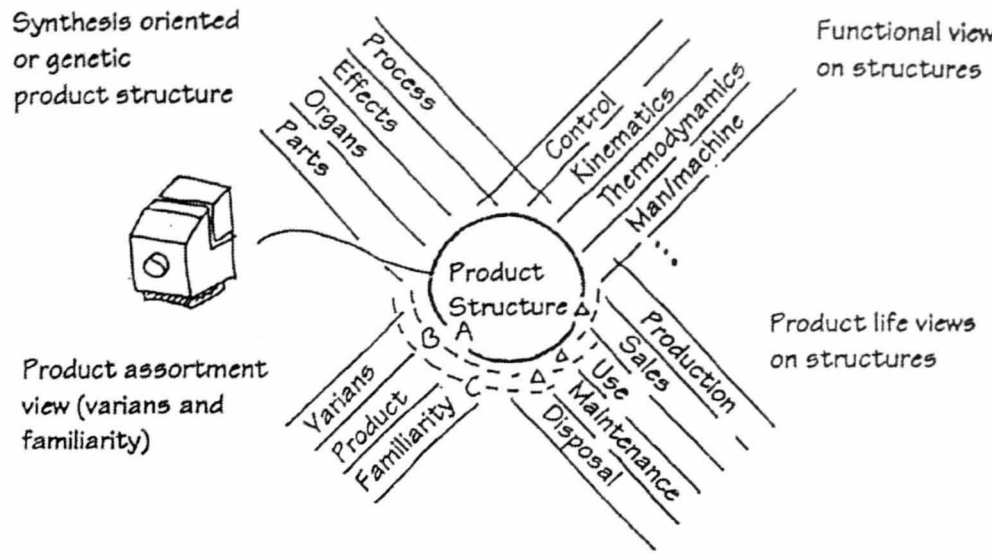


Figure 2.1 – Multiple product structure viewpoints - [Andreasen et al. 1996]

Product breakdown structure

Observing an artefact system from a constructional (parts) synthesis viewpoint gives rise to a *product breakdown structure (PBS)*, typically termed a compositional model [Kerr 1993; Tichem 1997] (Figure 2.2).

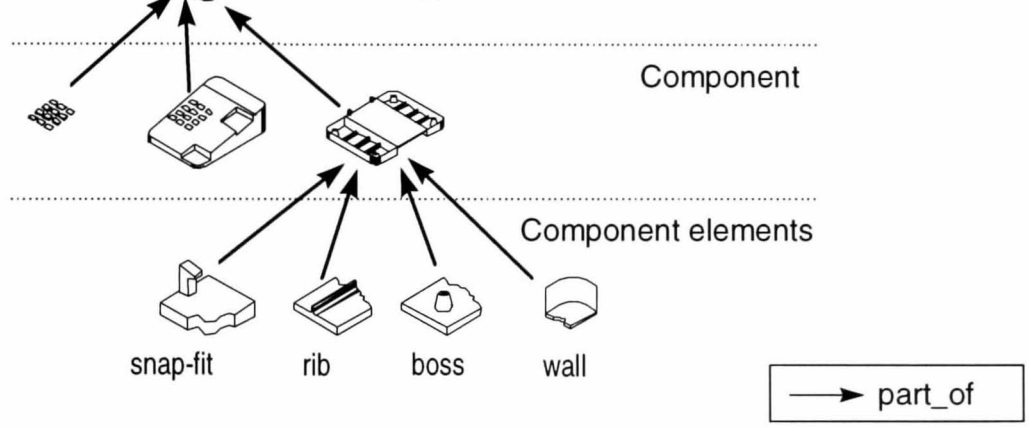


Figure 2.2 – Product design elements present in an artefact structure

Product design elements

In this structural view, an artefact system consists of a number of elements, termed in this research *product design elements (PDE),* related to each other with *'part of'* [Winston et al. 1987] relationships. In correlation with Blessing [Blessing 1994], a PDE can be :

- *a sub-assembly*: i.e. an element composed of a set of other PDEs. An example is a telephone's electronic circuitry enclosure, consisting of PDEs such as the numeric buttons and thermoplastic cover (Figure 2.2);

- *a component*: i.e. a single material element produced without any assembly operations; e.g. telephone's bottom plastic enclosure;

- *component elements:* these are elements that constitute a component e.g. for the bottom plastic enclosure, component elements include form features, material, snap-fits and rib features.

Of relevance to this research is that from a synthesis oriented viewpoint, only the *constructional* artefact description will be complete [Andreasen 1991a] at the end of the design phase and hence communicated to the other life-phases. During the design phase, the designer is thus gradually making definitions to the PBS with different PDEs, that will than be communicated to the next life-phases.

2.2 A Mechanical Artefact Life Model

It is difficult to characterize the life of different mechanical artefacts in terms of a single model. For this reason, this section presents a number of models as a foundation to the decomposable artefact life model used in this research.

Life

There are many terms related to 'artefact life' such as *'useful life'*, *'launch-to-finish'* time [Prasad 1996] and *commercial life cycle* [Willemse 1997]. For this research, the *artefact life* is the total elapsed time it takes from when the *need* for an artefact is established to when the artefact is removed from existence.

Phases

An artefact life is composed of a number of *phases* during which there is a transformation of an *operand* (information, material or energy) [Hubka et al. 1988] from an initial state, to another state. A *phase* is a *time segment* in the artefact's life. For instance, Hubka & Eder [Hubka et al. 1988] state that for technical systems the life-span can be divided into four phases*: origination, distribution, operation* and *liquidation*. Similarly, Ishii [Ishii 1995] distinguishes between a number of life-phases, *product manufacture, assembly, consumer service* and *reuse/recycle/disposal*.

Tjalve's Model

As Tjalve [Tjalve 1979] explains, all artefacts are *created*, *used* and eventually *discarded*. By expanding and arranging these events in a sequence, Tjalve provided one of the first artefact life models, shown Figure 2.3. The first phase in this model is *'design'*, during which Tjalve states that possible methods of satisfying the user needs are examined and the finally chosen product is completely specified. As acknowledged by Tjalve, the model is not intended

to detail every step in the life of the product - for example it omits the stage during which the production method is designed and chosen. The next phase is *'manufacturing'* during which the design solution is realized as a physical artefact. Following this, the product is sold in the 'Sale' phase, first to the dealer and then to the end customer. During the *'Using process'* phase, the artefact functions according to its intended purpose. The artefact's life ends with *'Destruction'*.

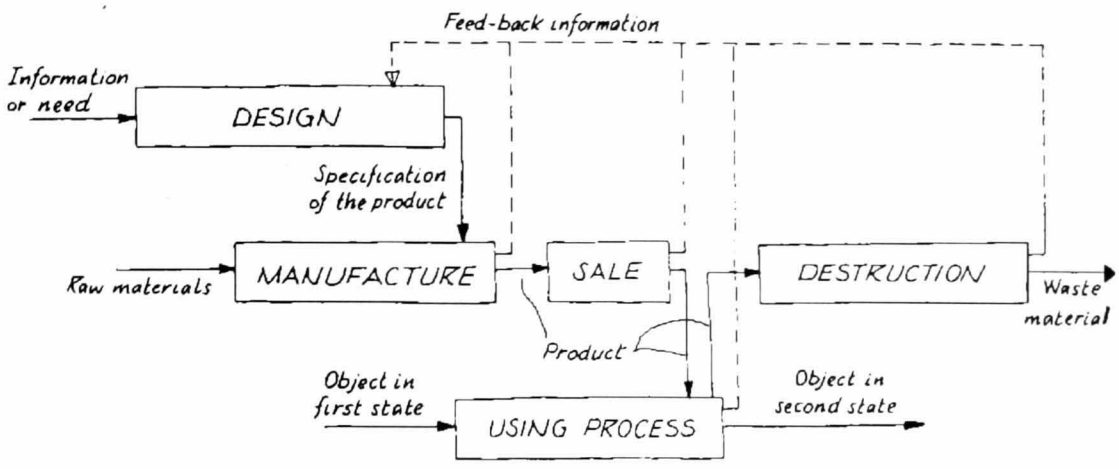


Figure 2.3 – Tjalve's artefact life model - [Tjalve 1979]

Tichem's model

A more recent model by Tichem [Tichem 1997] characterizes the artefact life in terms of *design, parts manufacture, assembly, distribution, use/service* and *retirement* phases (Figure 2.4). Tichem explicitly includes 'parts manufacture' and assembly, which are activities that are significantly influenced by design phase decisions.

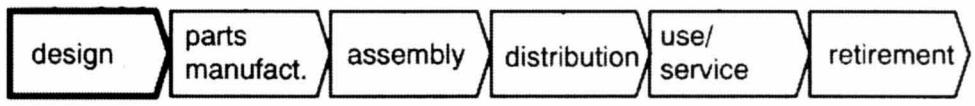


Figure 2.4 – Tichem's artefact life model - [Tichem 1997]

Olesen's model

Olesen [Olesen 1992] provides a more detailed life model by including typical systems effecting the transformations occurring in each phase (Figure 2.5).

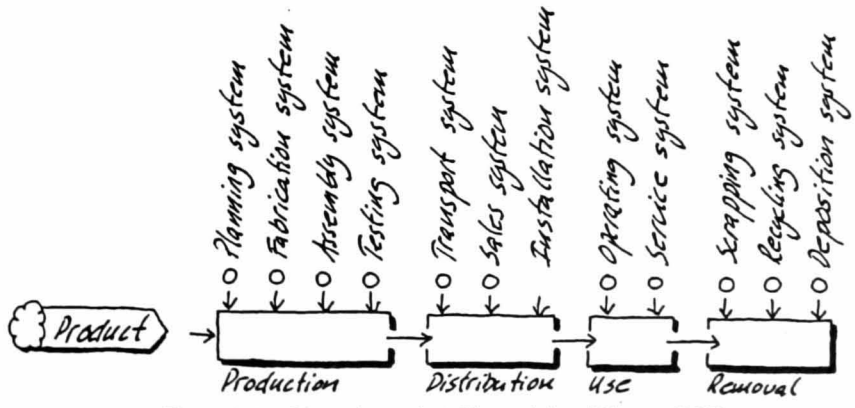


Figure 2.5 - Olesen's artefact life model - [Olesen 1992]

Roozenburg & Eekels' model

A model partially covering the artefact life is that found in [Roozenburg et al. 1995], consisting of the *product planning, strict development* and *realization*

phases (Figure 2.6). In these phases, a number of *activities* are executed e.g. an activity during strict development is production development. This model highlights that product development handles *information*, the outcome being a product design solution, marketing plan and production plan.

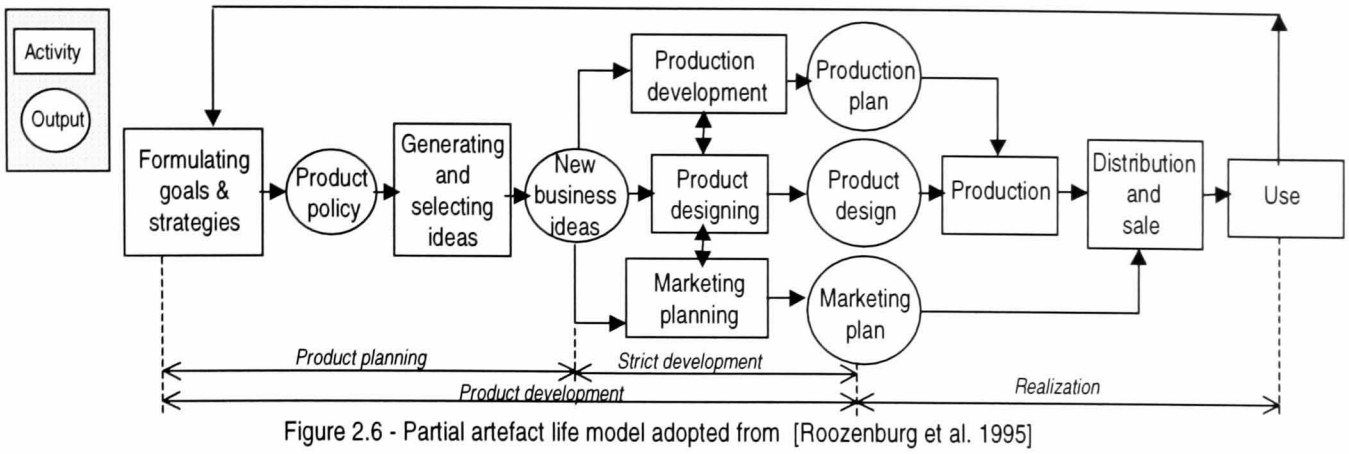


Figure 2.6 - Partial artefact life model adopted from [Roozenburg et al. 1995]

Hales' Model

Hales provides a model (Figure 2.7) characterizing the artefact life in terms of the artefact's *state* and corresponding life *activities* [Hales 1993]. For example, the initial state is information (idea/need/proposal/brief), which is input into the market and problem analysis activities in the 'task clarification phase'. Later on this evolves into a 'working product' that is input to the use phase. This model also represents information feedback loops between activities e.g. 'use experience' is fed back to the conceptual design phase.

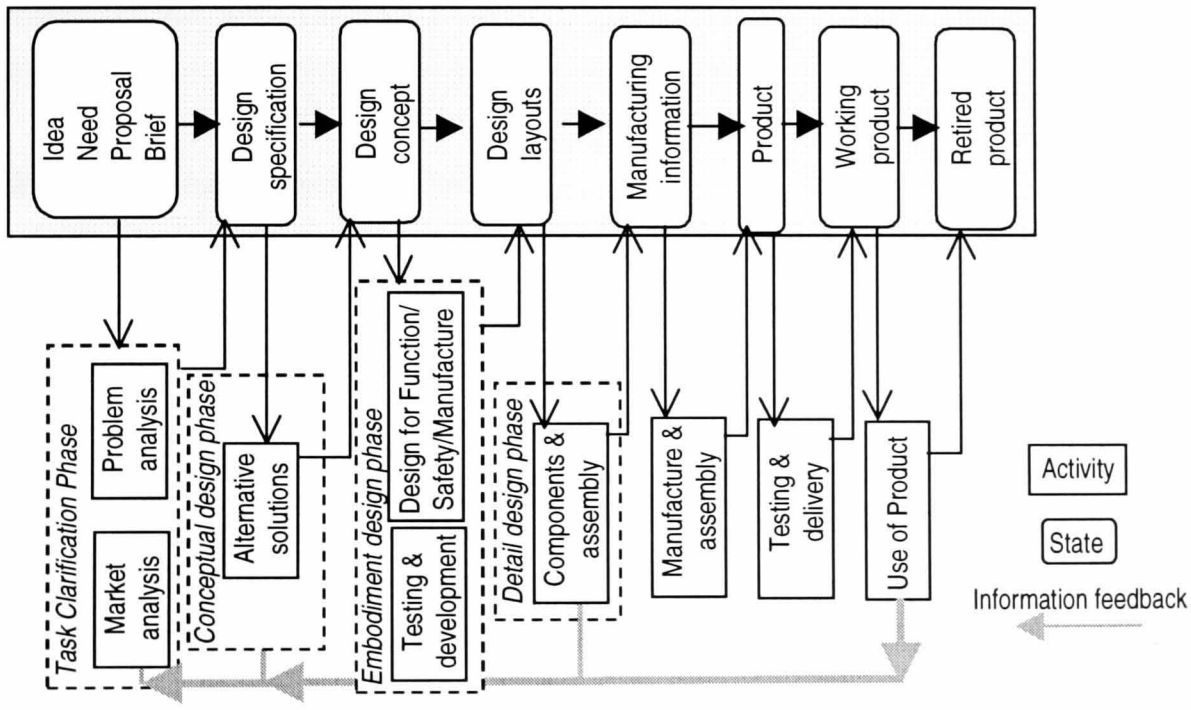


Figure 2.7- Hales' artefact life model – adopted from [Hales 1993]

2.2.1 Life-phase Transformations

Autonomous & purposeful transformations

The sample models presented reflect that an artefact life is composed of a number of phases during which there is a transformation of an *operand* from an initial state, to another state. Of relevance is that Roozenburg & Eekels

[Roozenburg et al. 1995] state that the world around us is being continuously transformed from a state S_1 through a natural, autonomous course of events (Figure 2.8a) to a state S_2. They explain that this autonomous course of events can be intervened by *purposeful action* to change the autonomous transformation to a new direction S'_2 (Figure 2.8b).

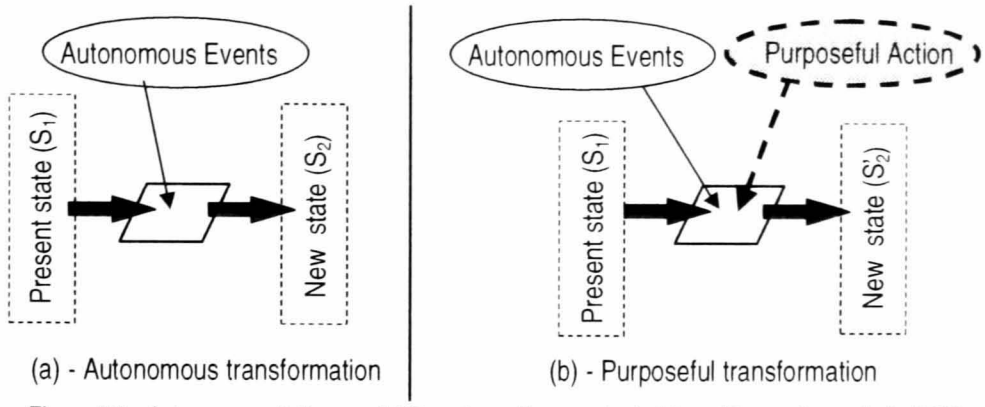


Figure 2.8 - Autonomous & Purposeful Transformations - adopted from [Roozenburg et al. 1995]

Transformation consequences

The purposeful transformation from state S_1 to S'_2 does not always lead to just the desired effects, but also to *side-effects* [Roozenburg et al. 1995] formally:

$$S_1 + \text{purposeful action} \Rightarrow S'_2 = (\text{desired effect} + \text{side-effects}).$$

This reflects that resulting effects *(consequences)* of purposeful actions are not only those that are desired and hence *intended* but include *unintended* side-effects. For instance, the transformation of a blank piece of mild steel into a component of a specific form and dimension (desired effect) with a milling process (purposeful action) is accompanied by side-effects: chip formation, noise, tool wear and material wastage.

2.2.2 Evolving Operand States

Operand states

Some models (e.g. Tjalve's) reflect that during the artefact's life, there is a flow of *information, energy* and *material*, this in accordance with the *theory of technical systems* [Hubka et al. 1988].

Intra-phase states

An artefact exists in different *intra-phase* states. For instance, during its realization phase, an artefact, (e.g. a fluid valve), may be in the form of raw material, then cast to the required form, resulting in a rough surface and later, following a grinding process, to a state in which it has a superior surface finish.

Inter-phase states

An artefact also exists in states during the interface between one phase and another, this termed an *inter-phase* state e.g. at the start of the use phase, the artefact is in a new condition, whilst at the end of the use phase, it is usually in a state in which its functional performance is lower.

2.2.3 Phase Composition

Life-phase systems

The concept of a phase involving an *autonomous* or *purposeful* transformation process of some operand, enables a phase to be viewed as consisting of a set of transformation systems [Hubka et al. 1988] that deliver the transformation effects (Figure 2.9a). For instance, Figure 2.9b illustrates an example from [Olesen 1992] of a production phase, consisting of a fabrication and assembly system. The fabrication system is in turn composed of a milling machine.

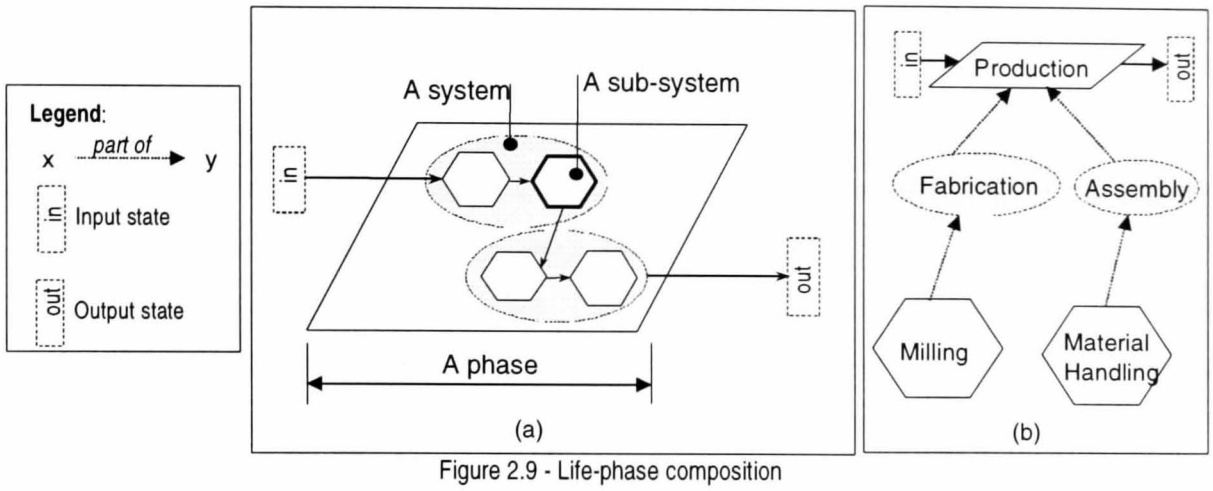


Figure 2.9 - Life-phase composition

2.2.4 A Decomposable Artefact Life Model

Phase nomenclature

As with any system concept, a life-phase can therefore be decomposed into different levels of resolution. As evident from the models presented, this results in a lack of correlation between the nomenclature and number of phases employed to describe the life of a mechanical artefact. To avoid confusion, this research considers a mechanical artefact life to consist of the *design*, *realization*, *use* and *disposal* phases, composed of systems (Figure 2.10) that cause the relevant transformations. Figure 2.10 also shows the *inter-* and *intra*-phase states in which a mechanical artefact (as an operand) exists during its life. An outline of the transformations performed and typical systems involved in these different life-phases follows next.

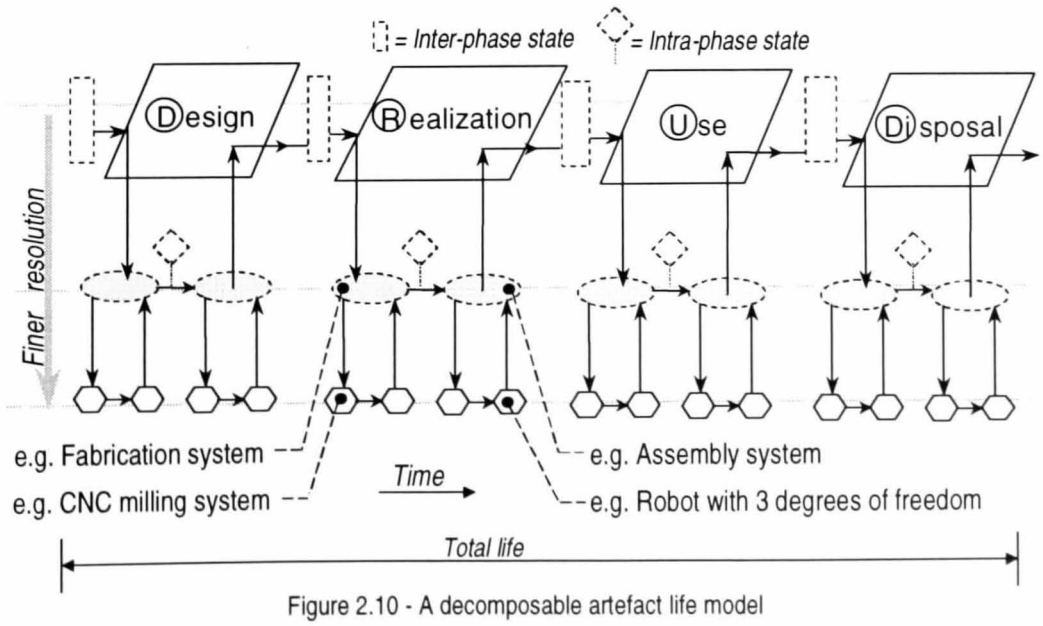


Figure 2.10 - A decomposable artefact life model

Design Phase

Design transformation

The role of the design phase is that of transforming a *design problem* into a *design solution description* that satisfies the required needs. The solution, an 'inter-phase' description, is used to initialize the transformation in the realization phase (Figure 2.10). The form in which artefact solution descriptions (information) are mostly communicated with are drawings [Ullman et al. 1990], either paper based or, with CAD/CAM systems, in the form of computer based models [McMahon et al. 1993].

Design phase systems

The human being, is a natural system [Hubka 1985] in the design phase that causes the transformation of a design problem (information) into a solution, through a number of purposeful activities such as decision making, sketching and synthesizing [Mills 1993; Thomson 1995]. Other systems may also be employed, such as a finite element analysis system [Cook 1995], purposely transforming a geometrical model into analysis results.

Realization Phase

Realization transformation

This phase transforms a design solution into a physical artefact. It basically consists of fabrication and assembly systems for *processing* and *assembly* operations [Groover 1996] involved with realizing mechanical artefacts.

Realization systems

Through a fabrication system, a technical process such as sand casting, transforms the input (raw material) into a different material state. The fabrication system itself may consist of sub-systems such as a mould system, basically consisting a cavity, core, runners and gates. The realization phase can also consist of assembly systems to effect the transformation of a number of separate PDEs into an assembly. This phase's composition is influenced by the *quantity* and *variety* of artefacts being realized as these influence the type of *automation (rigid or flexible)* that can be employed. Another influence is a shift from mass realization of artefacts to mass customization [Eastwood 1995].

Use Phase

Two transformations

This phase is concerned with the functioning of the artefact (normally) for the purpose it was intentionally designed. This research therefore views two transformations taking place during the use phase:

- the artefact as an 'operator' transforming an input operand into a new state; this *purposeful transformation* is that for which the artefact was designed (e.g. Table 2.1). This repeats itself many times during the use phase;

- the artefact as an 'operand' - a brand new artefact transformed into an artefact with a deteriorated functional performance - this *ageing transformation* is spread over the whole time span of the use phase;

Table 2.1 Typical use phase transformations - adopted from [Tjalve 1979]

Operand Input state	**Artefact**	**Operand output state**
A whole sheet of paper	Scissors	Sheet of paper cut into pieces
Plastic granules	Extruder	Continuous length of plastic profile with the required cross-section.
Person needing entertainment/ information	Television	A person entertained and informed.

Use systems

Table 2.1 shows that when an artefact is *being used* it operates *on* an operand to cause the *purposeful transformation* for which it was designed and developed. The artefact is however still an operand to the environmental system in which it is employed. Also, while in use, artefacts encounter service systems during which various activities (e.g. adjusting loose parts, replacing worn elements) revive the artefact's performance to an acceptable level [Tichem 1997] thus delaying the ageing transformation.

2.2.4.4 Disposal Phase

Disposal transformations

The disposal phase is concerned with transformations that take place *after* the intended useful function of the artefact can no longer be fulfilled. During the disposal phase a number of scenarios can take place [Wang et al. 1995]:

- *material disposal*: eliminating the artefact without any material recovery;
- *material recovery* - material is reused by either: *recycling*, *re-manufacture* or *re-use* of the artefact's PDEs;
- *energy recovery* - energy stored in the material is used for some purpose.

Disposal systems

The transformations in the disposal phase can be achieved through a number of systems such as a scrapping system or a recycling system [Olesen 1992].

2.3 Artefact Life Observations

Based on the decomposable artefact life model and a number of examples, this section presents a number of observations leading to the disclosure of the artefact life phenomena causing the need for a DFΣX approach.

2.3.1 PDE Life Sequence

The artefact life model discloses an overall *chronological* sequence of phases through which an artefact passes during its life. This sequence applies to different PDEs as reflected in the following examples:

Product — An aircraft cannot be *realized* and then *used* before all its elements are *designed*. Similarly, an aircraft does not meet a *servicing* system before the aircraft is *used*;

Sub-assembly — A jet engine cannot be *assembled* before its elements are *designed* and *realized*;

Component — The rotor blade of the jet engine cannot be *fabricated* before it is *designed*;

Component element — A hole cannot by *realized* before its parameters (e.g. diameter) are specified in the *design phase*. Similarly the surface finish of a hole need not be *serviced* before the hole's surface is subjected to wear during the use phase.

2.3.2 Interleaved PDE Lives

Separate PDE lives — As argued in section 2.1, mechanical artefacts are composed of a number of different PDEs. The existence of a PDE in an artefact depends on the existence of the artefact itself. However, when for some reason or other, a PDE reaches its disposal phase (e.g. a bearing), it does not necessarily mean that the artefact (e.g. a car) it belongs to, also reached its disposal phase (Figure 2.11). Rather, the car in this case, still in its use phase, needs to interact with a *service system* to replace the bearing.

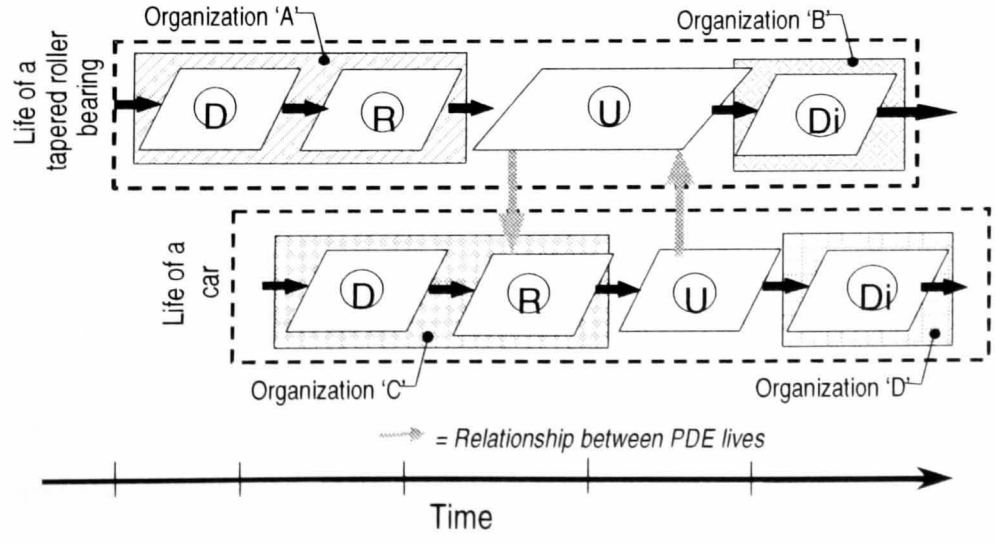


Figure 2.11 - Interleaved relationship between different PDE lives

This thesis therefore argues that a PDE can have a *separate* but *related* life to the artefact to which it contributes. Chal & Linde [Chal et al. 1997] provide similar arguments, that by considering the flow of material, energy, information and money, there exists a 'network of product lives'. This means that different PDEs forming part of an artefact, can encounter systems in different organizations (see Figure 2.11). This fact, relevant for artefact *life-specific* design, is not made explicit in the models presented in section 2.2. For instance Tichem treats 'parts manufacture' as forming part of the *product life model*.

2.3.3 Artefact and Life-phase System Interactions

Natural and artificial systems

During the course of its life, an artefact *meets* [Mortensen et al. 1996] a number of life-phase systems [Andreasen 1992] such as a service system, transport system and disposal system. Based on observations made, this thesis extends this meeting concept, in that during the course of its life, an artefact *interacts* with *natural* and/or *artificial* systems [Hubka et al. 1988]. An electrochemical machining system (ECM) [Groover 1996] is an example of an artificial system which an artefact purposely interacts with in the realization phase to be transformed from the initial raw material form into a different, desired form. Sea water, is an example of a natural system which a mechanical artefact (e.g. ship propeller) can interact with during its use phase. An artefact can therefore be in different states, (Figure 2.12) when interacting with life-phase systems.

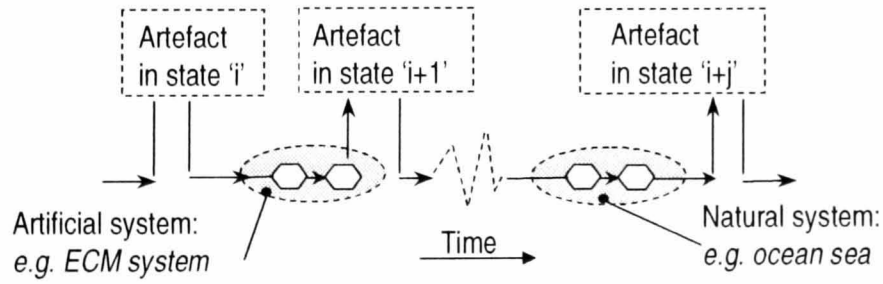


Figure 2.12 – Interactions between artefact and life-phase systems

Interaction resolution

A PDE meeting an assembly system is one level of resolution about the interaction with the realization life-phase. A more specific interaction level is that the assembly system being met by the PDE consists of a robot assembly device with 3 degrees of freedom, rather than a manual assembly system. This interaction resolution reflects that it is not enough for designers to think of life-phases (e.g. realization) as simply a black box. Otherwise, how can designers consider, say, assembly issues if the composition of the specific assembly system eventually interacting with the artefact is not explicitly known? The decomposable artefact life model (Figure 2.10) allows artefact and life-phase system interactions to be viewed at different resolutions.

2.3.4 Interaction Consequences

Derived from industrial visits carried out during this research and examples encountered in literature, this section will highlight typical consequences that result from the interactions between artefacts and life-phase systems.

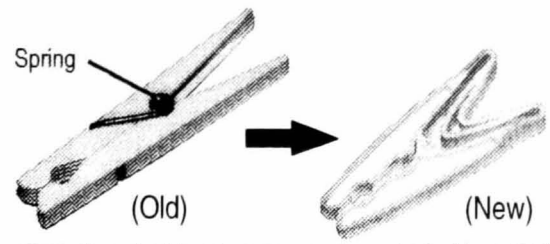


Figure 2.13 – Typical product level consequence - adopted from [Fabricius 1994]

Product level

Case 1

To improve the interaction between a clothes peg design solution and the realization phase systems, designers reduced the number of components in the artefact structure by eliminating the spring element (Figure 2.13). However, this influenced the interaction of the peg and the use phase as the repetitive opening/closing of the new peg was less durable than the older one, this resulting in a shorter useful life [Fabricius 1994]. Hence, the peg reached its disposal phase in a shorter time span, increasing costs to the user.

Sub-assembly level

Case 2

A designer of a car dashboard switch sub-assembly, revealed that the specification of an ultrasonic bond as a means to join two thermoplastic components had a positive consequence on the assembly system used in the realization phase, Figure 2.14. Fasteners were eliminated and hence components kept to a minimum, this complying with a *design for assembly* [Boothroyd et al. 1991] principle. Thus, handling and storage costs were reduced.

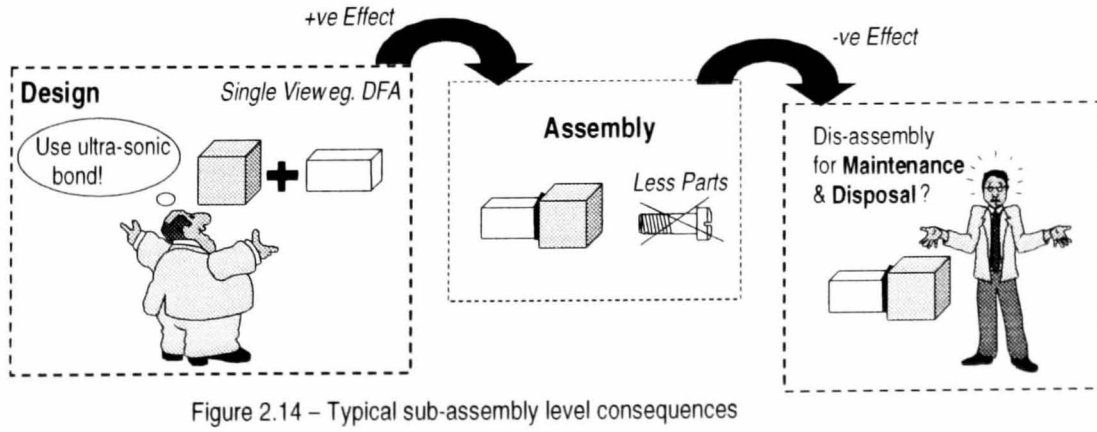


Figure 2.14 – Typical sub-assembly level consequences

However, the resultant sub-assembly was difficult to separate without damage when it interacted with the inspection system in the realization phase or the maintenance system in the use phase. Thus, a fault to one of the

components required the replacement of the whole switch sub-assembly. Also, separating the components made from different thermoplastic materials makes disposal phase recycling operations more time consuming and costly.

Case 3

In a design example concerning a printer sub-assembly, eliminating the steel insert (Figure 2.15) in the gear and the set screw, simplified the composition of the realization phase systems, as less parts had to be fabricated and assembled [Bralla 1996]. However this made the interaction with the service system more time consuming and costly.

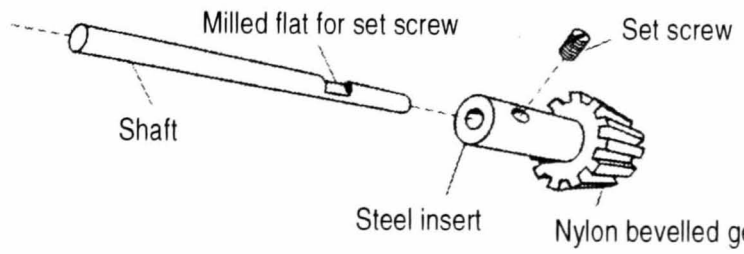


Figure 2.15 - Typical sub-assebly solution influencing the artefact life – adopted from [Bralla 1996]

Component level

A component is composed of PDEs such as form features and assembly features. Typical influences on systems encountered during the component's life, due to these PDEs will be presented through the following cases.

Case 4

Figure 2.16 illustrates alternative machine crank design solutions intended to serve the same function in the use phase. The material specified influences the type of fabrication process(es) that can be used. One alternative is to forge or cast a blank piece of material which then has holes generated by drilling, followed afterwards by reaming. The flats of the holes are milled to the required dimensions. The alternative design reduces the number of fabrication processes required in the realization phase as it is generated by powder metallurgy [Groover 1996]. Additionally, this process makes it possible to generate the component to tighter tolerances and with less material wastage. Further, the component's use phase reliability is improved as powder metal parts have a porous structure that retains lubricants. Also different tooling costs associated with the realization phase are influenced.

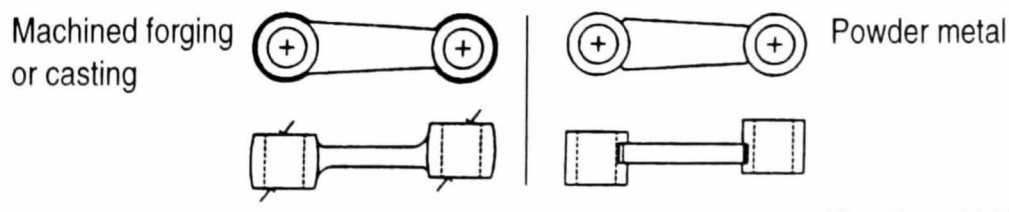


Figure 2.16 - Typical component material consequences - adopted from [Bralla 1996]

Case 5

Mould makers revealed examples of how thermoplastic component parameter values greatly influence interactions between the artefact and systems met

during the artefact life. For example, a sharp corner (i.e. radius = 0) gives rise to difficulties in the generation of the mould cavity as this has to be either generated through spark erosion (following the design and fabrication of an appropriately sized electrode) or through mould cavity construction (Figure 2.17). The latter introduces flashing defects when interacting with the injection moulding system and requires longer assembly and mould part alignment when constructing the mould. An alternative, positive radius value, permits the use of a milling system for fabricating the mould cavity, giving large savings in time and costs. In addition, filleted corners are less easily chipped thus increasing the useful life of the component and implicitly conveying a superior quality image to the customer in the use phase.

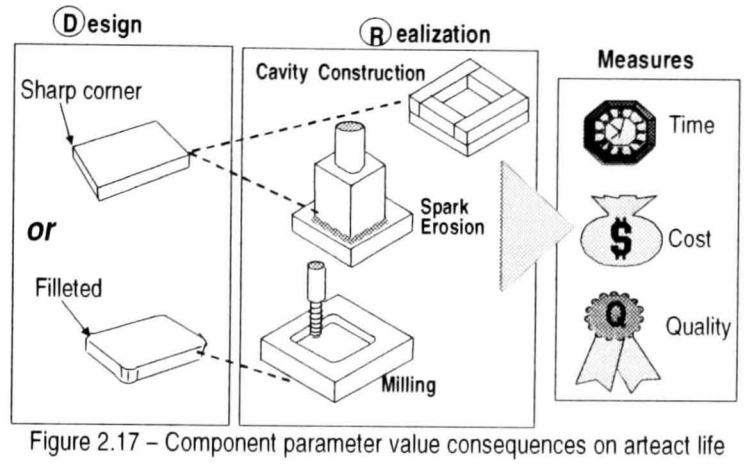


Figure 2.17 – Component parameter value consequences on arteact life

Case 6

An example of the influence parameter values have on *multiple* life-phases is derived from a development scenario of a photographic camera's thermoplastic facia.

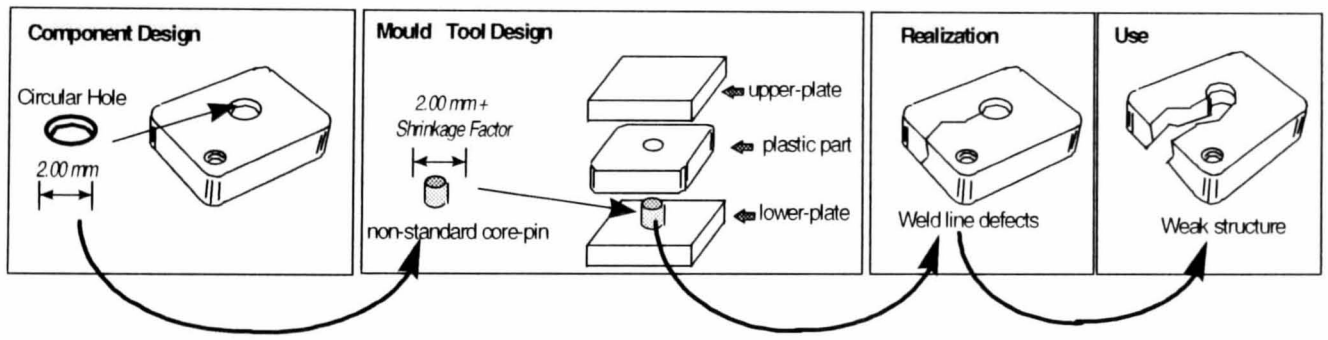


Figure 2.18 - Component parameter value consequence propagation effect

This facia had a circular hole (Figure 2.18) which the designer specified with a nominal diameter of 2.0mm. In practice, mould tool designers specify a diameter that takes into consideration the material's shrinkage factor [Pye 1989]. This required a non standard core-pin not available from mould tool part suppliers. Pinpointing this problem and re-specifying the hole diameter value to avoid this problem, resulted in time delays and extra development costs. Also such a mould system with core pins frequently generates *weld line defects*. These increase scrap during the realization phase and introduce weak areas in the components that giver poor performance in the use phase.

Case 7

Another case derived from industrial practice concerns the scenario when thermoplastic component designers, define oblique holes (Figure 2.19). This results in an increase in the mould ejection mechanism degrees of freedom, (Figure 2.19b), at a considerable increase in mould design time and construction costs. Also, such mould design results in a slower component ejection period thereby reducing the component's production rate during the realization phase.

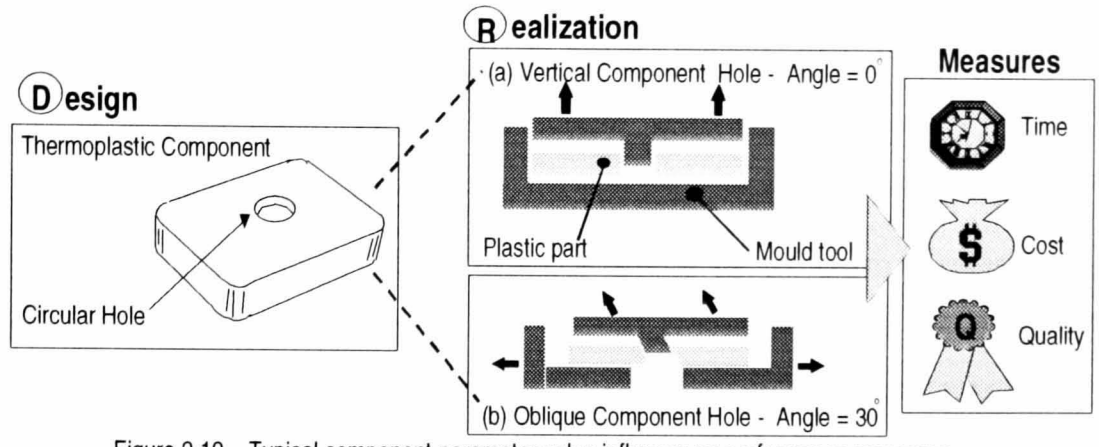


Figure 2.19 – Typical component parameter value influence on performance measures

2.3.5 Design Phase Implications

Undesired consequence generation

The design phase transformation involves decision-making. Correlating with arguments made in 2.2.1, the case studies presented in 2.3.4 reflect that decisions *purposely* made during the design phase, also result in *undesired* consequences on the subsequent life-phases. Similar arguments are found for instance in [Andreasen et al. 1990; Wallace 1997].

Influence on performance measures

The cases presented (e.g. case 7) reflect that design decisions influence a life-phase systems' composition and hence transformation performance in terms of measures like *time*, *cost* and *quality*, this illustrated for the realization phase in Figure 2.20. Thus, apart from the creation of the product, the design process has an influence on a number of life-phase performance measures termed *universal virtues* [Olesen 1992]. These are a number of measurable quantities *cost, time, quality, efficiency, flexibility, risk* and *environment.*

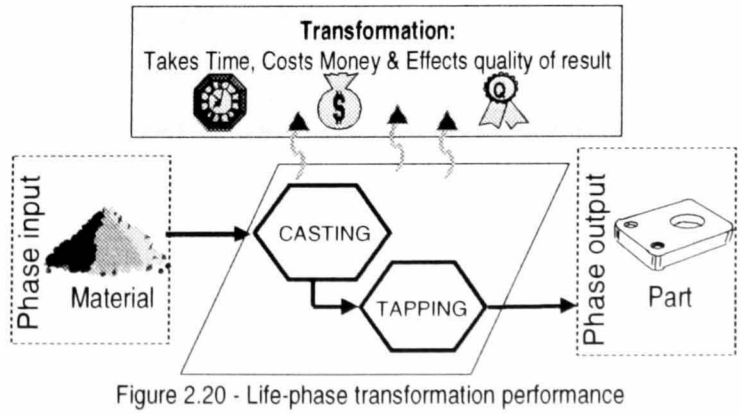


Figure 2.20 - Life-phase transformation performance

Cost

For instance, using *cost* as a performance measure, it is known that decisions made during early design, when about only 30% of the actual product development costs have been used, result in the allocation of about 70% of the production related costs [Andreasen et al. 1987].

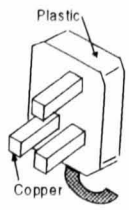
Quality

Empirical evidence [Andersson 1994] collected shows that the basic product quality is created during the *early*, conceptual design stage of the design process. As argued in [Swift et al. 1992; Morup 1993], both the quality (Q) experienced by the end-user and the quality (q) experienced by the company's internal stakeholders is influenced by design.

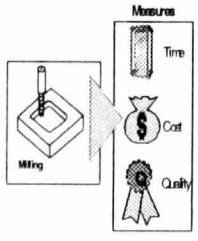
2.3.5.1 Life Cycle Consequences

Type of consequences

Therefore, design phase decisions are the source of different consequences on different life-phases, these being termed in this research as *life cycle consequences (LCCs).* As characterized in this research, these LCCs are:



i. *artefact's behaviour:* certain design decisions influence the artefact's behaviour e.g. defining the material of a component to be a kind of thermoplastic, results in the component not conducting electricity during the 'use' phase, unlike with say a copper component;



ii. *life-phase system behaviour:* design decisions influence performance measures such as cost, time and quality. For instance, case 5 shows that changing a component's radius parameter value influences the performance measures of the realization phase. Similarly specifying a diesel rather than a petrol engine to a car design, influences the service system employed, thereby influencing maintenance costs and time;

Aluminium
=>
{Supplier? &
Fabrication process?}

iii. *creation of a new decision space* in the same or different life-phases. For instance the decision to specify aluminum as a material gives rise to a space of decisions that need to be made. For instance, 'who will supply aluminum? ' and 'what fabrication process to employ with aluminum?'

Thermoplastic
=>
{Drilling?
Moulding?
EDM?
Milling ?
...? }

iv. *introduction of new artefact life-phase constraints*: for instance, the definition of a thermoplastic material to a component design, *restricts* the set of assembly means and the fabrication processes that can be employed - the component cannot be fabricated with a spark erosion machine (EDM). Similarly, the dimensional tolerance that can be attained is restricted.

2.3.5.2 Propagation Effect Phenomena

Sequential victimization

The examples presented show that a decision made during the design phase can be good (✓) for one life-phase, but problematic (✗) to *other* phases in terms of performance measures such as cost and lead time (Figure 2.21).

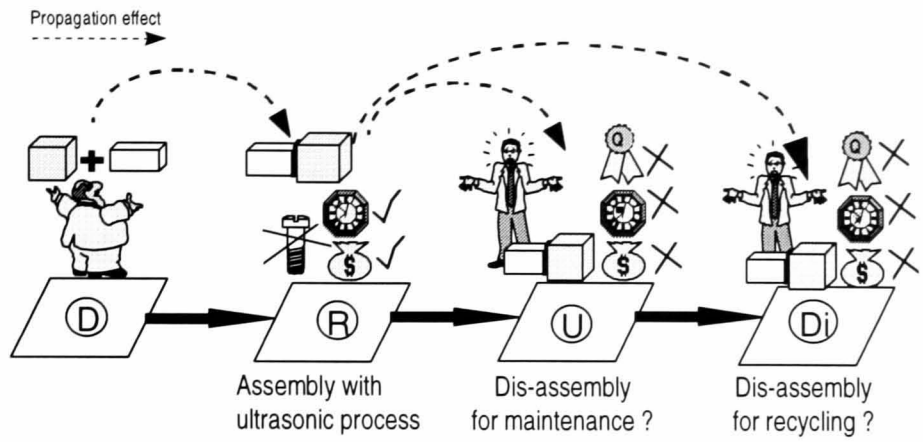


Figure 2.21 - Phenomena of propagation effects

Thus, a decision made *during* the design phase can *propagate* a number of consequences *across multiple* artefact life-phases, collectively giving rise to what is termed in this research as a *propagation effect* [Borg et al. 1996]:

> "A design decision made to achieve an intended effect gives rise to other decisions and influences on performance measures in other life-cycle phases. The cumulative consequence resulting from this design decision is termed the decision's propagation effect."

Influence on total life behaviour

Due to this propagation effect phenomena, design decisions influence *multiple* life-phases in terms of different measures. This correlates with Andreasen [Andreasen 1992] who states that to a high degree, the design solution determines its life-phases behaviour. This behaviour can therefore be collectively mapped to a DFX matrix [Andreasen et al. 1993] (Figure 2.22).

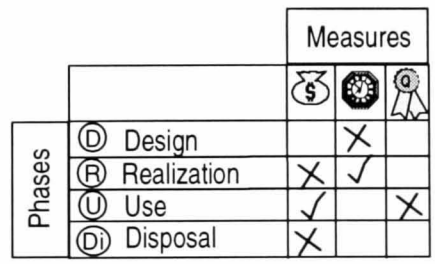


Figure 2.22 - DFX matrix concept - adopted from [Andreasen et al. 1993]

2.3.5.3 Designer Implications

Difficulty in acquiring consequence knowledge generated

During an artefact's life, due to the *interaction* between the artefact and different life-phases, important LCC knowledge is therefore being generated. However, due to an artefact's life chronological order, this knowledge generation takes place *after* the design phase. Thus, designers do not generally *acquire* knowledge concerning *LCCs* resulting from artefacts *interacting* with life-phase systems, unless explicitly provided with feedback.

Phenomena extends the designer's responsibility

By adopting a *Design For X (DFX)* approach [Bralla 1996], a solution with X-ability can be generated. After all, DFX is concerned with 'consequence fitting' [Andreasen 1992]. However, the examples presented in this Chapter show that due to the phenomena of propagation effects, a 'single-X'-ability approach can simultaneously result in a number of unintended LCCs. Thus, this research argues that the designer's responsibility covers *all* artefact life-phases. Rather than a narrow DFX approach, designers therefore need to adopt a *'Design for Multi-X (DFΣX)'* approach if they are to generate a solution fitting a *host* of artefact life requirements.

2.4 Chapter Conclusions

A decomposable artefact life model

This chapter characterized a mechanical artefact's life through a decomposable artefact life model composed of the *design*, *realization*, *use* and *disposal phases*. Different phases are concerned with the transformation of an operand from one state into another through a set of activities carried out by a number of systems.

Propagation effect phenomena

A significant characteristic occurring during the life of mechanical artefacts is the *interaction* taking place between the artefact and the various life-phase systems encountered. Examples presented reveal that such interactions are influenced by decisions made during the design phase. This reflects that, design phase decisions result in both *desired* and *undesired* consequences on the artefact's life. Moreover, as disclosed, such LCCs *propagate* across *multiple* life-phases. Therefore, due to an artefact's life chronological order, designers do not generally acquire experiential knowledge of such LCCs.

A design problem - need to DFΣX

As an implication of this propagation effect phenomena, the designer's responsibility therefore covers *all* artefact life-phases. Design must therefore handle this phenomena *during* the design phase if they are generate life-oriented design solutions. Designers therefore need to adopt a 'Design for Multi-X (DFΣX)' approach in order to generate a solution fitting a *host* of artefact life requirements. In order to identify what characteristics are required in a DFΣX approach for handling this propagation effects phenomena, the next chapter will therefore focus in more detail on the transformations taking place during the design phase.

chapter 3

3.0 Characterizing 'Design Synthesis for Multi-X'

Scope

The aim of this chapter is to disclose what characterizes a 'Design Synthesis for Multi-X' approach, which as argued in Chapter 2, is required to allow designers to handle the phenomena of propagation effects. To provide a basis for characterizing $D_sF\Sigma X$, section 3.1 presents design process characteristics. Section 3.2 then focuses on the significant role exploration plays in design, this leading to the disclosure of $D_sF\Sigma X$ characteristics in section 3.3. Chapter conclusions are made in section 3.4

3.1 Design Process Characterization

Design process models as a basis to understanding design

Understanding design requires the combined efforts of many different approaches [Smithers et al. 1990]. Design process models provide a means by which one can explain and perhaps even replicate certain aspects of design behaviour [Coyne et al. 1989]. Several researchers [Finger et al. 1989; Blessing 1994; Cross 1994] classify such models as essentially being either *descriptive* or *prescriptive.* Descriptive models reflect the way in which a design process actually occurs. They are therefore divided into *protocol studies* that describe observations of how designers work and *cognitive models* that attempt to describe the mental processes employed by a designer when designing. An example is [Smithers et al. 1990]. *Prescriptive* models provide a systematic or methodical sequence of stages or activities [Blessing 1994], in an attempt to pursuade designers to adopt improved ways of working [Cross 1994]. An example is provided by Pahl & Beitz [Pahl et al. 1996]. An extensive review of diverse design process models can be found in [Finger et al. 1989; Blessing 1994]. Of relevance to this Ph.D. research is that collectively, both types of models reflect that the transformation taking place during the design phase, can be discussed from different perspectives. Design may be considered as *a problem solving process*, *a stage based evolutionary process*, *a domain based process* and *a knowledge based process*.

chapter 3

3.0 Characterizing 'Design Synthesis for Multi-X'

Scope

The aim of this chapter is to disclose what characterizes a 'Design Synthesis for Multi-X' approach, which as argued in Chapter 2, is required to allow designers to handle the phenomena of propagation effects. To provide a basis for characterizing $D_sF\Sigma X$, section 3.1 presents design process characteristics. Section 3.2 then focuses on the significant role exploration plays in design, this leading to the disclosure of $D_sF\Sigma X$ characteristics in section 3.3. Chapter conclusions are made in section 3.4

3.1 Design Process Characterization

Design process models as a basis to understanding design

Understanding design requires the combined efforts of many different approaches [Smithers et al. 1990]. Design process models provide a means by which one can explain and perhaps even replicate certain aspects of design behaviour [Coyne et al. 1989]. Several researchers [Finger et al. 1989; Blessing 1994; Cross 1994] classify such models as essentially being either *descriptive* or *prescriptive.* Descriptive models reflect the way in which a design process actually occurs. They are therefore divided into *protocol studies* that describe observations of how designers work and *cognitive models* that attempt to describe the mental processes employed by a designer when designing. An example is [Smithers et al. 1990]. *Prescriptive* models provide a systematic or methodical sequence of stages or activities [Blessing 1994], in an attempt to pursuade designers to adopt improved ways of working [Cross 1994]. An example is provided by Pahl & Beitz [Pahl et al. 1996]. An extensive review of diverse design process models can be found in [Finger et al. 1989; Blessing 1994]. Of relevance to this Ph.D. research is that collectively, both types of models reflect that the transformation taking place during the design phase, can be discussed from different perspectives. Design may be considered as *a problem solving process*, *a stage based evolutionary process*, *a domain based process* and *a knowledge based process*.

3.1.1 A Problem Solving Process

Designing

The word *design*, when treated as a verb in literature, means the 'process' [van den Kroonenberg 1987; Hales 1993; Blessing 1994; Roozenburg et al. 1995] through which a problem is purposely transformed into an artefact solution. When, the term is used as a noun, design refers to 'the artefact' designed. This dissertation explicitly employs the term 'design' to refer to the process. Designing can be considered as a type of problem solving process [Andreasen 1991; Bahrami et al. 1993; Cross 1994; Roozenburg et al. 1995], where by a problem is understood [Roozenburg et al. 1995]:

> "We speak of 'a problem' when someone wants to reach a goal and the means to do so are not immediately obvious."

A well-defined problem, such as determining the volume of a cube, has clear goals, often one correct answer and associated procedures that generate an answer. On the other hand, a design problem is normally *ill-defined* [Dym 1994]. It is weakly understood, has vague goals and no definite solution.

Design types

The process of 'designing artefacts' can range from designing totally new concepts to re-designing [Bahrami et al. 1993] existing mechanical artefacts. Pahl & Beitz [Pahl et al. 1996] for instance distinguish between original, adaptive and variant design types. *Original* design involves the development of an idea that results in a totally new artefact solution. *Adaptive* design involves the adaptation of previous solutions to satisfy new design requirements. Adaptation requires some modifications to the previous design solution and thus the output is a combination of 'previous solution ideas' and 'new solution ideas'. On the other hand, *variant* design involves varying the size and/or arrangement of certain aspects of the chosen system with the function and the solution principle remaining unchanged. The difference in design types is essentially related to (i) the initial problem state or the goal state of the product and (ii) the *knowledge* available and necessary to solve the problem [Blessing 1993; Dym 1994].

Hubka & Eder's definition

Similar to many other researchers [French 1985; Pugh 1991; Blessing 1994; Pahl et al. 1996], Hubka & Eder [Hubka et al. 1988] consider design as essentially being:

> "A process performed by humans *aided by technical means* through which information in the form of requirements is converted into information in the form of descriptions of a technical system, such that this technical system meets the requirements of mankind."

This definition outlines key features of the design process. As a problem solving process, design is concerned with the *transformation* (conversion) of an operand (requirements) into a description of an artefact system. This transformation terminates when the solution satisfies the identified artefact requirements. Such a solution description should ideally contain all information necessary for the subsequent life-cycle phases of the artefact [Blessing 1994]. Further, it is human designers who effect and control the conversion process. It is humans who are therefore responsible in generating a solution catering for a host of artefact life requirements. Also, it acknowledges that *technical means* such as a calculator, play an important role in this process as they aid humans. The definition outlines the input and output associated with a traditional design process, however, providing little insight as to what *design problem solving activities* are involved in this transformation.

Problem Solving Activities

The transformation of a problem into a design solution is achieved through a number of *design activities* [French 1985; Andreasen 1991; Pugh 1991; Pahl et al. 1996]. A design activity as defined in [Thomson 1995] is:

> "A physical action or cognitive process to achieve a state change in the design and/or its associated elements (e.g. product design specification, domain knowledge, past cases)."

Basic design cycle

Roozenburg & Eekels [Roozenburg et al. 1995] describe these activities through the *basic design cycle* model (Figure 3.1), consisting of the activities *problem analysis, solution synthesis*, *simulation*, *evaluation* and *decision*.

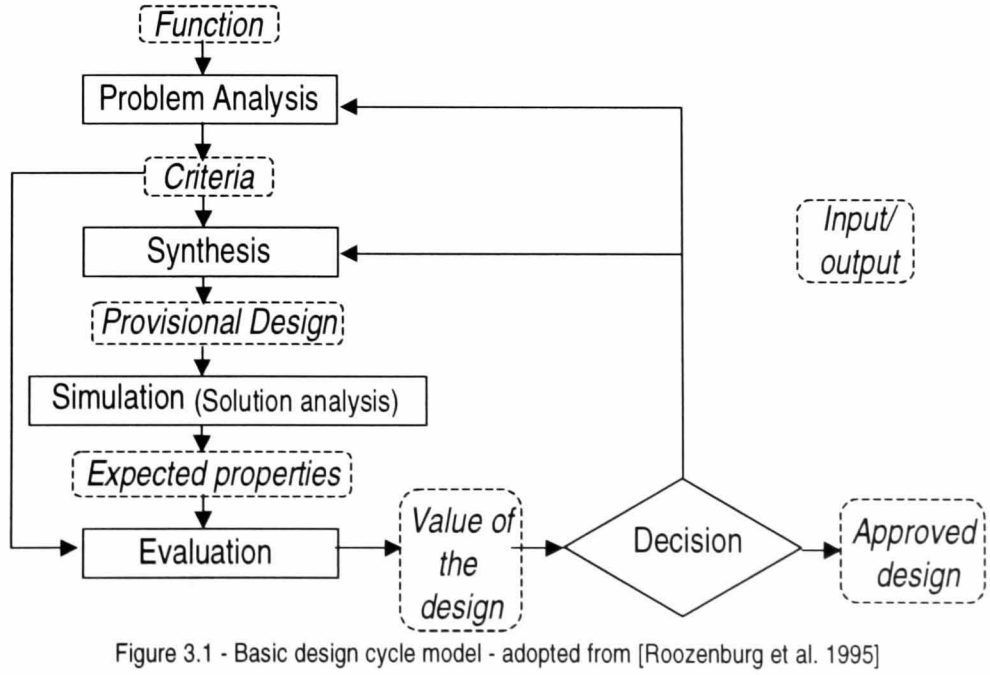


Figure 3.1 - Basic design cycle model - adopted from [Roozenburg et al. 1995]

Problem analysis

Problem analysis refers to the analysis of information on a problem. This activity results in a list of design requirements. Broad statements on the goal must be made, otherwise the designer will not know what has to be designed. To be able to determine later whether a proposed solution is indeed a solution to the problem, the goal is formulated as concretely as possible in the form of a list of requirements, termed the design specification.

Solution synthesis

Synthesis is concerned with the *generation* of a provisional design proposal. This activity is often regarded as the mysterious, 'human' creative part of designing [Cross 1994; Roozenburg et al. 1995], perhaps indicating why it is difficult to understand and model how it is carried out [Andreasen et al. 1996; Tomiyama 1998]. Basically, the word 'synthesis' means combining separate things, ideas and elements, into a complete whole [Roozenburg et al. 1995; Pahl et al. 1996]. This Ph.D. research argues that this assumes that sub-solutions have been found, *before* they can be combined into a whole. Thus, synthesis itself involves sub-activities [Olesen 1992] such as *search* and *discovery* together with the *composition* and *combination* of elements [Pahl et al. 1996]. Similarly, Blessing [Blessing 1994] states that *'generate'* concerns *creating* or *finding* elements of the solution whilst '*synthesizing'* concerns *combining* the elements into possible solutions by *structuring*, *varying* and *arranging.* Figure 3.2 provides a simple model, explaining the difference between some of these sub-activities, as seen in this project.

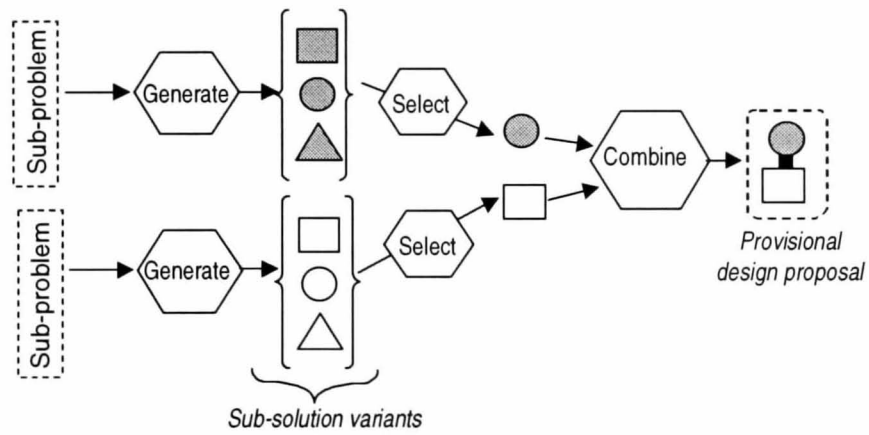


Figure 3.2 - Some sub-activities involved in synthesis

Synthesis is a knowledge intensive activity. Domain knowledge, as well as knowledge about the interactions between sub-problems, knowledge of how to map from a requirement to some sub-solution and knowledge of how combinations of partial solutions can be constructed is employed [Smithers et al. 1990; Duffy et al. 1996]. The result of the synthesis activity is a provisional design solution - it is not yet more than a possibility. Synthesis therefore expands the solution knowledge to a more concrete state [Smithers et al. 1990].

Solution analysis

Simulation, termed by engineering designers as 'analysis'[Roozenburg et al. 1995], is concerned with forming an *image* (artefact may not exist in reality) of the expected properties of the solution by deductive reasoning [Roozenburg et al. 1995] and/or testing models. Roozenburg & Eekels state a term used by engineering designers for this simulation activity is 'solution analysis.' For simulation, a whole array of technological and behavioral scientific theories, formulae, tables and experimental research methods is available to the designer. However, in practice, many simulations are based merely on *generalizations from past experience* [Roozenburg et al. 1995]. To avoid confusion with the terminology used to describe the activity by which the expected properties are estimated [Blessing 1994], this dissertation will use the term 'solution analysis'.

Evaluation

Every solution, independent of the artefact state, has to be examined to determine whether it is worth pursuing [Blessing 1994]. This requires the *comparison* of the solution with the problem statement and requirements, and with alternative solutions. To do so, the *expected properties* derived with solution analysis, are compared with the design specification criteria *(desired properties)*. As the two types of properties are likely to differ, a judgment as to whether those differences are acceptable or not is required.

Decision

Evaluation provides knowledge to enable a *decision* about the design cycle to be made, on whether to approve the design solution, or whether to try again and *generate* a better design proposal. Usually, the first proposal is not considered acceptable, thus requiring the designer to return to the synthesis step, (now with knowledge of the current solution evaluation results) to attempt to do better in a second, third or n^{th} *iteration* (Figure 3.1)

Evolutionary cycle

The basic design cycle is thus iterative in nature [Roozenburg et al. 1995]. Each iteration *incrementally* refining the knowledge of both the design problem and the solution. However, it does not provide structure to the design process with respect to the solution evolution occurring from one state to another.

3.1.2 A Stage-based Evolutionary Process

Four main design process stages

Due to the iterative nature of the basic design cycle, the design process consists of a number of *stages*, sometimes termed phases [Pahl et al. 1996]. Therefore, the principle mode of transformation taking place during each stage is still the same. A stage is a *segment* of the design process that

results in a more concrete artefact solution state, a function structure, a solution principle, an embodied design and eventually as a detailed design. In reality, each of these stages may cover a considerable period of time. A number of researchers segment design into such stages [French 1985; van den Kroonenberg 1987; Pugh 1991; Blessing 1994; Cross 1994; Pahl et al. 1996]. Typical models by French, Pahl & Beitz and Pugh are partially illustrated in Figure 3.3a, 3.3b and 3.3c respectively. Although the terminology used differs slightly, the models collectively reflect that design consists of four main stages [Roozenburg et al. 1995]: *task clarification, conceptual design, embodiment design* and *detail design.*

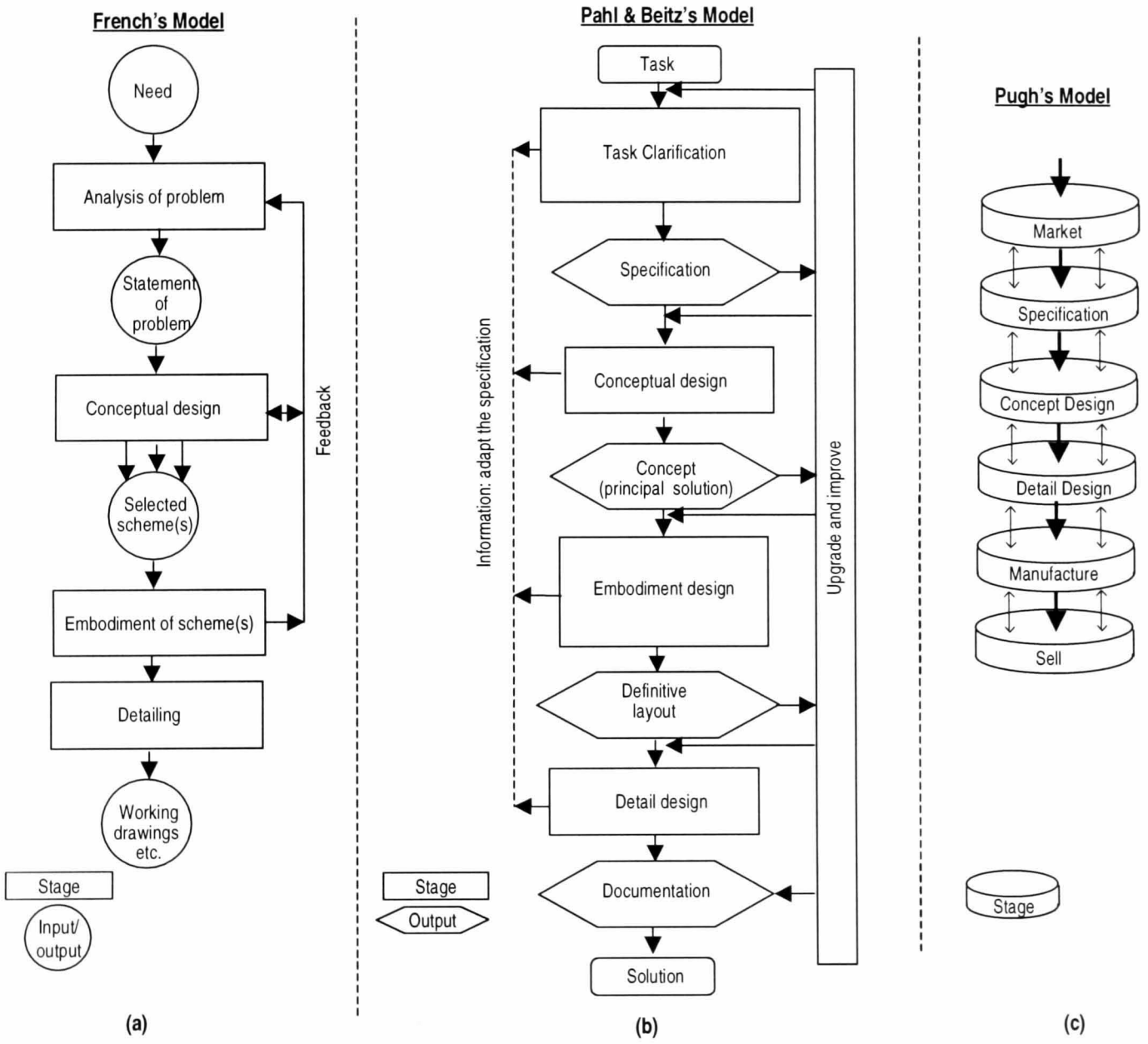


Figure 3.3 - Stage-based design process models

Task Clarification Stage

Transforms a need into a PDS

The information known about the artefact when design commences is that of a *need* [French 1985]. The purpose of the *task clarification st*age is that of transforming this need into a *Product Design Specification (PDS)* [Pugh 1991]. A PDS has sufficient detail so as to define specific targets towards which one

can aim the design effort and against which one can, eventually, measure the success of a solution. This transformation is achieved through various activities such as questioning the client [French 1985], market surveys [Pugh 1991] and the gathering of problem information [Pahl et al. 1996]. The output of the task clarification stage is a PDS that includes technical, social, economic and cultural requirements that the artefact should fulfil and constraints known at that time [Roozenburg et al. 1995]. The task clarification stage is important as the PDS generated directs the work carried out in the other stages. Work executed in the later stages may provide a new insight into the problem at hand, requiring a modification and refinement of the initial PDS as reflected by feedback loops in Figure 3.3.

Conceptual Design Stage

Transforms a PDS into a concept

The PDS is the input to the stage known as *conceptual design*, during which there is a search for a number of different solution *concepts* [Pahl et al. 1996] also termed *schemes* [French 1985], that can be used to solve the stated design problem. The transformation during the conceptual design stage is achieved by a number of activities [Pugh 1991; Pahl et al. 1996] *identifying the most crucial or essential problems*, *establishing a function structure*, *formulating a solution procedure* that can be applied to the design problem; *preparing concepts* and *evaluating* candidate conceptual solutions against the relevant criteria. The scheme generated, should represent the artefact in sufficient detail to enable certain desired properties such as weight and costs to be estimated. The conceptual design stage is the most open-ended part of design [Dym 1994]. Traditionally, in this stage, the designer's focus is mostly on the function of the artefact rather than its form [Dym 1994]. Of relevance to this research is that as argued by French, the conceptual design stage is where the most important *decisions* are taken. Hence, the final solution generated will have a trait of the solutions generated in the conceptual design stage [Pugh 1991]. Thus, it is the stage where there is the most scope for striking improvements [French 1985] because a weak concept cannot be turned into an optimum detailed design [Roozenburg et al. 1995].

Embodiment Design Stage

Transforms a concept into a definitive layout

During the *embodiment design stage,* the relatively abstract descriptions of the artefact (concepts) are made more concrete with the selection and sizing of major artefact subsystems. Embodiment design is essentially a process of continuously refining a concept, jumping from one subproblem to another,

anticipating decisions still to be taken and correcting earlier decisions in the light of the current state of the design proposal [Roozenburg et al. 1995]. The output of the embodiment stage is *one* selected scheme termed a definitive layout. This usually evolves from an intermediate solution termed a preliminary layout [Pahl et al. 1996]. The preliminary layout is obtained by a number of activities: *refining the conceptual designs*, *evaluating* and *ranking* them against the design specifications and than *choosing* the best. The definitive layout is obtained by *optimizing the preliminary design* and by *preparing preliminary parts lists and fabrication specifications*. The definitive design defines the layout of sub-assemblies and components as well as their geometrical shape, dimensions and materials [Roozenburg et al. 1995]. Pugh argues that embodiment takes part during the conceptual design stage.

Detail Design Stage

Transforms a definitive layout into detailed artefact solution documents

The scheme selected in the embodiment stage is worked out in greater detail during the *detailing stage*, resulting in documentation of the designed artefact, traditionally in the form of assembly drawings, detail drawings and parts lists, which are communicated to the realization phase [Roozenburg et al. 1995]. The transformation in this detail design stage is achieved by refining and optimizing the definitive layout [French 1985; Pahl et al. 1996]. This involves fully specifying and documenting the structure of the solution and the shapes, dimensions, tolerances, surface properties and materials of all the individual components into final fabrication and assembly documents.

Observations on Stage based Models

Stage based models portray the impression that complex problems can be split into sub-problems, for which solutions can be found and then combined into an overall solution. In reality, due to the relationships between the various output (e.g. PDS and principal solution structure), there is no clear division between the stages. Nevertheless, stage models make design more transparent, thus providing intermediate results that can be aimed for during design [Roozenburg et al. 1995].

3.1.3 A Domain Based Process

Artefact solution viewpoints

The artefact solution evolving and being handled during design can be viewed from different viewpoints which Andreasen [Andreasen 1991a] defines as *process, functional, organ* and *constructional* domains (Figure 3.4). Each domain describes a particular degree of abstraction and detail in the creation

of an artefact. Through the *process domain perspective,* an artefact is viewed as a *system* that transforms energy, materials and information. The *functional domain* views the artefact as a system of functions that are necessary for realizing the effects the artefact must produce. The *organ domain* views the artefact as a system of organs, each organ being a set of material elements (e.g. a journal bearing) that realizes a desired function (e.g. rotational freedom) by exploiting physical effects. The *constructional domain* views the organs realized as components or elements to make up an artefact.

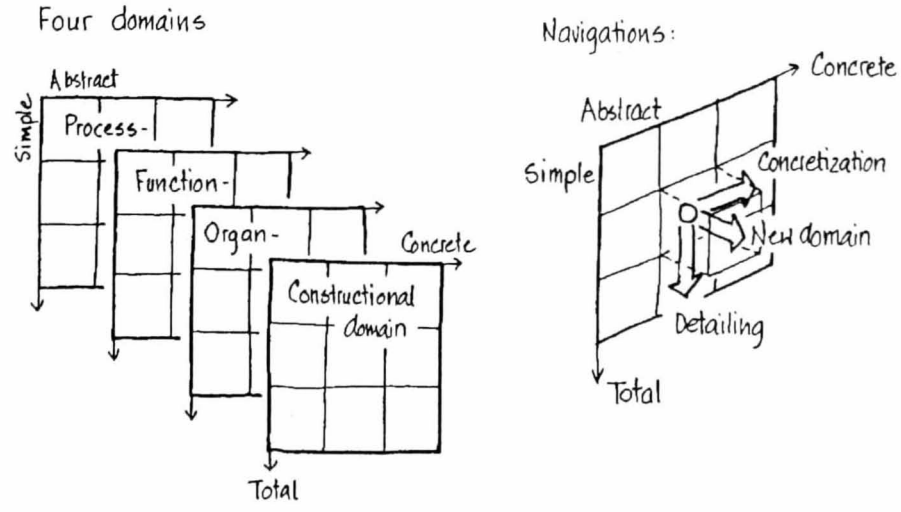


Figure 3.4 - Domain based design navigation - [Andreasen 1991a]

Designing as manoeuvring in the domains

Due to the causality between the domains, when designers create an artefact, they actually create four definitions of the artefact's structural relations, related to the four domains. Andreasen argues that in practice only the constructional structure is fully defined to specify the artefact. When handling solution models in one of the domains, design process steps are carried out (Figure 3.4) resulting in the determination of one or more *characteristics* [Andreasen 1991a; Mortensen 1997] directly definable by the designer. These steps cause a progression from *abstract to concrete* (i.e. determination of parameter values), from *undetailed to detailed and complete* (determination of more elements), or as a movement from one domain to another. Choices made in each domain, all contribute to determining the properties of the artefact [Olesen 1992]. These movements reflect the designer's *navigational manoeuvres*, achieved through activities such as *specification decomposition*, *synthesis*, *analysis*, *optimisation*, *documentation* [Mortensen et al. 1994]. Andreasen argues that ideally, design should follow the causal relationships mentioned earlier, but in practice, many other sequences are followed, depending upon the individual designer, experience and type of design task [Mortensen et al. 1994]. Thus, the explanation provided by the domain model differs from a stage based design process, where each domain, is basically regarded as a stage that should be finished before the next stage commences.

3.1.4 A Knowledge Intensive Process

Many models of design fail to characterize the true nature of design as it is carried out by *humans* [Smithers et al. 1990]. Smithers et al. argue that the ability to design is a kind of intelligent behaviour which explicitly makes use of knowledge. This section aims to disclose characteristics of knowledge used during design and presents a model describing *how* it is being processed.

Knowledge types

Knowledge used in design involves *operational* and *substantive* knowledge [Roozenburg et al. 1995]. Operational knowledge includes *design activity* knowledge which is concerned with how to carry out particular design activities and *design process* knowledge concerned with how to organize design activities and how to execute the design process [Zhang 1998]. Zhang explains that substantive knowledge is *'design knowledge'* that concerns the nature of the artefact – for example, what is the intended use, how it works and how it is constructed. Zhang argues that design knowledge consists of *current working knowledge (CWK)* and *domain knowledge (DK).* CWK is the knowledge of the artefact design solution on which the designer is currently working. DK is knowledge of past designs in a domain consisting of *generalized knowledge* and knowledge of specific past design cases.

Knowledge sources

Knowledge in any specialty including design can be found in *private* and *public* sources [Walters et al. 1988]. Public knowledge, which is common to different organizations, includes published definitions, facts, and theories of which textbooks and references in the domain of study are typically composed. Private knowledge consists largely of rules of thumb frequently called heuristics. Human experts generally possess private knowledge that has not found its way into the published literature. Public and private knowledge sources can also be *distributed.* For instance, a thermoplastic component design guideline book is a different public source than a book dealing with ceramic components.

Knowledge dimensions

Three distinct dimensions of knowledge are *depth, breadth* and *extent* [MacCallum et al. 1987]. The *depth* dimension is concerned with the range of knowledge in which generality increases with depth. The *breadth* dimension refers to the variety of different aspects that can be taken of a design concept. The *extent* dimension is concerned with knowledge about the object of the design process. It is concerned with the artefact model itself, but also with the variety of models that exist during design, the design problem and its specification.

Knowledge permanence & certainty

Knowledge employed during design varies in *permanence* and certainty [Walters et al. 1988; Tang et al. 1997]. Knowledge can be (i) *permanent,* it will not change with time e.g. knowledge of how to determine a sphere's volume; (ii) *static,* this does not change frequently but *likely* to change e.g. a safety factor; (iii) *dynamic,* this concerns knowledge that changes with time or context e.g. material costs. Also, in reality, there is a great deal of *uncertainty* because not all facts, rules making up knowledge are absolutely true or false.

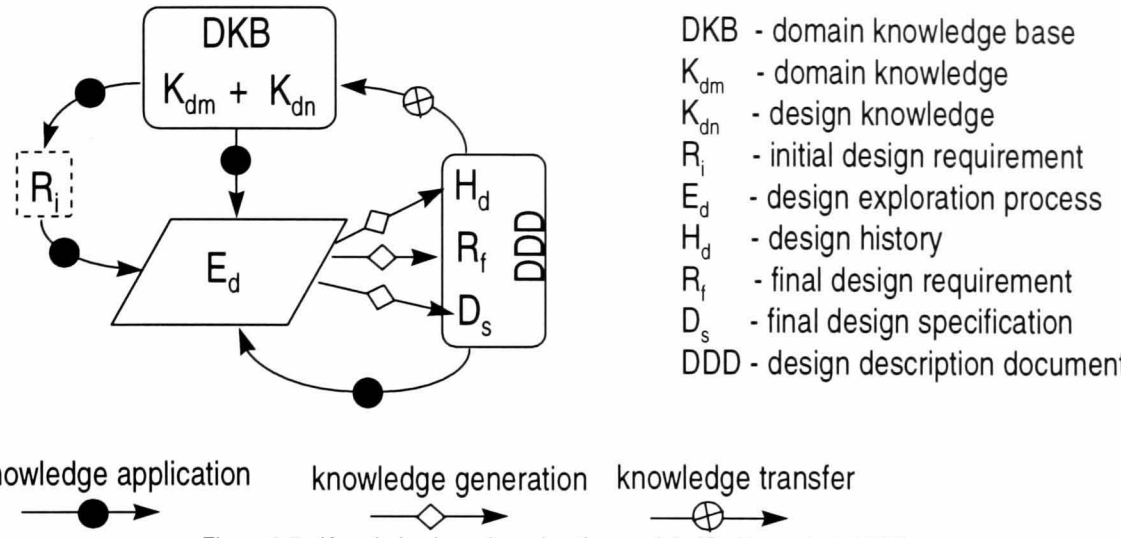


Figure 3.5 - Knowledge based, exploration model - [Smithers et al. 1990]

Design as a knowledge based exploration activity

To explain *how* knowledge is processed during design, Smithers et al. provide an *exploration-based* model of design (Figure 3.5). The design process (E_d) is considered as the *exploration* of a space of possible design solutions (SPD). This exploration activity is considered to be a search connected by intuitive leaps, analytical assessments, synthesis, simulations, prototypes, decisions, choices and expert judgements - these constituting a history (H_d) of the design process. Similar to other models, design starts from an initial design requirement (R_i), usually incomplete, inconsistent and ambiguous, which *evolve* as design proceeds. To facilitate the transformation from the R_i to the final design specification (D_s), a domain knowledge base (DKB) is employed. The DKB consists of *domain knowledge* (K_{dm}) and *design knowledge* (K_{dn}). In this model, K_{dm} partially defines the *SPD* to be explored whilst K_{dn} is knowledge about *how* the space can be explored. As design proceeds, designers *acquire* more knowledge of the nature of the design space, this allowing them to discover incompleteness and inconsistencies in the initial requirement (R_i). As a result, R_i evolves to a final design requirement description (R_f), which reflects the current state of the designer's knowledge about the problem. This exploration is continued until a solution in the SPD is found that satisfies the *evolved* R_f. This results in a final, complete and consistent, requirement description (R_f) and an associated design specification

(D_s) which is consistent with it. Smithers et al. argue that R_f and D_s are different statements about the same thing i.e. *problem description terms* and *solution domain terms*. Thus these two aspects of design are tightly interactive in terms of the knowledge *used* and *generated*. Therefore, they cannot be pursued either separately or sequentially due to the co-evolutionary nature of the problem and solution space. The design process ends with the generation of a design description document (DDD) consisting of R_f, D_s and H_d, that collectively represent the knowledge *used* and *generated* during design. This model explicitly shows that design involves the *generation, transfer* and *application* of knowledge. Thus, carrying out design tasks does not just result in solutions to particular design problems but also in greater knowledge and understanding of the design problem and a more experienced designer.

3.1.5 Design Process Characteristics

Based on the diverse perspectives of design presented, this section now discloses characteristics of a *traditional* design process.

Integrated stages and activities

The transformation of a design problem into a solution and hence the expansion of knowledge about the artefact from one state to another is achieved through a number of *activities* that *re-occur* [Blessing 1994; Roozenburg et al. 1995] during the different *design stages* (Figure 3.6).

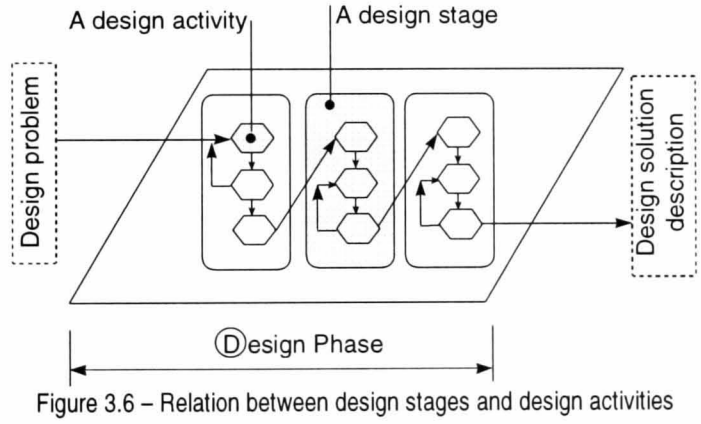


Figure 3.6 – Relation between design stages and design activities

For instance, conceptual design involves problem definition, synthesis of (partial) solutions, evaluation and problem redefinition [Chakrabarti et al. 1991]. Blessing thus provides a problem-oriented, process-based model, which explicitly integrates design stages and design activities (Figure 3.7).

Stages	Activities		
	Generate	Evaluate	Select
Problem definition			
Conceptual design		Steps	
Detail design			

Figure 3.7 - Design matrix integrating stages and activities - adopted from [Blessing 1994]

Design reoccurs for different system levels

What is not explicitly evident from the models presented earlier is that design re-occurs for different artefact system levels [van den Kroonenberg 1987; Blessing 1994]. For example even the design of a component for a specific function, requires component concepts to be generated, evaluated and a solution concept selected, before being worked out in more detail. Thus, design re-occurs for different artefact system levels (Figure 3.8).

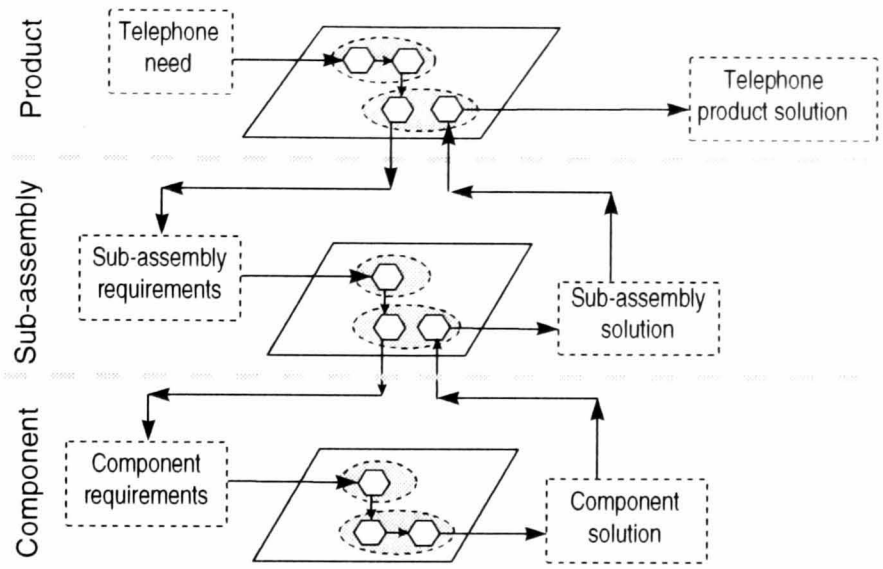


Figure 3.8 - Reoccurrence of design process for different artefact system levels

Process is driven by use-phase requirements

The different models reflect that design ends with a *final* solution description, viewed from a constructional perspective, describing a means by which the artefact's desired function in the *use phase* can be attained. In fact design can be considered as a transformation from function to form[1] [Andreasen 1991a; Roozenburg et al. 1995]. This reflects that design is traditionally *use-phase driven.* Thus the end result in traditional design, is an artefact solution description - there is *no* accompanying artefact life-phases solution.

Process explicitly evolves artefact knowledge

The artefact states evolving from *abstract* and *undetailed* to *concrete* and *detailed* during design reflect that there is a *current working knowledge (CWK) expansion* about the artefact system. In traditional design, this CWK expansion is not *explicitly* accompanied by an expansion in knowledge about the *life-phase systems* that will interact with the artefact during its life.

Designers manipulate basic characteristics during synthesis

Five *basic properties* (structure, form, material, dimensions, surface quality) distinguish themselves from other characteristics by the fact that together, they completely describe a mechanical artefact [Tjalve 1979]. These are therefore the *characteristics* that are manipulated by the designer [Olesen 1992; Mortensen 1995] when generating a design solution that satisfies a set of *desired properties.* All other properties, termed *behavioural* [Mortensen

[1] An artefact has 'geometrical' and 'physico-chemical' form [Roozenburg et al. 1995]

1997], such as cost, weight, appearance, fabrication and ergonomics are determined when these basic characteristics have been defined [Olesen 1992].

Synthesis involves PDE reuse

A large proportion of design work consists of the reuse and fitting of known solutions to new conditions [Andreasen 1992; Blessing 1994]. An observation of typical artefacts reveals PDEs in their structure (see Figure 2.2). This research therefore argues that *synthesis* involves the manipulation of *reusable* PDEs. Such *solution reuse* re-occurs during synthesis which takes place at different artefact system levels and from different domain viewpoints. For instance, designing from a functional perspective involves reusing existing *function means* such as 'petrol engine' to provide power. Similarly, component synthesis would involve the reuse of *component elements* such as form features, assembly features and materials.

Implicit reuse of consequence knowledge

A significance of reusing solutions is that designers can reuse associated consequence knowledge, such as costs or risks [Andreasen 1992]:

> "It is very important to control the design activity in such a way, that a maximum amount of the work becomes reuse in widest sense, because this leads to low cost, low risk, known quality, etc."

However, based on examples presented in Chapter 2, this research argues that although solutions being reused implicitly have associated LCC knowledge, the designer *may* or *may not* be explicitly processing knowledge about such consequences.

Space of alternative solutions

During different stages and for different domains, alternative solutions may satisfy the design problem being tackled. For example, during detail design, alternative component materials and form can satisfy the problem. Such a solution space gives rise to 'Design Degrees of Freedom' [Andreasen 1991a], as illustrated in Figure 3.9.

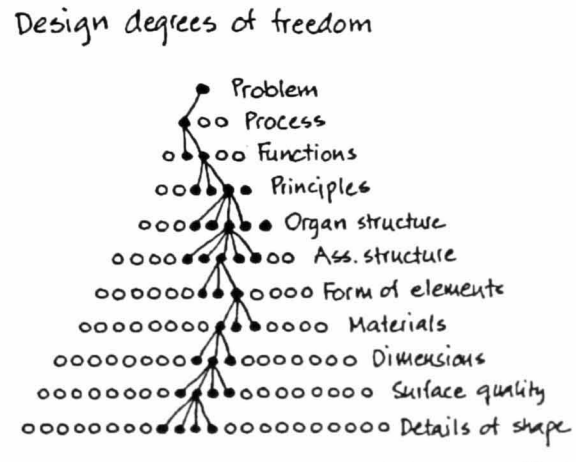


Figure 3.9 – Design freedom caused by solution space - [Andreasen 1991a]

Alternatives make design decision intensive

Alternatives exist when designing in different stages (e.g. different solution concepts) and when designing from different domain perspectives (e.g. alternative function-means). For design to progress, decisions between alternatives in the solution space therefore need to be made [Blessing 1994; Pahl et al. 1996], irrespective of the design stage or synthesis domain.

Factors influencing design decisions

A number of *factors* such as 'design engineer' and 'technical information' play a crucial role in design (Figure 3.10). This is due to their influence on design characteristics such as the 'quality of result' and 'cost of designing' [Hubka 1985; Andreasen 1991; Blessing 1994]. Through the knowledge processing model of design [Smithers et al. 1990], it can be appreciated that besides the 'design engineer', other humans can also influence design. For instance, other participants can be involved in design such as a *customer* who provides the 'need' and *producers* (that realize the artefact) that provide a valuable source of knowledge for evaluation [Jin et al. 1998].

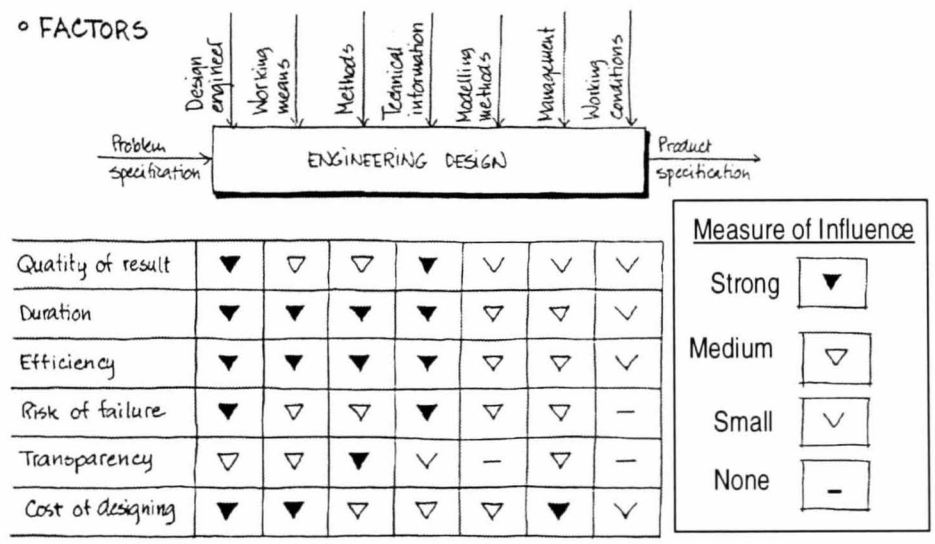


Figure 3.10 - Factors influencing design - adopted from [Hubka 1985]

Solution & problem co-evolve

Unlike the impression portrayed by stage models, in reality, designers do not usually have a complete problem description before commencing conceptual solution synthesis [Maher et al. 1996]. Rather, during design, *knowledge* of the solution as well as the problem co-evolves [Smithers et al. 1990; Roozenburg et al. 1995]. Hence, as argued by Cross [Cross 1994], designers need to *explore* and develop the problem and solution together. Thus, an important activity not explicitly reflected in the basic design cycle is the *exploration* of the space of possible design solutions [Smithers et al. 1990] as this helps generate knowledge about the problem at hand. As in the context of DFΣX, designers have to cater for artefact life problems co-evolving with the solution, the role of exploration will be discussed in section 3.2.

3.2 The Role of Exploration in Life-oriented Design

A knowledge generation activity

The *open-ended nature* [Dym 1994; Roozenburg et al. 1995] of ill-defined problems requires the proposal of solutions as a means of *generating* knowledge to help *understand* the problem. Exploring provisional solutions is a forceful aid to gaining insight into the true nature of the problem [Roozenburg et al. 1995]. Kerr [Kerr 1993] in fact argues that exploration is a significant activity in design. Kerr explains that exploration promotes the *creation* of knowledge potentially useful for synthesis and evaluation:

> "Designers can use this [exploration] activity to generate suitable knowledge to assist in the expansion or contraction of design solutions"

Similar arguments are made in [Smithers et al. 1990], that the exploration activity results in the *generation* of knowledge that can be *applied* to evolve both the 'design requirement description' and the 'final design specification'. This section aims to demonstrate that exploring alternatives in the solution space generates LCC knowledge useful for reasoning taking place when PDEs are being selected from the space of options, both when designing from different *domain viewpoints* and at different *artefact system levels*.

Domain viewpoint based exploration

Process solution exploration

Figure 3.11 provides an example of three solution variants by which the *sub-process* of 'uniting tea leaves and water' when designing a tea brewing machine, can be achieved [Hubka et al. 1988]. The different variants require a different sequence of operations, influencing differently the sub-process time in the use phase and the required structural configuration.

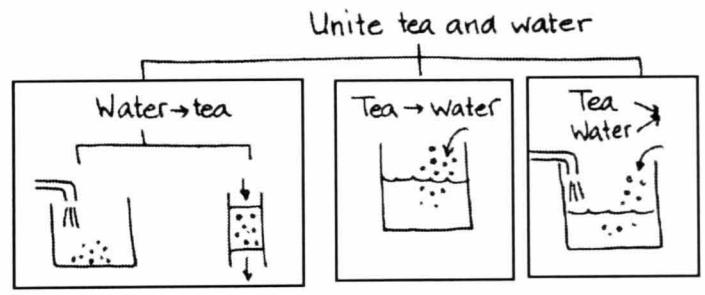


Figure 3.11 - Space of process solutions - [Hubka et al. 1988]

Constructional solution exploration

Some alternative arrangements of the three main constructional elements (heating container, brewing container and serving container) of the tea brewing machine example found in [Hubka et al. 1988] are shown in Figure 3.12. For instance, variant '1' has problems with realizing the tubing, whilst variant '10' has a lower height than variant '2', this influencing differently the required packaging configuration and hence artefact handling costs during distribution.

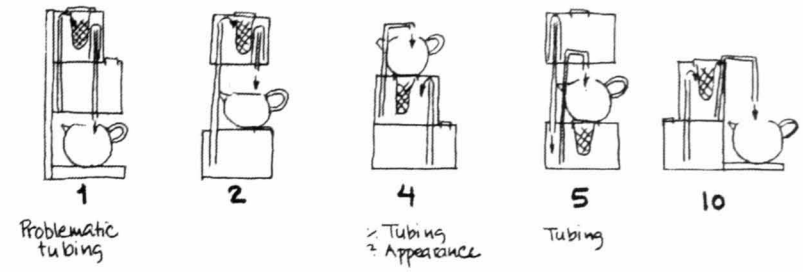


Figure 3.12 - Space of constructional solutions - [Hubka et al. 1988]

Organ solution exploration

In an example concerning the design of a tumble dryer, [Hansen 1997] provides an example of the organ 'burner' used as a means by which to 'heat air'. The tumble dryer exists in variants which employed different organs for the function 'heat air' (Table 3.1). These different organs have different LCCs such as running costs, environmental influences and maintenance frequency.

Table 3. 1 - Space of alternative organs

Function	**Alternative Organs**		
Heat air	Gas fired burner	Electric heater	Steam heater

System level based exploration

Product level exploration

The two excavators in Figure 3.13 have a driver cabin and bucket with different degrees of freedom. This influences the translational and rotational means required, together with their servicing frequency and costs. Also, the two excavators can be employed during the 'use phase', for different ground terrain environments. These are some of the associated LCCs a designer can consider when exploring this *product level* solution space.

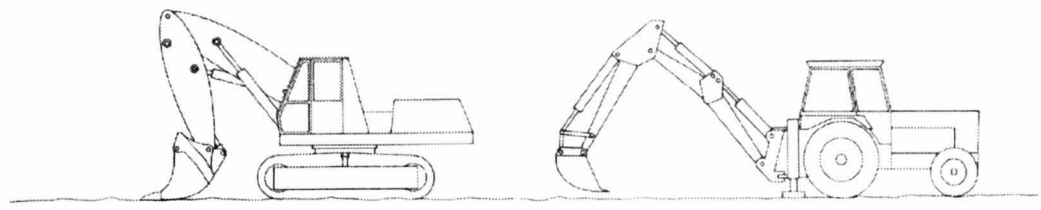


Figure 3.13 – Exploring the product level solution space - adopted from [Tjalve 1979]

Sub-assembly level exploration

The example in Figure 3.14 reflects some constructional structure variants for a *sub-assembly* forming part of an excavator [Tjalve 1979]. The different variants have different configurations and number of linkages, which influences differently assembly in terms of time and cost, during realization, and dis-assembly during the service activity in the use phase.

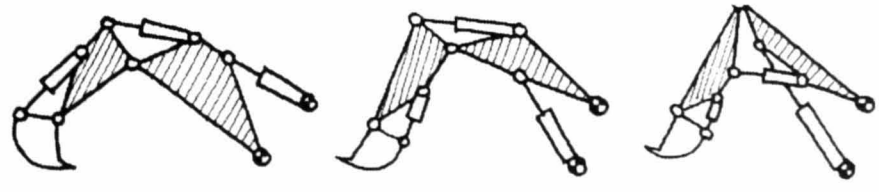


Figure 3.14 – Exploring the assembly level solution space - adopted from [Tjalve 1979]

Component level exploration

Figure 3.15 displays a number of alternative *form* concepts for a fork-joint component found in [Tjalve 1979]. The different solution variants have different LCCs associated. For instance, they withstand different stress levels in the use phase and require different fabrication processes, the latter influencing time and costs during the realization phase.

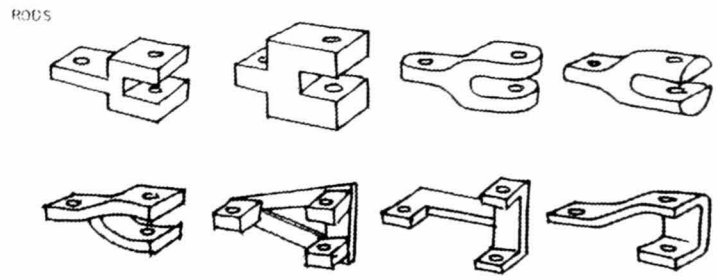


Figure 3.15 - Exploring the component level solution space example- [Tjalve 1979]

Exploration Observations

The examples presented collectively reveal that the exploration activity is useful to explicitly generate and utilize LCC knowledge associated with the different solution options, during design taking place at either from different *solution viewpoints* or at different *artefact system levels*.

3.3 $D_sF\Sigma X$ Characteristics

Based on the traditional design process characteristics disclosed in section 3.1 and the discussion in section 3.2 on the role exploration plays in design, this section argues that a $D_sF\Sigma X$ approach requires:

i. knowledge of multiple life-phase requirements
ii. coping with co-evolving artefact life solution & problem space
iii. designers to foresee life problems co-evolving with synthesis
iv. dynamic updating of life-phase requirements
v. designers to foresee artefact life interactions
vi. concurrent synthesis
vii. designers to manipulate basic life-phase system characteristics
viii. concurrent modelling
ix. artefact life exploration
x. the exploration of abstract & undetailed solutions
xi. synthesis based provident thinking
xii. a large amount of distributed LCC knowledge.

(i) Requires knowledge of multiple life-phase requirements

Transformation driven by multiple life-phase requirements

In the context of DFΣX , a designer's aim is to generate a solution that not only functions as desired during the 'use phase' but that satisfies a *host* of artefact life requirements. Unlike traditional design, DFΣX therefore requires designers to be knowledgeable of *multiple* life-phase requirements [Andreasen et al. 1997].

(ii) Requires coping with co-evolving artefact life solution & problem space

Co-evolution of artefact solution & life problem space

The evolution of the artefact solution taking place during design *co-evolves* the artefact life problem space. For example, a typical sub-problem in the design of an artefact is that of finding and defining a suitable means (a PDE) for joining two components together. There are a number of solution variants to how this can be achieved, such as, using 'fasteners' or a joint such as a 'weld'. Selecting for instance a fastener *evolves* the artefact solution and the problem - a hole with all its parameters *now* needs to be defined. In addition, this sub-solution concurrently evolves the realization phase problem – suitable 'hole generating' processes have to be identified, alternatives evaluated and eventually one selected. Therefore, a characteristic of DFΣX is that designers have to cope with a co-evolving artefact solution and artefact life problem space.

(iii) Requires designers to foresee life problems co-evolving with synthesis

Influence of synthesis output

The synthesis activity is associated with the *externalization* of a solution idea (from the designer's mind) whether a new or older one is being reused, this *described* in some form e.g. verbal, a sketch, or a model [Roozenburg et al. 1995]. It is this synthesis output, frequently a PDE, which *if* eventually *selected*, is passed onto the other design stages and eventually the remaining artefact life-phases. It is this output which therefore drives the co-evolution of the solution and problem space.

Provident thinking

DFΣX therefore requires designers to *foresee* the artefact's life problem space co-evolving with such a selected PDE. As argued by Olesen [Olesen 1992], 'foreseeing' which he terms *provident thinking*, is an important element of concurrence. As defined by Olesen, providence means:

> "Taking into account aspects of the total life that are fixed or determined during design'"

(iv) Requires dynamic updating of life-phase requirements

Evaluation difficulties due to dynamic multiple life-phase requirements

The co-evolutionary nature of the artefact solution and artefact life problem space makes the comparison of the expected behaviour of the proposed solution with the artefact life requirements, *dynamic* and thus difficult to handle. This is because the designer needs to know the currently known total life requirements and also how, due to the phenomena of propagation effects, the solution being evaluated is itself influencing multiple life-phase requirements. This requires the designers' knowledge of multiple life-phase requirements to be *dynamically* updated as the artefact solution evolves. Since human beings have a difficulty in coping with complexity [Kerr 1992], it is not difficult to visualize that in DFΣX, designers have a difficulty in making an evaluation judgment when the criteria are constantly changing.

(v) Requires designers to foresee artefact life interactions

Need to foresee interactions with artefact life phase systems

As argued in Chapter 2, interactions of an artefact with different life-phase systems can give rise to a number of unintended LCCs. DFΣX therefore requires that designers *foresee* what life-phase systems will be met during the life of an artefact and that they also *foresee* the outcome (consequences) of such interactions during design. For example, the sub-solution to assemble two components together with *fasteners* requires the selection of an 'assembly system'. Possible alternatives are a 2 degree of freedom pick-and-place device or a 3 degree of freedom robot equipped with a vision system (Figure 3.16). The assembly system eventually employed will have a number of specific requirements that need to be satisfied, knowledge of which would guide designers *during* artefact synthesis. If the assembly system (eg. 2 d.o.f. pick and place device) to be used is not known, then, due to this 'ignorance' the designer may position and/or orientate the fasteners in a non-vertical way. The interaction of the resultant artefact with this assembly system would give rise to unnecessary assembly costs and time delays.

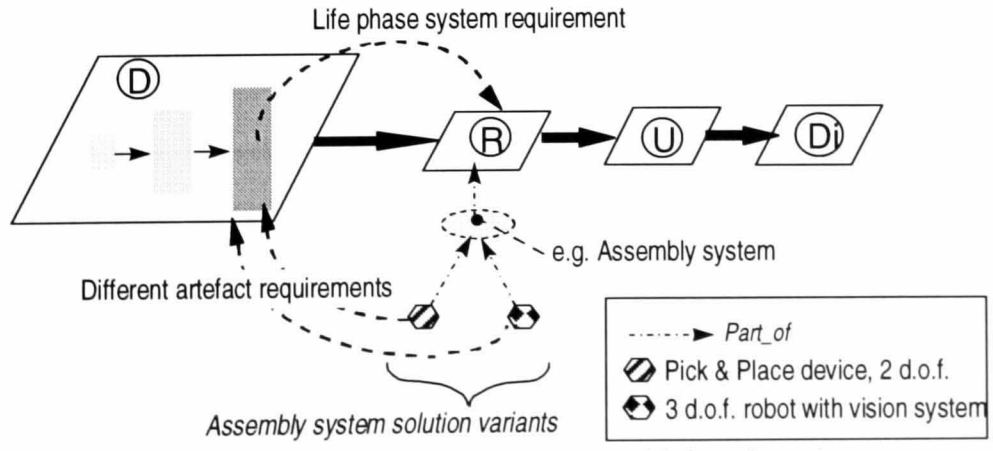


Figure 3.16 - Foreseeing life-phase systems and their requirements

Harmonization

Thus DFΣX requires designers to harmonize their artefact solution to a number of life-phase systems forming part of an imagined life. Of relevance is that Andreasen et al. [Andreasen et al. 1996b] state that :

> "Designing is closely linked to foreseeing product life phases primarily the 'use' phases, but also establishment, maintenance and liquidation. [....]. But in any case the designer's task is to fit the product to an imagined life scenario."

(vi) Requires concurrent synthesis

Foreseeing interactions needs concurrent artefact and life-phase system synthesis

An implication arising from the need to foresee artefact life interactions between an artefact and different life-phase systems, is that for DFΣX, designers need to concurrently synthesize the life-phase systems and the artefact. Concurrent synthesis would cause an expansion in knowledge about the *artefact life*, knowledge, which as demonstrated through the previous example, is essential in the context of DFΣX.

(vii) Requires designers to manipulate basic life-phase system characteristics

LCPEs

An implication of (vi) is that DFΣX requires designers to manipulate basic life-phase system characteristics in addition to basic artefact characteristics that completely describe an artefact. As argued in Chapter 2, a life-phase can be viewed as a transformation system decomposable into systems of a finer resolution. The life phases forming part of an artefact's life, involve the *re-use* of well-known technical systems that realize the relevant transformation effects, such as a milling system. These systems can be decomposed further into sub-systems, such as the workpiece holding sub-system, giving rise to a life-phase compositional model (see Figure 2.9b), consisting of what are termed in this thesis *life cycle phase elements (LCPE).* Life-phase solution synthesis thus requires LCPEs to be manipulated and defined.

(viii) Requires concurrent modelling

Artefact and life-phase systems concurrent modelling

Concurrent synthesis results in the generation of both artefact and life-phase systems solution descriptions. In traditional design, designers evolve and manipulate artefact solution models. Modeling, allows designers to *infer* properties that cannot be defined directly by the designer [Mortensen 1995]. Thus, building and exploring artefact life models is beneficial to a DFΣX approach. Morup [Morup 1993] states:

> "...problems can be avoided by using total life models that show the interplay between the product, the customers, and company activities through the product's life. ...total life models can reveal the dispositional relationships within the company that are often subtle and go unrecognized, and the exploration of these can lead to competitive advantage."

A requirement of DFΣX therefore is that designers cope with *concurrently* 'evolving and manipulating an artefact model' *and* 'evolving and manipulating artefact life phase models'. Designers therefore have to evolve, handle and evaluate more than one solution model at a time.

(ix) Requires artefact life exploration

Exploration generates knowledge suitable for DFΣX

In DFΣX, a solution *selected* from the many PDE alternatives must satisfy a host of total life requirements. At the same time, as argued in (ii), the selected PDE may co-evolve its own 'life requirements' that can only become apparent and hence influential when the PDE is defined. In this context, *exploring* the use of alternative PDEs generates knowledge relevant to assisting in the selection of solutions that cater for a host of total life requirements, this is useful when evaluating between feasible PDEs. As argued in [Roozenburg et al. 1995], one characteristic of human designers is that they have a mind, in which mental processes taking place, can be more or less influenced. Artefact life exploration is therefore a significant and necessary activity in DFΣX since LCC knowledge generated can *influence* the selection of solutions being defined. Further, the constraints of product development deadlines in the real world limit the search and exploration of the solution space. Time and resources for most projects do not permit the development of appropriate, quantitative information. As a result, important decisions are based on primarily qualitative information [Blessing 1994]. Therefore, to be effective, artefact life exploration needs to be rapid.

(x) Requires the exploration of abstract & undetailed solutions

Handling abstract & undetailed solutions

Chapter 2 has shown that LCCs are generated by the *final* design solution emerging from the design phase. However, as explained in this chapter, artefact solutions evolve from one design stage to another (Figure 3.17).

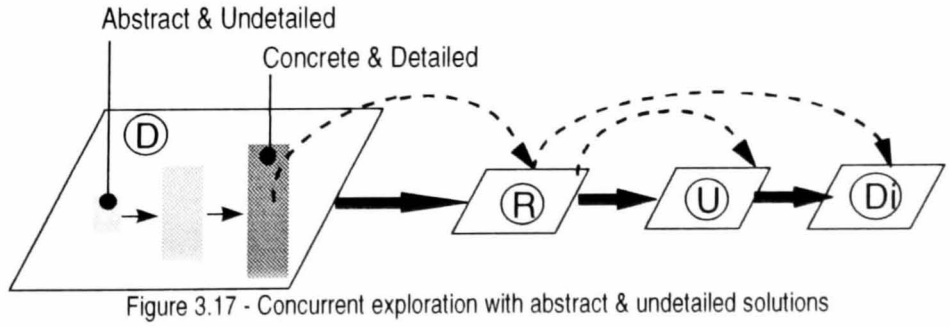


Figure 3.17 - Concurrent exploration with abstract & undetailed solutions

Then, the final solution will have a *trait* of the abstract solutions generated in the early conceptual design stage, this also reflected by Pugh [Pugh 1991]. At the same time, Olesen [Olesen 1995] argues that:

> "A concept consists of a number of chosen conceptual solutions for both the product and the product life systems".

Considering artefact life issues *late* in the design process is thus less receptive as even a simple change could lead to a major redesign [Ishii 1991]. Hence, bringing knowledge about the artefact life into an early design stage is critical [Tomiyama 1996]. Therefore, artefact life exploration needs to take place from the early design stages. However, exploring abstract and undetailed solutions, in order to reveal artefact LCCs that reflect 'details' of the real world, presents a difficulty - little is yet known about the solution and hence its impact on different life-phases.

(xi) Requires synthesis based provident thinking

DFΣX dimensions

From the arguments made so far, DFΣX can take place by designers, either foreseeing and catering for artefact life issues during the *synthesis* of a conceptual design solution, or, during the *analysis* of a candidate conceptual design solution. This gives rise to the DFX dimensions shown in Figure 3.18.

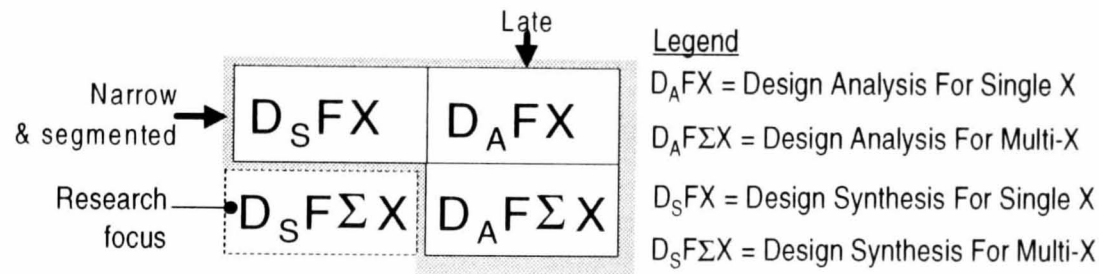


Figure 3.18 - DFX dimensions

$D_sF\Sigma X$

As argued by French [French 1985], the conceptual design stage is where most important design decisions are made and therefore the stage where there is most scope for making improvements. As final solutions have a trait of abstract solutions generated in the early stages, this research argues that to be beneficial, foreseeing LCCs should take place as early as possible in the design process. Therefore, the mode of DFΣX approach should be 'conceptual design synthesis for multi-x' ($D_sF\Sigma X$). Thus, $D_sF\Sigma X$ requires designers to engage in provident thinking *during* synthesis, when the solution is still *incomplete* and *undetailed*, with the artefact life problem still ill-defined.

(xii) Requires a large amount of distributed LCC knowledge

Whether a problem is considered as variant or original depends upon the designer's level of expertise [Mills 1993]. Therefore, designers are less knowledgeable about artefact LCCs when facing design scenarios having a higher degree of originality to that with which they are familiar.

Designers need to utilize a vast amount of distributed LCC knowledge

At the same time, due to (v), this research argues that designers need to *possess* and *utilize* a wide breadth of knowledge related to different life-cycle phases, this traditionally not in their domain, to enable them to reveal interactions and hence possible LCCs. For instance, with respect to design for economic manufacture, it is argued that [Swift 1987]:

> "In short, he [the designer] needs to have expertise in a wide range of fields, including the specialized topics of manufacturing engineering...".

This is even more the case with DFΣX. However, due to the traditional formal training received, designers do not generally possess a wide breadth of public LCC knowledge. Further, as argued in Chapter 2, due to an artefact's life chronological order, designers do not generally *acquire* experiential knowledge concerning LCCs resulting from artefacts interacting with different life phase systems. The cost and time of dis-assembling a component with certain assembly features during the service activity in organization 'ABC', is private experiential LCC knowledge possessed by a service engineer. Thus, such experiential LCC knowledge is *acquired* and *distributed* (Figure 3.19) amongst various human artefact life-actors (e.g. machining operators).

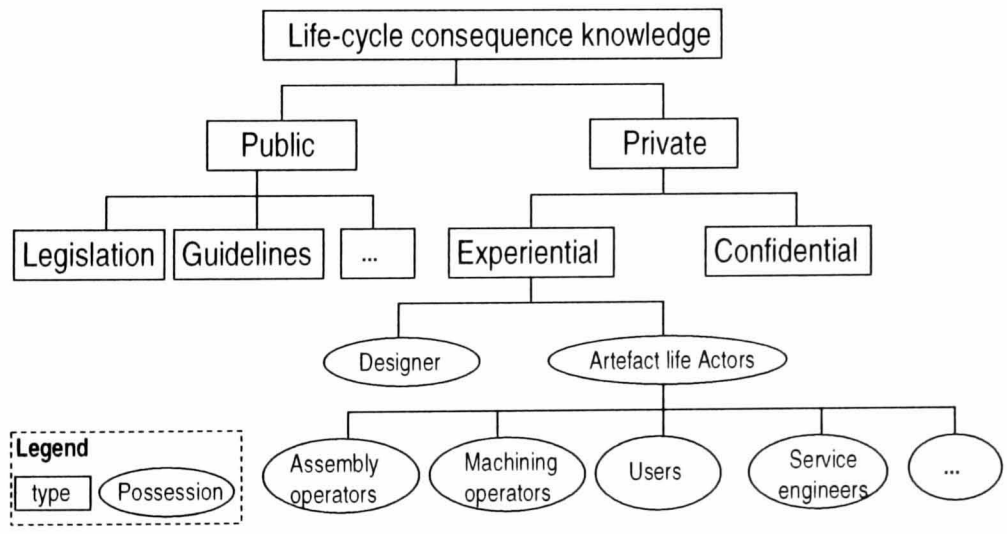


Figure 3.19 - Distribution & possession of LCC knowledge

Public and private LCC knowledge sources can be *distributed*, internally and externally to an organization making its explicit use during design even more difficult. Further, an individual's *limited* knowledge base is known to result in low quality decision making [Duffy et al. 1995]. Thus, due to a lack of LCC knowledge, design decision making takes a *narrow* and *segmented* view, this affecting the quality of the solution result with respect to artefact life issues. Therefore a $D_sF\Sigma X$ approach requires a vast amount of distributed LCC knowledge to be *acquired*, *readily available* and *easy to access* in order for it to be explicitly *utilized* during design synthesis.

3.4 Chapter Conclusions

Design process characteristics

As disclosed in this chapter, during design, a problem is transformed into a solution description via a number of *design stages*, through the execution of re-occuring *activities*. Key activities are *problem analysis, solution synthesis, solution analysis* and *solution evaluation.* The expansion of knowledge about the artefact taking place during design is mainly *driven* from a 'use phase requirements' point of view. As discussed, knowledge of the solution as well as the problem co-evolve. Thus an equally important activity is that of *exploring* provisional solutions to gain an insight into the true nature of the problem. Design is thus knowledge intensive– knowledge is *used*, *generated* and *acquired*. The knowledge based, exploration model by Smithers & Troxel discloses *how* knowledge is processed whilst the basic design cycle by Roozenburg & Eekels describes *why* knowledge is processed. On the other hand, Andreasen's domain model discloses, from a generic point of view, *what* domain knowledge is processed. Further, irrespective of the design stage and domain, design is a decision intensive process. Artefact solution evolution requires making decisions about the *basic characteristics* manipulated by designers. A lack of appropriate knowledge is a reason why such design decisions can result in unintended LCCs.

$D_sF\Sigma X$ characteristics

A distinguishing characteristic of $D_sF\Sigma X$ is that designers require knowledge of 'multiple life-phase requirements', the latter as argued in this Chapter co-evolving with the 'artefact solution'. $D_sF\Sigma X$ therefore requires that designers *foresee* 'what' life-phase systems the artefact will interact with during its life, to enable them to also *foresee* the consequences of such interactions. Foreseeing such interaction consequences requires designers to possess and utilize a vast amount of LCC knowledge that is distributed amongst various artefact life-actors. Moreover, to be beneficial, designers need to foresee LCCs from early in the design process, where decisions have a striking effect. Early artefact life exploration is therefore significant in $D_sF\Sigma X$ since LCC knowledge *generated* can be *utilized* to support the selection of solutions that are 'life-oriented'. Thus, $D_sF\Sigma X$ requires the concurrent synthesis, manipulation and exploration of both 'artefact' and 'artefact life-phase' models. As an implication, for $D_sF\Sigma X$, designers also need to manipulate basic *life-phase system* characteristics termed in this research as life-cycle phase elements (LCPEs).

Need to exploit artefact life providence

Supporting designers in foreseeing artefact life interactions and their propagation during design, directly supports a $D_sF\Sigma X$ approach – this however requires a vast amount of distributed LCC knowledge to be *acquired*, *readily available* and *easy to access* in order for it to be explicitly *utilized* during design. This chapter therefore concludes that a key characteristic of $D_sF\Sigma X$ is that designers should engage in *artefact life* synthesis and exploration to explicitly acquire knowledge of artefact life interactions. In order to identify the strengths and limitations of how designers currently acquire knowledge of artefact life interactions during design, the next chapter presents a review of available means supporting life-oriented design from the perspective of 'providence'.

chapter 4

4.0 A Review of Means Supporting Providence

Scope

Arguments have been made in Chapter 3 that supporting providence, directly supports a $D_sF\Sigma X$ approach. For this purpose, this chapter will present a review of means supporting life-oriented design from the perspective of 'providence', in order to identify how effectively designers are being supported in foreseeing artefact life interactions during design. Section 4.1 introduces the criteria used to critically review means by which providence is *indirectly* or *directly* supported, these discussed in sections 4.2 and 4.3 respectively. Section 4.4 discusses the current state of providence support. Using the identified limitations, the conclusion in section 4.5 establishes the need for a means providing improved support to $D_sF\Sigma X$ from the perspective of providence.

4.1 Review Classification And Criteria

Providence means

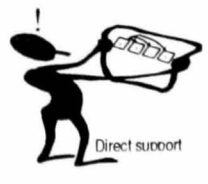
In order to avoid confusion about terminology such as *tools, approaches* and *methods,* this evident from the work reported in [Araujo et al. 1996], a *providence means* is being defined for the purposes of this research as:

> *"A means which indirectly or directly aids a designer to take into consideration artefact life issues that are being fixed or influenced during design."*

Means classification

where, by a *means* is understood 'that by which a result is brought about' [Allen 1990]. There are a number of means by which providence is currently being supported in life-oriented design. This review distinguishes between means that support providence *indirectly* or *directly* :

- means providing *indirect providence support* are those, which force or help motivate designers to consider *revealing* knowledge of relationships between an artefact and life-phase issues;



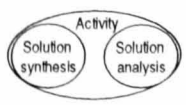
- means *directly supporting* providence are those, which *explicitly* provide designers with codified knowledge that relates an artefact solution and life-phase issues.

Other state-of-the-art reviews

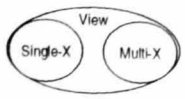
Other reviews related to *means* supporting the consideration of artefact life issues can be found in the literature. For instance 'Design for Manufacturing and the Life-Cycle' are reviewed in [Finger et al. 1989a]. It concludes on the need of analysis tools supporting the early stages of design when critical decisions are made based on qualitative information. Ishii [Ishii 1995] identifies significant research issues related to developing an integrated life-cycle design tool, namely *design representation and life-cycle evaluation measures*. A review of life cycle engineering from an 'environmental' point of view is found in [Alting et al. 1995]. This provides a comprehensive overview of *life cycle assessment* (LCA) tools and methods. Methods such as cross-functional teams, good/bad examples and feature-based evaluation for assessing *producibility* are discussed in [Subramaniam et al. 1998]. A survey of 'Automated Manufacturability Analysis' is found in [Gupta et al. 1997]. This latter review concludes that such software tools vary significantly in terms of (a) their underlying *approach* which could be based on the identification of infeasible manufacturability attributes directly from the design description or indirectly from a manufacturing plan; (b) *manufacturability measures* used by the tools (e.g. abstract quantitative versus real 'time and cost' estimates); (c) *level of automation*: i.e. the amount of designer interaction involved and (d) the *amount and type of feedback information* provided such as redesign suggestions. Computer-aided simultaneous engineering systems are reviewed in [Molina et al. 1995]. This review discusses decision support systems developed to consider product life cycle concerns. As exposed by Molina et al., such systems can be either *stand-alone* tools that allow different aspects of the life cycle to be considered but do not support teamwork, or as *'integrated environments'* that support simultaneous engineering teamwork.

Review criteria

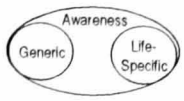
The above reviews provide an excellent cross-section of research related to means supporting life-oriented design. However, it is essential for this thesis to complement these with a review that utilizes key $D_sF\Sigma X$ characteristics disclosed in section 3.3, as a reference point from which to assess how designers are being supported from the perspective of *providence*. This chapter thus focuses on reviewing *providence means* in terms of:



- *timing:* whether providence takes place during *early design synthesis*, with an incomplete and imprecise solution or whether providence is during *solution analysis*, after a candidate solution has already been generated;



- *life-span view*: whether awareness of LCC is *simultaneously* across *multiple* life-phases, or only for a single life-phase;



- *awareness type:* whether artefact life knowledge provided is 'generic' or 'life specific', since the same artefact solution can be exposed to *different* lives thus encountering different life-phase systems.

4.2 Means Indirectly Supporting Providence

Four major *means*, which are considered relevant to *indirectly* supporting providence as they allow knowledge of artefact LCCs to be revealed and utlized during design, are *Teams, Quality Function Deployment, Failure Modes & Effects Analysis* and *Rapid Prototyping*.

4.2.1 Team Based Design Approach

Teams provide a body of knowledge

One way of revealing artefact life consequence knowledge to be utilized during design is through a team based design approach. A team provides the mechanism of bringing together the knowledge possessed by all the life cycle experts [Ishii 1991] (fabrication, assembly, servicing etc.) to the same place at the time design decisions are being made [Finger et al. 1989a].

Team based Approach Strengths

Teams break up the 'walls'

Providing a candidate design solution to such a synergy of different experts allows potential life-cycle problems to be *uncovered*. This gives positive results because the relevant gain and victim areas are represented in the team [Olesen 1992]. This is the basis through which teams support and exploit providence during design, unlike with a traditional 'over-the-wall' [Hird 1993] artefact development approach (see Figure 4.1).

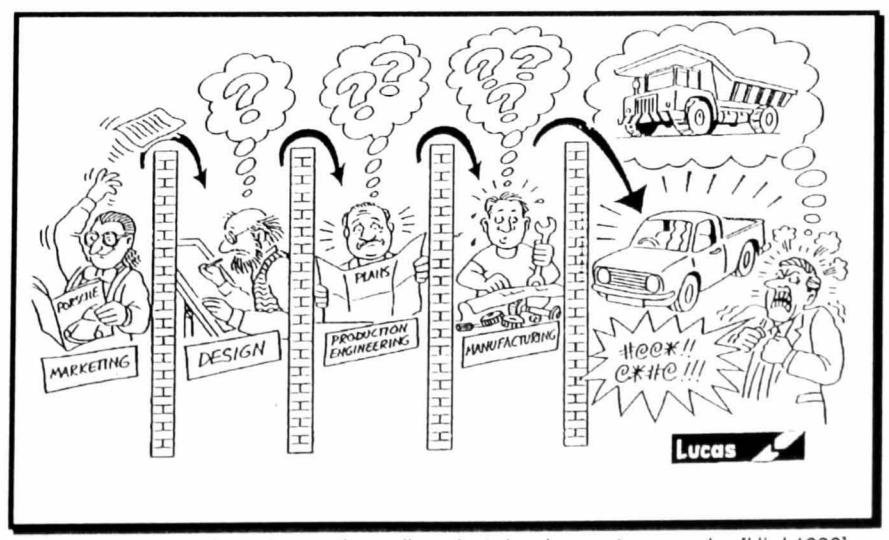


Figure 4.1 - A traditional, over-the-wall product development approach - [Hird 1993]

Teams support multiple life-phase views

It is possible to foresee *multiple* artefact life *specific* consequences during solution *synthesis*, but this depends on *all* team members being *continuously* present during design. Thus, with the *right* multifunctional team using a disciplined approach, good decisions can be made [Clausing 1994].

Design changes reduced with teams

The resulting *information exchange* and *group decision making* with a team reduces the *number of design changes* needed, thus moving design changes earlier in the process [Askin et al. 1994]. This substantially reduces costs and development cycle time by avoiding wasting effort [Edwards 1997].

Team based Approach Limitations

Life-span awareness influenced by personality

Dominating personalities [Edwards 1997] influence teamworking. Under such situations, providence results in a biased assessment of various X's [Willemse 1997].

Life-span awareness depends on proximity

For effectiveness, communication between team members needs to be frequent, but this depends on the spatial *proximity* between them [Askin et al. 1994]. However, teams present many logistic and management difficulties [O'Grady et al. 1991; Bowen 1995]. Frequently, team members are found on different sites and in the case of sub-contractors, in different companies [Allen et al. 1990]. Thus, in practice, it is not straightforward to ensure that team members can interact as *often* and easily as required.

Loss of valuable expertise influences providence

Team members *change* or *retire* causing a loss of expertise [Salzberg et al. 1990]. Thus, there is no guarantee that knowledge *acquired* by team members is *shared* and *reused* in future design projects to support providence.

Difficulty of teams to handle propagation effects

Dispositional mechanisms [Olesen 1992] are so complex that they cannot be readily dealt with, even by a really interdepartmental teams [Andreasen et al. 1990]. This means that the effectiveness of a team in handling propagation effects depends on the ability of the 'human' members to use their synergy to foresee such complex dispositional relationships at the right time.

Ineffective providence due to short & infrequent team meetings

Literature reveals that in reality, few meetings [Dym 1994] are held. The result is that team members work individually for long periods of time without actual communication. Thus, a team is not making decisions and assessments about a candidate design solution *continuously* and *collectively*. Rather, individual team members meet other team members during *design review meetings*, which are never *frequent* or *long enough* [Ishii 1991]. This problem has been

explicitly highlighted in discussions made during this research with practicing designers[1] [2] [3]. For example, one practitioner reported that teams basically met with *all* members present, *only* during the launch of the design project and towards the end, before a design is released for production. In between, individuals only approached other *individual* team members and informally.

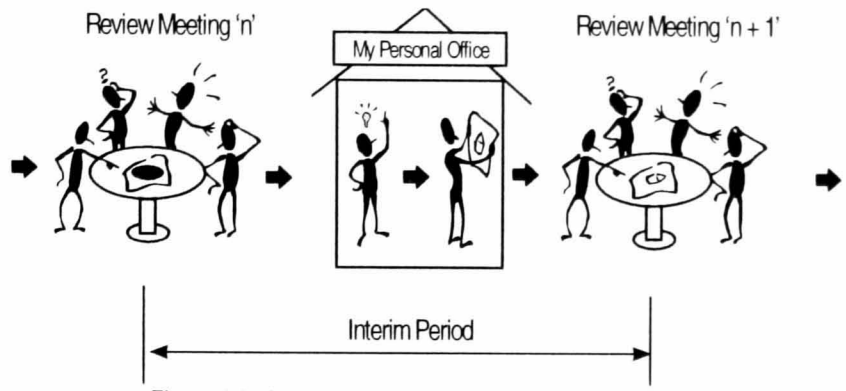


Figure 4.2 - Interim isolation with a team based approach

Interim isolation influences providence life-span awareness

As a result, during the *interim period* (Figure 4.2) between meetings, designers work individually on the evolving candidate solution by making various decisions such as specifying function means, materials and parameter values. This 'interim isolation' violates the concept of a team, as the body of artefact life knowledge is *not* present at the same place and at time decisions are being made. This research argues that this influences the effectiveness of how a team can support providence. One way of overcoming this isolation problem is through a computer based teamwork approach termed, *virtual teams* [Cleetus 1993]. Such a co-location concept can take place at the *same time* or at *different times* [Maher et al. 1997]. Although with virtual teams, the expertise of the different team members is being shared, this does not mean that it will be *reused* in subsequent virtual team sessions to support revealing artefact life issues, especially if team members change or retire.

Limited exploration

As team members meet for design 'review' meetings, a team-based approach essentially employs providence to *criticize* the candidate design in an attempt to optimize it to the artefact life view they represent. Thus, unless truly, group participation is involved *during* solution synthesis, providence is not being exploited for the *exploration* of artefact life opportunities and problems.

---

[1] 1995 - Design team member of a firm based in the Malta which designs and manufactures electro-mechanical devices for automobile companies such as BMW, Mazda, Ford and General Motors.

[2] 1997 – Product design consultant with the Scottish Design Agency.

[3] 1997, Engineering manager, Digital Equipment Scotland, Ltd.

4.2.2 Quality Function Deployment

Use of QFD

Quality Function Deployment (QFD) [Roozenburg et al. 1995] is a method which is aimed at involving from the beginning, various artefact life aspects during decision making. It allows a great deal of information about a particular solution to be assimilated on a chart (Figure 4.3) to enable users to make important comparisons and decisions [Fox 1993; Sivaloganathan et al. 1997]. Hence QFD is performed by interdisciplinary teams [Clausing 1993; Roozenburg et al. 1995; Sivaloganathan et al. 1997].

QFD principles

QFD explicitly focuses on the customer [Roozenburg et al. 1995]. The first task is thus to identify '*what*' the customer's requirements are - written in block '1' of Figure 4.3. From the identified requirements, the designer (or design team) engages in a brainstorming session to identify '*how*' each individual requirement can be met (block '2'). The next QFD step is to *identify relationships* and strengths between the '*whats*' (e.g. durable) and the '*hows*' (e.g. suitable material). One '*what*' can relate to more than one '*how*' requirement. Through QFD, users are therefore systematically motivated to reveal such *what-how* relationships. Identified relationships, are ranked (weak, medium or strong dependency) [Sivaloganathan et al. 1997] and described in block 3. The rank allows designers to focus on areas having a strong dependency.

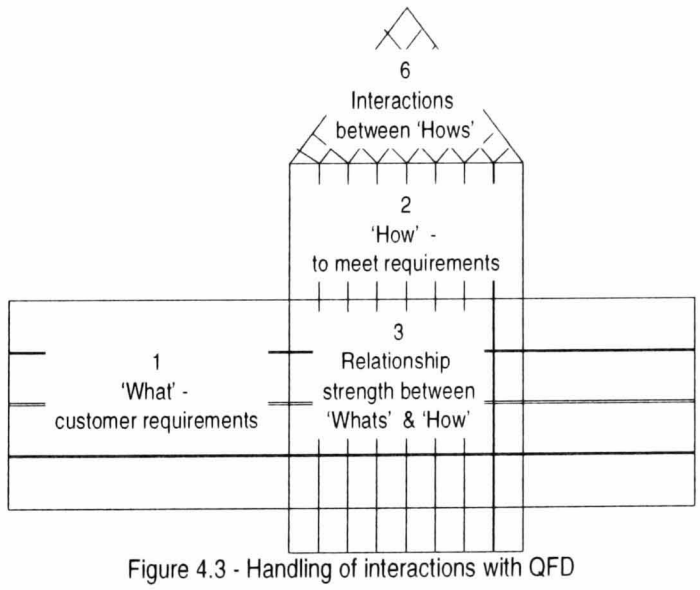


Figure 4.3 - Handling of interactions with QFD

The various 'hows' identified (block '2') can *interact* with each other in a *reinforcing* or *interfering* way. Known positive or negative interactions are assigned by the QFD users in block '6', termed the 'correlation matrix' [Sivaloganathan et al. 1997].

Strengths of QFD

Supports providence as from the task clarification

QFD *motivates* users to systematically consider a host of design issues as from the stage when the design requirements are being specified [Jacobs et al. 1994]. Thus, it motivates providence in order to support the generation of life-oriented design requirements as from the design task clarification stage.

Systematic handling of trade-offs

The QFD matrix provides a method of representing *known* interactions between the various requirements. Documenting interactions (block 6) among the evolving specifications enables users to explicitly focus and overcome inherent conflicts, which is better than rework [Clausing 1993]. For instance artefact requirements such as 'must be reliable' and 'cheap' give rise to conflicts. Thus QFD provides guidance to 'where' engineering effort should be applied and similarly where not to invest time and money [Eccles 1994].

QFD limitations

Mainly focuses on 'use phase' awareness

QFD mainly supports designers in taking a 'use phase' view of their design problem [Roozenburg et al. 1995]. An extension is to employ cascading QFD matrices that cover the development process [Sivaloganathan et al. 1997]. In this way, QFD can handle interactions covering multiple product development stages. However, being sequential, changes to the product QFD chart have to be laboriously propagated to the other QFD charts (part, process and production) at the expense of cost and time [Jacobs et al. 1994].

Influenced by limitations of a team based approach

QFD requires profound knowledge from various fields such as marketing, design and production [Jacobs et al. 1994]. Thus limitations discussed earlier of a team-based approach such as 'interim isolation' are inherited by QFD. Also, for practical reasons, the number of team members involved in the use of QFD is often limited, thus restricting the amount of knowledge directly available during its use [Hague et al. 1998].

No proactive support to revealing interactions

With QFD, it is the users who identify *which* design requirement *interacts* with artefact life issues. This ability is subject to the user's knowledge of the problem domain. QFD does not pro-actively support designers in revealing 'what' these interactions are – it only assists in documenting those revealed.

Difficult to rapidly explore

The number of 'whats' that can be handled with QFD is limited to about 20-35 because if there are about 30 'hows', then there are about 600 items to fill in [Sivaloganathan et al. 1997]. This limits requirement exploration due to the difficulty in quickly returning to an earlier state [Jacobs et al. 1994].

4.2.3 Failure Modes & Effects Analysis

A formalized analytical method

One way enabling designers to cater for artefact life consequences is through 'Failure Mode and Effects Analysis' (FMEA), this being a formalized analytical method for the systematic identification of possible failures and the estimation of the related risks [Pahl et al. 1996].

Strengths of FMEA

Promotes provident thinking

FMEA *promotes* systematic thinking [Ranky 1994] by asking "What could go wrong with the artefact or the process involved in creating the artefact?, How badly might it go wrong? and What needs to be done to prevent failures?"

Records artefact 'life failure' knowledge for future reuse

The result of an FMEA procedure is documented as an FMEA chart (e.g. see [Pahl et al. 1996]). This provides a concise format for formally documenting possible artefact failures, their consequence, their likely cause and possible remedial measures. Thus, FMEA charts support the retention of 'artefact life' failure knowledge for reuse in subsequent design sessions.

Limitations of FMEA

No pro-active support

FMEA does not infer failures, their consequences, causes and remedies. FMEA only *promotes* designers to reveal potential failures. That is FMEA does not pro-actively support designers in providence.

Segmented views

The awareness provided through the use of FMEA is mainly related to the 'use' life-phase. FMEA can be applied to realization processes at the process planning stage for revealing fabrication or assembly artefact deficiencies [Healey 1994; Ranky 1994]. These FMEA variants are used *independently*, this leaving it up to the user to reveal any interactions [Healey 1994].

Late Awareness

The FMEA tabular chart requires designers to list down the components making up the sub-assembly (or artefact) being assessed [Healey 1994; Ranky 1994; Pahl et al. 1996]. That is, FMEA provides a means of taking into consideration potential problems, *late* in the design process, before 'a design is signed off and before production commences' [Healey 1994].

Requires a team effort

FMEA requires a team effort [Ranky 1994; Pahl et al. 1996]. This makes providence with FMEA to be artefact specific. However, it makes FMEA subject to limitations associated with teams discussed earlier in this chapter, subjective and sometimes bureaucratic [Norell 1993].

4.2.4 Rapid Prototyping

Principles of rapid prototyping

Prototyping is a means used to realize physical artefact prototype models that can then be *assessed by humans* to reveal knowledge of how an artefact behaves during different life-phases. One such means is *Rapid Prototyping (RP),* which commences with the generation of a 3-dimensional geometric model using a CAD system. Special software slices up the geometric model into layers, these then sequentially physically realized to eventually generate the physical model. Different technologies such stereolithography and laminated object manufacturing [McMahon et al. 1993] can be used to realize the physical model from liquid polymer, plastic powders or paper. Details of these technologies are found in [Ranky 1994].

Strengths of Rapid Prototyping

Rapid realization of an artefact prototype

With RP, the artefact development team can realize a physical model *within hours* rather than months [Ranky 1994], directly from 3D geometric models.

Prototypes can be viewed from multiple perspectives

RP allows development teams to realize *several* different physical models *without* the need of expensive tooling (e.g. moulds) to assess customer requirements, manufacturing, assembly, maintenance and other artefact life issues [Ranky 1994]. Rapid prototyping therefore enables a team to reveal knowledge of *multiple* artefact life issues before the *real* artefact's realization.

Limitations of Rapid Prototyping

Providence is late

The need of a geometric model *before* the physical model can be realized means that RP supports designers in considering artefact life issues very *late* during design, *after* the synthesis of a detailed component solution.

Awareness is not solution specific

Whereas the form and size of a physical model realized with RP may be very similar to the real artefact, the model's material is normally not [Grote et al. 1995]. Thus, a drawback with RP that the insight provided is *not* specific to the real artefact's material. Exploring different materials and their LCC is therefore not readily supported with RP.

No pro-active providence support

RP does not *pro-actively* support users in revealing artefact life problems and/or opportunities. Artefact life issues revealed depend on team members. Therefore, revealing any *interactions* between the different artefact life perspectives is subject to the ability and expertise available within the team.

4.3 Means Directly Supporting Providence

This section reviews four of the major means that are considered relevant to *explicitly providing* designers with artefact life knowledge. These are *DFX guidelines, Numerical Analysis, Feature Based Design Tools* and *Artificial Intelligence (AI) [Rich et al. 1991] Based Tools.* These means basically differ in how the knowledge is codified and processed.

4.3.1 DFX Guidelines

Background to DFX guidelines

A class of means that allows captured relationships between an artefact and artefact life issues to be explicitly provided and utlized during design to *predict 'what-if' effects* [Huang et al. 1997], are *Design For X (DFX)* guidelines. Examples are found in [Boothroyd et al. 1991; Bralla 1996; Pahl et al. 1996]. They are basically prescriptive design guidelines for creating artefact families, artefact structures and component geometry that address X-ability issues [Nowack 1997]. They *guide* designers in converging onto a design solution satisfying an X-ability. For example, a design for assembly (DFA) guideline is to 'minimize the number of parts in an artefact to reduce assembly operations'. 'X' has two meanings [Andreasen et al. 1993], a *life-phase aspect*, e.g. assembly (DFA) or a *performance measure* e.g. cost, (DFC) [Feng et al. 1996].

DFX Guideline strengths

Provide designers with life-phase knowledge

DFX guidelines explicitly provide designers with codified knowledge of areas with which they are not usually familiar [Olesen 1992]. Various organisations are aware of this benefit. For instance, DuPont [DuPont 1992] provide their customers with DFX guidelines that prescribe rules for building in producability when designing components made from their thermoplastic materials.

Provide a means for capturing and sharing useful knowledge

DFX guidelines provide a means of formally capturing knowledge concerning relationships between artefact solution parameters and life-phase system behaviour. This makes it possible for such knowledge to be *shared, distributed* and *reused* during subsequent design sessions.

DFX Guideline limitations

Segmented awareness

The body of knowledge compiled in DFX guidelines tends to be *segmented* by artefact life aspect (Figure 4.4). Thus DFX guidelines essentially allow designers to foresee LCCs with respect to a single 'X', (e.g. assembly, with

DFA) thereby guiding designers in generating solutions that satisfy a single life-phase aspect. As examples in chapter 2 (e.g. case 2) reflect, this may result in propagation effects on other 'Xs'. Further, DFX guideline knowledge is material domain segmented (Figure 4.4). For instance, a thermoplastic component DFM guideline is different from DFM for sheet metal components. Thus, DFX guidelines aid in the generation of solutions satisfying an X-ability for a specific domain [Shankar et al. 1993].

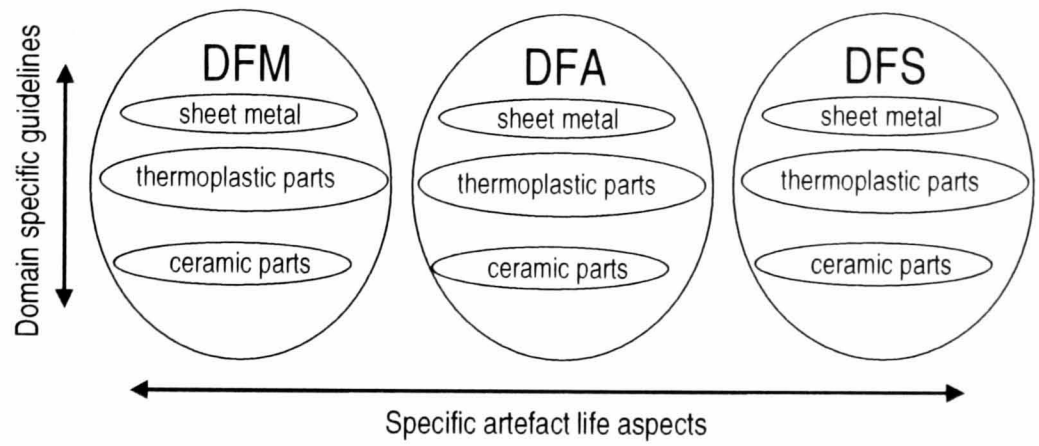


Figure 4.4 - Life-aspect and domain specific segmentation of DFX guidelines

Hindrance to explore across multiple domains

This domain specific segmentation hinders designers from *rapidly exploring* alternative domains and foreseeing associated total life opportunities and problems. For instance die-cast components made of aluminium and zinc alloys can be equally made from GRIVORY GV [EMS 1996]. Choosing GRIVORY results in a number of consequences including a 4-5 times longer mould tool service life and a significant component weight reduction.

Provide generic awareness

DFX guidelines tend to be generic in the sense that they do not cater for an artefact's life specific scenario. For instance, DFM guidelines do not reflect the actual manufacturing concerns of the user [Molloy et al. 1993].

Providence is late

In principle, DFX guidelines can be employed during synthesis so as to build in X-ability into the solution, or during solution analysis to estimate the solution's behaviour from an X-ability point of view [Tichem 1993]. DFX guidelines are numerous. Deciding *which* one is applicable to the solution in hand is difficult and confusing [Huang et al. 1997]. This depends upon the designer's ability to relate specific guides and the current solution. Thus, due to the effort this requires, DFX guidelines tend to be used separate from the synthesis activity. Further, due to the information required for their use (e.g. engineering drawings), DFX guidelines tend to be used for candidate solution analysis. Although this ensures that a solution released for realization caters for artefact life issues, it means that DFX is taking place *late* during detailed

design, 'when the game is over' [Gardner et al. 1993], namely for 'corrective re-design' [Nowack 1997]. Thus, DFX benefits are lost [Dalgleish et al. 1998] as providence is not being exploited during early design, when the DFX knowledge can be effectively utilized for generating life-oriented solutions.

4.3.1.2 DFX Meta-Methodology

Use of meta-methodolgy

For life-oriented design, a solution has to satisfy *multiple* 'X's [Watson et al. 1996; Tichem 1997]. The use of multiple DFX guidelines, besides being impractical [Huang et al. 1997], can result in conflicting recommendations [Andreasen et al. 1997]. For this purpose, a 'meta-methodology for the application of multi-DFX guidelines' has been proposed [Watson et al. 1996].

DFX Meta-methodology Strengths

Helps handle conflicting DFX guidelines

Similar to QFD, the DFX meta-methodology makes use of a weighted matrix approach, in this case to enable different DFX guidelines to be compared and any *competing* or *reinforcing* interactions between them to be revealed.

Universal application

As claimed by the authors, this meta-methodology is suitable for application to any industry sector and size of enterprise [Watson et al. 1996].

DFX Meta-Methodology Limitations

No pro-active support

The meta-methodology requires that the user first specifies *which* DFX guidelines are to be compared for any interactions. Secondly, it is the methodology user who has to identify any *interaction* between the guidelines selected for comparison. Therefore, whilst directly supporting designers with DFX knowledge, designers are not pro-actively supported in revealing knowledge of interactions between the different DFX guidelines.

Limited life-span awareness

The process of determining each relationship can become tedious for a large number of guidelines [Watson et al. 1996]. Generally not more than three DFX tools can be handled at any one time. This thus restricts the methodology's effectiveness of handling the phenomena of propagation effects.

Late providence

The meta-methodology is essentially employed for analysis, "after technically feasible solutions have been developed" [Watson et al. 1996].

4.3.2 Numerical Analysis

Operating principle

A computer-based class of means allowing designers to explicitly acquire knowledge of an artefact's life performance is numerical analysis (N.A.). With computer based N.A., the user can build up a mathematical model of a component or assembly for analysis by appropriate N.A. software. These tools are based on different types of methods such as *finite difference (FD)* [Tizzard 1994], *finite elements (FEM*) [Cook 1995] *or boundary elements (BEM)* [Tizzard 1994]. Basically, a 2-dimensional or 3-dimensional geometric component model is prepared during a *pre-processing stage* (Figure 4.5) and divided into a number of *elements* by *discretization* [McMahon et al. 1993; Cook 1995]. The *physical properties* of each element are then defined e.g. partial differential equations describing the behaviour of the function (e.g. temperature) to be predicted. The resultant system of equations, when solved yields a value for the function for each element making up the component. *Boundary conditions* describing how the component will be *loaded* during its use phase and *restrained* from moving are then defined. The resultant physical model is then input to the N.A. solver for processing, during which knowledge of the input model's life performance is explicitly generated. During the *post-processing* stage, the results are *analyzed* by the user, this providing knowledge of any areas of the original model that need to be modified to improve the component's behaviour.

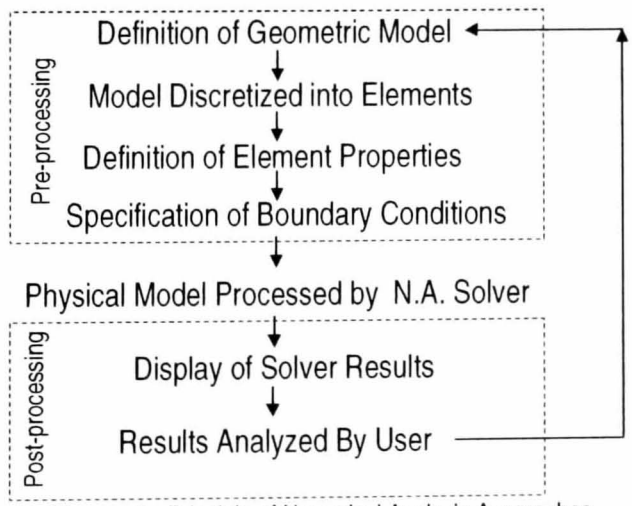


Figure 4.5 - Principle of Numerical Analysis Approaches

Strengths of Numerical Analysis

Avoids the need of physical models

With N.A., a component's behaviour can be predicted without the need of testing physical prototypes, this saving time and costs [Tizzard 1994].

Prediction of various behavioural properties

N.A. approaches allow users to predict various artefact behavioural properties such as stress, vibrations, fluid flow and heat transfer [Tezuka 1992; Balendra et al. 1995; Cook 1995]. The FEM has also been applied to predict the

behaviour of life-phase systems. Applications include predicting the influence of punch tool dimensions on a blanking process [Choy et al. 1995], or the heat transfer of an injection mould tool [Tizzard 1994; Jiafu et al. 1995] in order to foresee realization problems such as component warping and shrinkage.

Provide specific awareness

As the N.A. procedure (Figure 4.5) involves the definition of life specific boundary conditions, these approaches support designers with 'life specific' providence.

Limitations of Numerical Analysis

Providence is late

Numerical analysis require a geometric model [McMahon et al. 1993; Tizzard 1994; Cook 1995] of the candidate solution *before* discretization (Figure 4.5). Therefore, N.A. supports providence *after* conceptual solution synthesis.

Provide a narrow life-span view

N.A. is mainly employed to foresee the behaviour of artefacts during the 'use phase', or separately, to predict a life-phase system's performance e.g. a mould tool [Jiafu et al. 1995]. Thus as currently applied, N.A. does not support designers in concurrently foreseeing *multiple* life-phase issues. Thus, providence is with a *narrow* and *segmented* view.

Artefact life exploration is difficult

Using an alternative component form or material requires the user to go through the pre-processing steps, before the new model can be *re-submitted* to the numerical solver and new insights obtained (Figure 4.5). Thus, with N.A., artefact *exploration* is laborious. Also, as currently employed, N.A. limits exploration to the artefact solution and not the life-phase systems.

4.3.3 Feature Based Design Tools

Feature based design & recognition

Designers can foresee artefact life issues during design through a *feature based* design approach [Salomons et al. 1993]. As described by Weber [Weber 1996], a *feature* is an information unit (element) representing a *region of interest* within an artefact. Examples are a slot or snap-fit. Such features explicitly capture knowledge *relating* [Andreasen et al. 1996] an artefact region to artefact life issues such as *process planning* [Wierda 1991; Shah et al. 1995], *manufacturability* [Molloy et al. 1993], *assembly* [Jared et al. 1994], *production cost* [Feng et al. 1996] and *compatible realization systems* [Terpenny et al. 1993]. This concept of features is employed in computer based life-oriented design tools [Vajna et al. 1997] (see Appendix B) either for *solution synthesis* or for *solution analysis*. Synthesis tools provide a *set* of

features that enable designers to generate descriptions of the solutions - an approach termed feature based design (fbd). Once the description is *complete*, it is then submitted to an analysis module which explicitly reveals knowledge about the artefact life issues as in [Changchien et al. 1996]. With analysis tools a given geometric solution is first submitted to a *feature recognition* [Bartholomew et al. 1991] module to identify 'features' in order to generate a feature based description of the input solution. Again, this description is submitted to an analysis module to reveal artefact life issues.

Strengths of Feature Based Design Tools

Providence supported with synthesis in different viewpoints

Features are applicable to artefact solutions being described from different synthesis viewpoints, as they do not necessarily relate to geometry [Vajna et al. 1997]. For instance, *functional features* [Schulte et al. 1993] can be used during functional domain synthesis and *form features* [Feng et al. 1996] during constructional domain synthesis. This allows artefact life issues to be *related* [Andreasen et al. 1996a] to viewpoint-specific 'regions', thereby supporting providence during artefact synthesis taking place from different perspectives.

Limitations of Feature Based Design Tools

Providence is late

With current synthesis and analysis feature based tools, a description of the candidate design is required before artefact life issues can be revealed. Thus providence takes place *after* the candidate solution has been generated.

Provide a generic insight as life-phase models are not being modelled

As features concern artefact regions, they can be considered to be *Product Design Elements (PDEs)* (see section 2.1). Literature indicates that the concept of features is not being applied to reusable *Life Cycle Phase Elements (LCPE)* (see section 3.3). Thus, with a feature-based approach, designers can model artefact solutions and hence foresee consequences of their solution interacting with an assumed *(fixed)* life-phase model, the latter not necessarily reflecting the artefact's specific life scenario.

Mainly used for component design

Feature based design tends to be employed for component level design, normally with *form* features. In this sense, synthesis results in a geometric model that is neutral to other design characteristics such as the material.

4.3.4 Artificial Intelligence Based Means

AI based means that are being used to explicitly provide designers with knowledge of artefact life issues are *constraint networks*, *knowledge based systems* and *case-based reasoning tools*. This class of providence means is discussed here, with a review of representative work presented in Appendix B.

4.3.4.1 Constraint Networks

Principle of a constraint network approach

A constraint network provides a computer based approach that can be utilized to support life-oriented design [Bahler et al. 1994; O'Sullivan 1997] by allowing users to explicitly acquire knowledge of the consequences of a designer's decision on life-phase issues [Bowen et al. 1990; Oh et al. 1995]. A *constraint network (CN)* is a collection of *objects* (parameters) and a set of *constraints* which must be satisfied by the values that are assumed by the objects [Bowen et al. 1990; Oh et al. 1995]. A constraint is some relationship which must be satisfied by some subset of the parameters in the network [O'Sullivan 1997]. These constraining relationships allow, for instance, the effect of a designer's decision on manufacturing options [Bowen et al. 1990] to be foreseen.

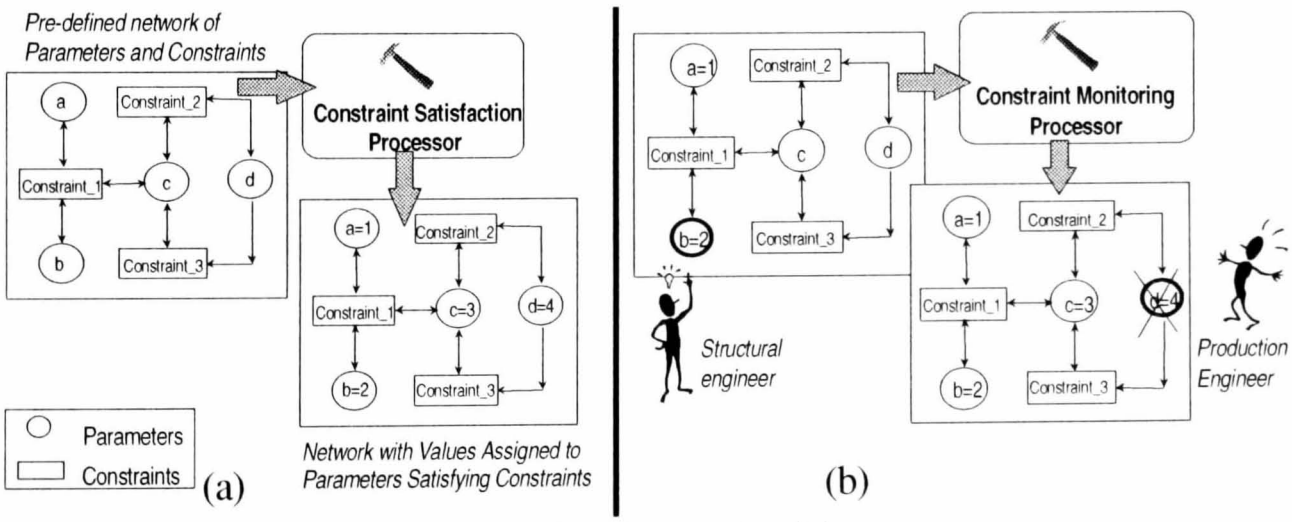


Figure 4.6 - Constraint processing techniques

Selecting and *testing* parameter 'values' in a CN can be done by a combination of computers and humans [Bowen 1991]. If done by a computer, the process is termed a *constraint satisfaction problem (CSP)*, this analogous to automatic design [Bowen et al. 1990] as illustrated in Figure 4.6a. If performed by a combination of humans and computer, then it is termed *constraint monitoring* [O'Grady et al. 1991] (Figure 4.6b). In constraint monitoring, the designer interacts with the network asserting values to one or more parameters. The constraint monitoring processor than *infers* values for other objects attached to the network (where possible), *analyzes* the new state

in the network, reporting back any constraint violations caused by the user's decisions or by the inferred consequences of these assertions.

Strengths of Constraint Networks

Support the representation of artefact life knowledge

Constrains can express the restriction exerted on objects in a design problem by, for instance, the functionality, material properties and life-cycle issues. Thus, DFX guidelines can be represented as constraints [O'Sullivan 1997].

Provides non-directional inference

CN support *non-directional inference [Bowen 1995].* Thus, when values are assigned (or acquired) by any of the objects in the network, values can be inferred for other objects forming part of the network.

Useful to foresee conflicts between decisions

Non-directional inference allows CN to be employed for monitoring assertions/retractions in co-operative design, where *many* experts from various disciplines can *view* the candidate design from their perspective [Bahler et al. 1994; Tang 1996]. For example KLAUS3 [Bowen 1995] supports the interaction between *several* design team members concerned with making decisions on printed wiring board (PWB) design. KLAUS3 reports back any violations detected to the *appropriate* team members.

Useful for parameter value exploration

During design, decisions may be made under certain assumptions. Decisions made may therefore need to be *revised* in the light of new information. Constraint monitoring can thus help foresee any violations when exploring alternative parameter *values* forming part of the network [Tang 1996].

Limitations of Constraint Networks

Providence is late

As argued in Chapter 3, during design, both the solution and the artefact life problem space co-evolve. As a CN monitors a constraint-based artefact model [O'Sullivan 1997], CN do not readily support providence *during* solution synthesis, when *new* parameters are being added. Rather, CN support artefact life providence *after* solution synthesis.

Problem with multiple DFX

Handling multiple objectives as in multi-DFX, cannot be readily achieved [O'Sullivan 1997] as this requires an extension to the CSP paradigm .

No pro-active providence support

Although "at run-time users can declare additional parameters and constraints..." [Bowen 1995] forming part of the network, this actually requires that designers *themselves* add new constrains between these new parameters

and the previously defined parameters. Thus with CN, designers are not *pro-actively* supported in foreseeing artefact life issues and new decision spaces co-evolving with their newly added parameters.

Mixture of definable & derivable properties hinders exploration

The powerful non-directional inference provided by CN provides a 'modeling world' that allows designers to for example 'determine the impact of cost decisions on functionality' [Bowen 1995]. However, as argued in Chapter 3, a distinction needs to be made between characteristics that can in reality be *defined* by designers and others that are only *derived* [Mortensen 1995]. This thesis argues that designers can assume a target value (goal) for derivable properties but they can *only* define the basic characteristics of the artefact and of the life-phase systems.

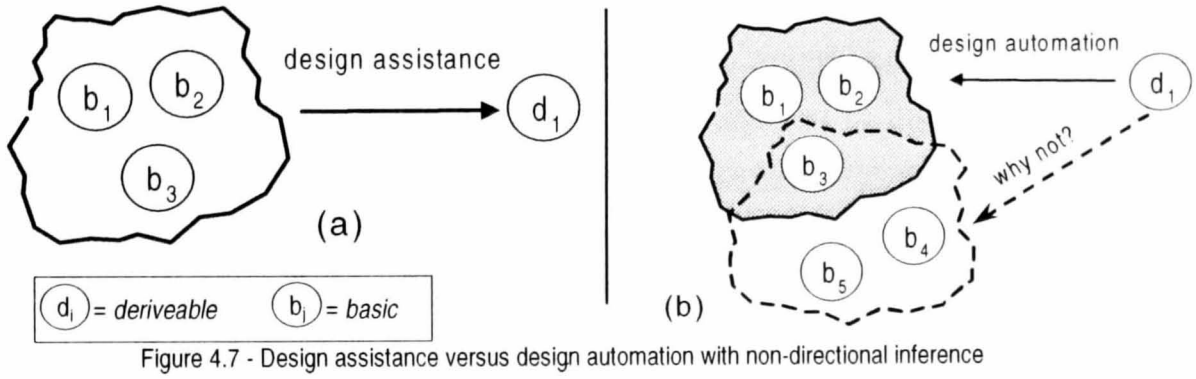


Figure 4.7 - Design assistance versus design automation with non-directional inference

Thus, inferring derivable properties (d_i) from basic properties (b_j) provides design assistance (Figure 4.7a) but inferring values for a *set* of basic parameters (b_j) for a specified value to a derivable property (e.g. cost) is design automation. This is because in reality, it may be possible to have *different sets* of (b_j) for a given (d_i) (Figure 4.7b). Thus, inferring (b_j) from (d_i) is possible with CN but *only* for pre-defined relationships. This therefore hinders design solution exploration.

Mixes domain & candidate solution knowledge

With CN, designers can build a constraint based description of an artefact model [O'Sullivan 1997]. Thus, the resulting CN represents a *mixture* of the domain knowledge (e.g. constraining relationships between different design parameters) and the current working knowledge about the candidate solution. This mixture makes it difficult to *reuse* domain knowledge in new design scenarios. This also makes domain knowledge *maintenance* difficult to achieve with CN [Molloy et al. 1994], an issue that cannot be ignored with design support tools [Duffy 1997].

4.3.4.2 Knowledge-Based Systems

The amount and variety of knowledge [Tomiyama et al. 1995] that has to be processed in order to generate life-oriented design solutions has attracted the application of Knowledge-Based Systems (KBS). Knowledge in these tools is either expertise [Rychener 1988] or public knowledge found in sources like textbooks [Giarratano et al. 1994]. As a result, the terms *KBS* or *expert system* are often used synonymously [Giarratano et al. 1994] to describe these type of tools. Their utility in supporting life-oriented deign from the perspective of providence is evident from various applications such as those in [Ishii 1991; Levitt et al. 1991; MacCallum 1991; Molloy et al. 1993; Victor et al. 1993; Swift et al. 1994; Changchien et al. 1996; Su et al. 1997].

Principles of KBS

A KBS [Rychener 1988; Giarratano et al. 1994] consists of a *knowledge base,* an *inference engine* and a *user-interface*. The *knowledge base* is the main repository of the knowledge employed by the system to address the specific problem for which it has been developed. It is common for the knowledge base to consist of a combination of knowledge in the form of concepts, rules, models and strategies [Rychener 1988]. *Concepts* are a declarative representation of domain objects. For instance 'ABS' is a thermoplastic material, it has a set of properties with certain values. *Rules* are relationships linking 'causes and effects', 'evidence and likely hypothesis', 'situations and desirable actions to perform' [Eubanks et al. 1993; Changchien et al. 1996]. *Models* are collections of interrelated rules, usually associated with a particular problem. These can for example represent a sub-system of a complex mechanical structure [Walters et al. 1988]. *Strategies* are rules and procedures used to aid in the utilization of the rest of the knowledge base, by for instance resolving conflicts when rules are equally applicable for a given situation. The *inference engine* provides the problem solving strategy applied by the KBS. Two common methods are *forward chaining* and *backward chaining* [Giarratano et al. 1994].

Strengths of Knowledge-Based Systems

Support knowledge retention and amplification

A characteristic of KBS relevant to life-oriented providence is that they provide a means by which the *breadth* of expertise possessed by different artefact life actors can be *retained, distributed* and explicitly *re-used* even *after* their retirement as reflected in typical implementations [Ishii 1991; Meerkamm 1994]. Thus, a KBS allows the pooling of expertise of a number of specialists,

to provide a *'knowledge amplifier'* [MacCallum 1992] to a designer attempting to purposely transform a design problem into a solution (Figure 4.8).

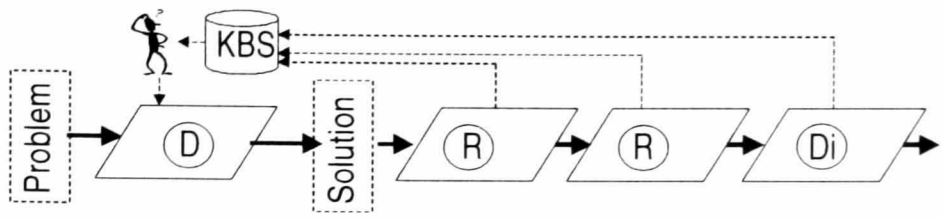


Figure 4.8 - Ability of KBS to retain a collection of distributed life-cycle knowledge

Predictive power useful for providence

KBS are useful for supporting providence due to the inference engine's *predictive* power. In essence, given a description i.e. facts about a candidate solution, a KBS can predict consequences using knowledge embedded in the knowledge base [Hague et al. 1995]. For example, the *Design Critique System* [Changchien et al. 1996] predicts manufacturing and assembly issues associated for a given rotational part. Appendix B provides details of a number of other KBS exploiting this predictive power for artefact life providence.

Limitations of Knowledge-Based Systems

Providence is exploited late

The predictive power offered by KBS is generally being employed for *candidate solution analysis*. For instance, both the MIDAS system [Bonfield et al. 1997] and the 'Design For Service' tool [Eubanks et al. 1993] require a candidate solution as an input before revealing artefact life issues (see Appendix B).

Not exploited for concurrent life-phase synthesis

As argued in Chapter 2, during an artefact life, various life-phase systems *interact* with the artefact. The concept of using KBS to support the synthesis of an artefact model has been taken e.g. the NODES system [Duffy et al. 1996]. Some work is reported concerning supporting the synthesis of life-phase models [Young 1996]. However, the KBS reviewed do not support the concurrent synthesis of artefact and life-phase models making it difficult to foresee *life-specific* interactions.

Artefact life exploration is difficult

KBS tend to employ *chunks* of heuristic knowledge with no underlying *model* of the solution being described [MacCallum 1991]. This thesis argues that such a *knowledge-centred* rather than a *model centred* design approach limits design exploration. For instance, a heuristic found in the thermoplastic domain is that if a component has a rib, then a consequence is that a sink mark is likely to form [DuPont 1992] during the realization phase. Such a knowledge chunk can be employed within a KBS, as in the rule found in [Huh et al. 1991], reproduced here:

IF: The Material is GE NORYL N190

and The Wall Thickness is [t]

and The Input Root Thickness [T_b] is bigger than 0.8 [t]

THEN The Possibility of Bad Sink mark = 9/10

and The Possibility of Warpage = 8/10

and Warning Message: Reduce [T_b] smaller than 0.8[t].

This example *assumes* that the realization phase process is injection moulding. Therefore, using such *domain specific* knowledge with no underlying life-phase models does not readily allow designers to explore alternative processes and see the resulting artefact LCCs or to explore alternative process parameters (e.g. injection pressure) in an attempt to avoid the formation of sink marks.

Provide segmented views

With no underlying life-phase models, a KBS can only provide *multiple* but *segmented* views of a candidate solution (e.g. mfk [Meerkamm 1994]). A KBS aimed at providing multiple views of a printed circuit board solution is *MIDAS* [Bonfield et al. 1997]. This provides users with a *process model* containing information about the available production facilities. However it has no underlying life-phase system models.

Knowledge maintenance difficulties

A problem with KBS, which is more intense in life-oriented design due to the *breadth* of knowledge involved, is that of *knowledge maintenance*. Unless the knowledge is properly structured, maintenance will be difficult, making the KBS obsolete [Duffy 1997] in a world where knowledge is dynamic.

4.3.4.3 Case Based Reasoning Tools

Principles of case based reasoning

Case-based reasoning (CBR) tools apply human experience, stored in a computerised form termed a 'case', in an attempt to assist solving similar problems, in slightly *altered* contexts. In a case-based, life-oriented design approach, captured artefact life knowledge stored in *cases*, is explicitly used to augment designer experience in solving a design problem. Examples are found in [Wood III et al. 1996; Kim 1997] (See Appendix B). The process of CBR involves *recalling* a *relevant* case from the *case base* and then *adapting* this case for the solution of a new problem (Figure 4.9). CBR design tools vary in the way the case base is organized, the procedures for recalling relevant cases, the methods and knowledge available for adapting a case [Maher et al. 1995].

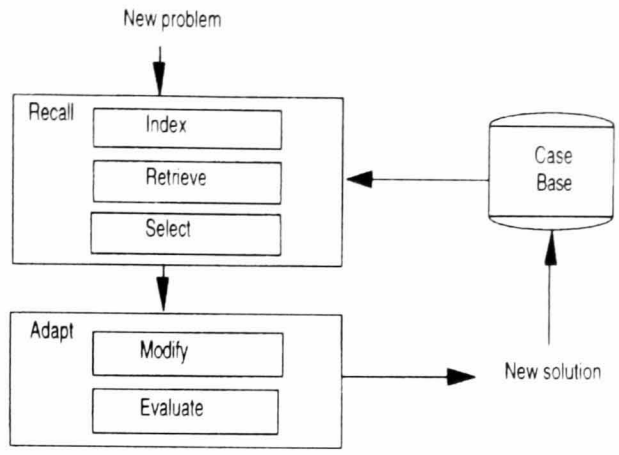


Figure 4.9 - CBR design approach - [Maher et al. 1995]

To allow CBR systems to quickly and accurately search for relevant cases, cases need to be organized in a structured way. Recalling relevant cases is a pattern-matching problem. This can be broken down into (Figure 4.9) *indexing* the pattern input by the designer then *retrieving* suitable cases by searching the case-base for individual cases that match the indexed pattern. An appropriate case is *selected* from those retrieved, with a *rank* to indicate how close the match is. The selected case is usually *modified,* through *case adaption,* to become applicable to the current design situation. This employs additional knowledge to help adapt the case [Maher et al. 1995].

Strenghts of CBR

A case integrates artefact and life-phase knowledge

A case can represent artefact and related artefact life knowledge such as on assembly issues [Kim 1997] in various forms including multimedia [Wood III et al. 1996]. CBR thus provides a means for storing artefact and related life-phase knowledge in single case in order to directly supporting providence.

Limitations of CBR

Difficulty to reveal multi-X interactions

A single retrieved case can explicitly provide designers with knowledge of relationships between an artefact and related life-phase issues [Wood III et al. 1996]. However, foreseeing any *interactions* between different life issues stored in different cases is left up to the user.

Providence is separate from synthesis

As the retrieval of a relevant case is initiated by designer input (Figure 4.9), then when a case is retrieved and adapted, any problems inherent in the resultant solution are not automatically re-submitted CBR. It is the user who has to *foresee* co-evolving problems and then input these to the tool as a suitable new query. Thus, CBR does not readily support designers in foreseeing artefact life problems co-evolving with their solution during synthesis.

Providence tends to be generic

Cases provide knowledge about similar artefacts and related life issues. Such a knowledge *chunk* supports designers in harmonizing an artefact solution to life-phase systems' requirements, which are however assumed *fixed* in the case retrieved (Figure 4.10). Therefore, without an underlying explorable life-phase model, CBR does not support designers in foreseeing *life-specific* interactions between artefact and life-phase system solutions.

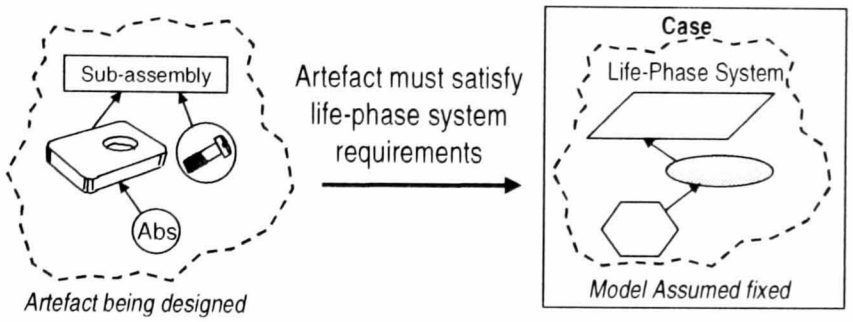


Figure 4.10 - Case retrieves 'fixed' knowledge on life-phase system

Life-oriented providence requires a large case-base

Expanding the design focus to a *multiple* of artefact life phase requirements presents unique problems for a case based approach [Wood III et al. 1996]. This is because design information must be considered at many levels of *abstraction* and from many *viewpoints*. Thus, a limitation to life providence is that the cases need to be *large* due to the storage of multiple life issues. An important issue in case based providence design is the *organization* and *searching* of relevant cases. Such large case bases introduce *indexing* problems to ensure the retrieval of relevant cases. One way used to overcome this issue is through an *intelligent thesaurus* [Wood III et al. 1996].

Lack of pro-active case adaption support

Cases retrieved are rarely a perfect fit to the current solution, meaning that *case adaption* is required to foresee artefact life issues specific to the current solution. With the current state-of-the-art, designers employing such CBR tools have to *assist*, or even entirely perform the case adaption process [Maher et al. 1995]. This is a drawback to pro-actively aiding providence, as designers require additional artefact life knowledge for successful case adaption. A promising avenue for addressing this problem is the use of *hybrid case-based design systems* [Maher et al. 1995], where multiple reasoning methods are incorporated within the CBR paradigm.

Knowledge duplication & maintenance

Knowledge about an artefact and related life-ssues are stored in a single case. Thus, similar artefact life knowledge may unnecessarily be duplicated in different cases. This makes artefact life knowledge maintenance difficult.

Table 4.1a – Comparative Matrix of Providence Means

Means	*Solutions generated:* *(N)one, (A)rtefact, (L)ife-phase*	Artefact Life Knowledge			Other Comments
		Revealed during (s)nthesis or (a)nalysis	*View Single (X) = 1(X) Multiple (X) = Σ(X)*	*Type (G)eneric or life (S)pecific*	+ strengths / - limitations
INDIRECT SUPPORT					
Teams	Normally (A) but it is possible to generate (L) solutions;	(s) - if all team members are present physically or *virtually* [Cleetus 1993] during synthesis; otherwise mainly during (a) in review meetings	Σ(X)	(S) if the team members represent 'solution specific' artefact life actors	+ life specific LCCs can be foreseen *if* all members are present during synthesis + the right multifunctional team provides a synergy of life-cycle knowledge; - it is difficult to assembly a team with all adequate life-cycle expertise [O'Grady et al. 1991]; - team meetings tend to be short & infrequent [Ishii 1991; Dym 1994] - decisions with potential LCC can be made in the interim period between meetings; - expertise is lost with retirement/change of members [Salzberg et al. 1990]; - difficulties in recognizing complex dispositional effects [Andreasen et al. 1990]; - logistic management difficulties [Bowen 1995] influence providence;
QFD	Early (A) solution, in the form of 'how' individual customer requirements can be met; currently also used for selecting the critical 'manufacturing processes' and 'process parameters' [Sivaloganathan et al. 1997]	(s)	1(X)-mainly use phase; currently also used for considering 'manufacturing processes' [Sivaloganathan et al. 1997]	(S) only *if* a 'product specific' design team is employed [Sivaloganathan et al. 1997]	+ motivates provident thinking as from the task clarification design stage; + supports the systematic handling of trade-offs between requirements; - being a team-based approach, it suffers limitations associated with teams; - which interactions between specifications and artefact life issues are revealed, depends on the user's domain knowledge and solution interpretation; - solution exploration is difficult;
FMEA	(N) – solution is input to FMEA	(a)	Can be Σ(X) but segmented; mostly used for 'use' phase providence	(S)	+ promotes provident thinking of possible use phase failures; + provides a means of documenting potential failure modes for future reuse; - being a team-based approach, it suffers limitations associated with teams; - failure modes are revealed late [Norell 1993] during detail design ; - not pro- active i.e. providence depends on the user's solution interpretation; - single focus at a time i.e. does not handle propagation effects;
Rapid Prototyping	(A); sometimes it can be used to generate (L) e.g. 'rapid tooling'; (A) model generated is normally of a different material from the real artefact;	(a)	Σ(X) but segmented; mostly used for 'use' phase providence	(G)	+ can rapidly generate physical artefact prototype models; + realized prototype model can be viewed from multiple life-phase perspectives; - artefact life issues & interactions revealed depend on team members knowledge; - providence late - requires a geometric model as an input; - awareness not specific - the prototype's material is different from that of the artefact;

Table 4.1b – Comparative Matrix of Providence Means

Means	Solutions generated: (N)one, (A)rtefact, (L)ife-phase	Artefact Life Knowledge			Other Comments
		Revealed during (s)nthesis or (a)nalysis	View Single (X) = 1(X) Multiple (X) = Σ(X)	Type (G)eneric or life (S)pecific	+ strengths / - limitations
DIRECT SUPPORT					
DFX guidelines	(N) they support the generation of an (A) solution, but they are used separately from solution modelling	Mainly during (a) – as guidelines are numerous, they tend to be used separate from synthesis [Huang et al. 1997].	Each guideline provides 1(X) e.g. DFA, or DFM	(G) - as they do not consider the user's actual resources [Molloy et al. 1993];	+ provide a means of capturing & sharing useful relationships between artefact solution parameters and life-phase system characteristics; - narrow, segmented views i.e. difficult to handle propagation effect phenomena; - being segmented, multi-material domain exploration is not supported; - as guidelines are numerous, deciding which is applicable is difficult & confusing [Huang et al. 1997];
DFX Meta-Methodology [Watson et al. 1996]	(N) Methodology supports the generation of an (A) solution, but is used separately from solution modelling	(a) of feasible solutions	Σ(X) but limited to 3	(G)	+ helps handle conflicting DFX guidelines - interactions between different DFX guidelines need to be revealed by user
Numerical Analysis	(N) – solution is input to the Numerical Analysis solver	(a) - geometric model of the candidate solution required as an input	1(X) but segmented; mainly for Use phase; some for Realization phase	(S)	+ prevents the need of physical models; + can be used to foresee various artefact behavioural properties; sometimes used to predict the behaviour of life-phase systems e.g. mould tool cooling; - tool provides a narrow, segmented view – normally artefact use phase behaviour; - artefact exploration is laborious; artefact life exploration not supported;
Feature based design tools	(A) models; most models are geometric, some include other characteristics such as materials and tolerances;	(a) – feature based description of the candidate solution is required before artefact life issues can be revealed	can be Σ(X) but segmented;	(G) as there is no underlying life-phase models to reveal 'life-specific' interactions	+ can be used to support synthesis taking place from different viewpoints; + features provide a formal artefact representation scheme that supports the integration of product life activities [Changchien et al. 1996]; - features are the result of a user's interpretation of an artefact's region [Jared et al. 1994] - features are related to artefact regions and are thus used for artefact synthesis ie. currently no 'life-phase' oriented features;

Table 4.1c – Comparative Matrix of Providence Means

Means	*Solutions generated:* *(N)one, (A)rtefact, (L)ife-phase*	Artefact Life Knowledge			Other Comments
		Revealed during (s)nthesis or (a)nalysis	*View Single (X) = 1(X) Multiple (X) = Σ(X)*	*Type (G)eneric or life (S)pecific*	+ strengths / - limitations
DIRECT SUPPORT - Artificial Intelligence Based Means *(see Appendix B for a review of a representative sample)*					
Constraint Networks	(A)	(a) of candidate solution - solution has to be first represented as a network of parameter relationships [Bowen 1995]; e.g. for post-design DFA analysis [Oh et al. 1995]	Σ(X) & interacting - assumes the relationships between the different perspectives are known & represented in the network;	(S) – when used for co-operative design between 'solution specific' artefact life actors	+ support the representation of various types of knowledge; + support 'non-directional' inference; + useful for parameter value optimization; - they treat design as a search to 'satisfy constraints' i.e. assumes constraints are known in advance and not co-evolving with the solution; - providence used for improving a solution rather than for guiding solution synthesis; - handling multiple objectives as in DFX not readily achieved [O'Sullivan 1997]; - mixture of definable & derivable properties hinders exploration; - difficult to maintain knowledge as they mix domain & candidate solution knowledge;
Knowledge based design tools	Currently, can be used for generating (A) or (L), but separately i.e. not concurrently	Currently used for (a) – a candidate solution is required as an input	Can be Σ(X), but segmented	(G) as there is no underlying life-phase models to reveal 'life-specific' interactions	+ support the retention of distributed artefact life knowledge; + they have a predictive power due to the inference engine, suitable for providence; - currently not used for the concurrent synthesis of artefact and multiple life-phase systems; - with no underlying life-phase models, artefact life exploration is difficult; - knowledge is not organized in terms of resources which have to be coordinated [MacCallum 1991]; - limited in handling interacting relationships [MacCallum 1991];
Case based reasoning tools	(N) – they support solution synthesis but they are used separately from solution modelling	During (s) but separate from solution modelling	Can be Σ(X) but, segmented	(G) as there is no underlying life-phase models to reveal 'life-specific' interactions	+ cases can conveniently integrate artefact and related life-phase knowledge; + represents knowledge in a variety of formats e.g. multimedia [Wood III et al. 1996]; - knowledge captured in the retrieved case has to be adapted to the current problem, frequently by the user [Maher et al. 1995]; - case adaption may introduce inconsistencies [Maher et al. 1995]; - interactions between the different views is left up to the user reveal unless this is explicitly stored as part of the case; - cases enable designers to optimize the candidate design solution, without supporting the exploration of 'artefact life' commitments; - similar artefact life knowledge may be captured in different cases, making knowledge maintenance difficult;

4.4 Current State Of Providence Support

The characteristics of the means reviewed are summarized in Table 4.1. Based on the review criteria, the following key observations can be made to how providence is currently supported and exploited for life-oriented design.

Providence Is Too Late – Separate from Solution Synthesis

Providence is exploited after the synthesis activity

As argued in section 3.3, *delaying* the consideration of artefact life issues leads to a series of lengthy and costly design iterations. Table 4.1 reflects that both indirect (e.g. FMEA) and direct (e.g. DFX) means support providence allowing designers to generate life-oriented design solutions. However, providence is supported late in the design process, as most means make artefact life knowledge available *after* candidate solution synthesis. An exception is a team based approach, but this requires all members to be *continuously* present during synthesis, which in practice is not the case.

Providence Life-Span View is Narrow & Segmented

Lack of support to foreseeing interactions across multiple life-phases

As argued in [Bonfield et al. 1997], work on means providing multiple perspectives is limited. Table 4.1 indicates that current means either provide a *narrow* view by focusing on one life-phase at a time (e.g. DFX guidelines), or multiple, but *segmented* views (e.g. mfk system [Meerkamm 1994]). Thus individual designers lack means supporting them during early design to foresee artefact life issues across *multiple* life-phase issues in an integrated way. Constraint networks go some way towards this issue but they mostly support designers to *monitor* the impact of their decisions on multiple life-phase issues rather than to foresee the impact. This is because constraint networks require that parameters of a solution and parameters of life-phase issues to be first expressed and related by the user him/herself.

Providence Is Mostly Generic – No Explorable Artefact Life Model

Lack of support to concurrent synthesis results in generic providence

As argued in Chapter 3, foreseeing *life-specific* interactions between an artefact and life-phase systems is a necessity for $D_sF\Sigma X$. As Table 4.1 indicates, with current means, designers are not generally supported in concurrently generating an artefact and a number of associated life-phase models. For instance, DFM guidelines do not reflect the actual manufacturing concerns of the user [Molloy et al. 1993] that the artefact will encounter. A tool which attempts to address this issue is MIDAS [Bonfield et al. 1997] (See

Appendix B). MIDAS allows designers to specify available production facilities. However providing knowledge about such life-phase systems with a 'fixed perspective' [Kerr 1993] is not enough. For example, it is misleading to assume that LCCs associated with a standard mould part (e.g. a core-pin) will be the same when purchased from different suppliers. Thus, current means lack to support 'artefact life specific' providence - rather they provide generic knowledge on artefact life issues.

Inadequate Providence Due to Lack of Exploration Support

Lack of supporting multi-material domain exploration

Current means (e.g. DFX guidelines and KBS) tend to be material domain specific. This makes it difficult for designers to explore alternative domains *during* synthesis. Also, current means focus on supporting the generation of an artefact solution to fit an assumed life-phase system. Hence designers lack tools that support 'artefact life exploration'. Since, as argued in section 3.2, exploration generates useful LCC knowledge, then with current means, providence is not being adequately exploited to support $D_sF\Sigma X$.

Providence Not Exploited For Life-Oriented Design Guidance

Lack to provide life-oriented design guidance knowledge

Current means exploit providence to provide artefact life knowledge for *improving* a candidate design solution (see for example [Oh et al. 1995; Changchien et al. 1996; Kim 1997] in Appendix B). They do not adequately exploit providence to provide knowledge for *guiding* artefact solution synthesis. For example, *when* defining the use of fasteners as part of a candidate solution (Figure 4.11), current means do not generally support designers in foreseeing feasible assembly system alternatives and their associated requirements. Ideally, when selecting say a robot assembly system from the alternatives 'foreseen', support tools should provide relevant guidance knowledge (e.g. to introduce counter-sunk holes to facilitate fasteners' insertion [Willemse 1997] by the robot's gripper) to aid harmonizing the evolving artefact solution and the assembly system.

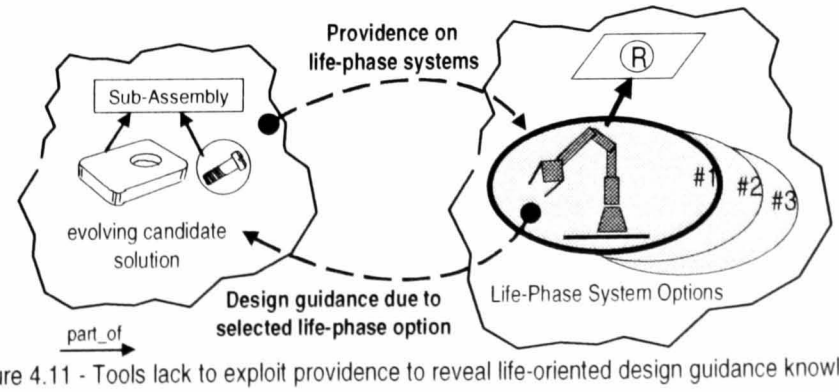


Figure 4.11 - Tools lack to exploit providence to reveal life-oriented design guidance knowledge

Providence Difficult To Achieve

Lack of pro-active providence support

Means indirectly supporting providence, such as FMEA and rapid prototyping contribute to the generation of life-oriented design solutions by motivating designers to consider revealing life issues associated to the artefact solution. However, the major drawback of these type of providence means is that the ability of revealing artefact life issues depends on the users of these means. Further as outlined in Table 4.1, these means require a team based approach. On the other hand, means directly supporting providence can provide access to a large volume of captured artefact life knowledge, in various forms such as different *DFX guidelines* [Fabricius 1994; Seliger et al. 1994], or a *number* of knowledge based systems e.g. [Victor et al. 1993; Meerkamm 1994]. However, utilizing knowledge embedded in these means to foresee artefact life issues requires user interaction to decide *which* guideline or tool is applicable for a given design scenario. That is, designers lack *pro-active* support enabling them to *utilize* the *right artefact life knowledge* at the *right time* and thus to effectively foresee life-oriented design issues.

Manual versus Computer Based Means To Providence

Computers help overcome limitations of the human brain

Teams have a number of drawbacks such as that of 'interim isolation' and the 'loss of valuable expertise' that influences the effectiveness of indirect, manual means such as QFD and FMEA. At the same time, designers have high demands placed on their thinking ability [Pahl et al. 1996] and the human brain has knowledge processing limitations [Ellis et al. 1989; Kerr 1992]. Thus, as argued by Ishii [Ishii 1991], any computer environment that helps engineers to incorporate life-cycle values plays an important role. In this sense, a suitable avenue to supporting life-oriented design from the perspective of foreseeing artefact life interactions during design, are computer based means that directly support providence.

4.5 Chapter Conclusions

Current state of Indirect and direct means support to providence

This chapter has presented a review of a number of the main means that are currently used to *indirectly* or *directly* support 'providence' - foreseeing artefact life issues during design. The review reflects that different means have different strengths and weaknesses to supporting providence. In this sense, a hybrid of these means could amplify the effectiveness of support to life-oriented providence. Table 4.1 reflects that the means reviewed contribute

to supporting designers in having an insight into artefact life issues during the design phase, before the design solution is released for realization. However, the review has established that individually, the means reviewed exhibit one or more of the following limitations. Providence:

- takes place *late* during candidate solution analysis, not during synthesis;
- covers a *narrow* and *segmented (Single-X)* rather than a multiple, life-span (Multi-X) view
- is *generic* and not 'life-specific', as life-phase solution modelling is not supported;
- is not exploited for life-oriented design guidance as *artefact life exploration* is not supported.

A gap in providence means

This chapter therefore concludes that for $D_sF\Sigma X$, there is a need for a means that collectively (Figure 4.12) supports designers to foresee and explore during solution synthesis, multiple, artefact 'life-specific' interactions, in order to adequately handle the phenomena of propagation effects at the right time.

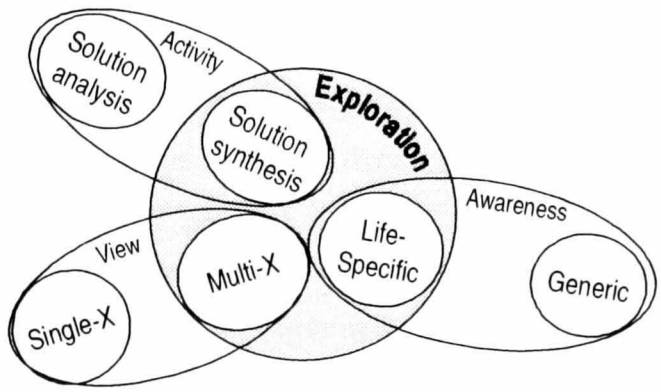


Figure 4.12 - Gap in providence means supporting $D_sF\Sigma X$

chapter 5

5.0 Established Research Problem

Scope

This chapter discloses the research problem established from the preceding chapters. For this purpose, section 5.1 presents the main outcomes of Chapters 2, 3 and 4. Based on these outcomes, section 5.2 presents the established research problem and the research questions arising from this problem. Section 5.3 then presents the Ph.D. research boundary, with the dissertation's Part A conclusions made in section 5.4

5.1 Research Problem Foundation

Chapter 2 - Designers need to $D_sF\Sigma X$

Chapter 2 established that during the life of mechanical artefacts, the *interaction* between an artefact and different life-phase systems results in a number of consequences. Thus, knowledge of such consequences is generated during such interactions. Due to the chronological order of an artefact's life, such knowledge is however not readily acquired by designers. Decisions made during the design phase can thus influence these interactions. As examples presented reflect, design decisions can result in intended and unintended consequences that can propagate across *multiple*-life phases, these termed life-cycle consequences (LCCs). As an implication of this propagation effect phenomena, this research argues that the designers' responsibility covers *all* life-phases. To generate 'life-oriented' design solutions, handling this phenomena is a necessity. Thus, designers need to adopt a $D_sF\Sigma X$ approach.

Chapter 3 – For $D_sF\Sigma X$, designers need to foresee life-cycle consequences

As argued in Chapter 3, in order to be guided in generating life-oriented solutions, a 'Design Synthesis for Multi-X' approach requires that designers engage in provident thinking as from early design, in order to foresee LCCs co-evolving with the artefact solution. This however requires that a large amount of distributed LCC knowledge is *acquired*, *is readily available* and *easy to access* to permit it to be explicitly *utilized* during design.

Chapter 4 – Designers lack adequate providence means

Chapter 4 disclosed that distributed artefact life actors are too busy to continuously form part of a design team to support life-oriented providence. Thus, due to the lack different team members and hence the knowledge they

possess, design decision making takes a *narrow* focus, with designers *frequently unaware* of a number of unintended LCCs being generated. At the same time, the review in Chapter 4 established that designers lack appropriate means supporting life-oriented design from the perspective of providence.

5.2 Research Problem

LCC Knowledge modelling & explicit utlization

Thus, in order to adequately handle the phenomena of propagation effects at the right time, this Ph.D. argues that there is a need of a LCC knowledge intensive means that allows designers to foresee during synthesis, mutliple, artefact 'life-specific' interactions, in order to *guide* them in generating life-oriented design solutions. Given that human beings have mental, knowledge processing limitations, the Ph.D. research problem is therefore concerned with:

- Developing a *computational framework* that collectively allows designers to interact with a LCC knowledge model, when generating and describing mechanical artefact solutions, in order to explicitly foresee *multiple* LCCs co-evolving with the decisions being made.

Lack of design for multi-X (DFΣX) knowledge

Using segmented DFX knowledge 'as-is' during tool implementation, can result in conflicts between a guideline in one *X-area* and another in a different *X`-area*. This is as 'design for' guidelines do not cross-correlate [Pugh 1991]. Thus, to develop such a computational means, one needs to distinguish between the integration of different DFX knowledge i.e. Σ(DFX) and knowledge that explicitly models interactions between different 'X' i.e. DFΣX. As literature reflects, there is indeed an abundance of DFX type knowledge for various domains, but an apparent lack of DFΣX knowledge.

Need to identify 'what' to model

In developing knowledge intensive computational tools, there is a need to distinguish between *'what'* needs to be modelled versus *'how'* it is represented [Tomiyama 1996]. For example, the review in Chapter 4 reflects that KBS have various characteristics that make them suitable for supporting life-oriented design. They also provide different formalisms to 'how' knowledge can be represented. However, their application to effectively support life-oriented providence, depends on 'what' knowledge is actually embedded and how readily this knowledge can be utilized at the right time during synthesis. Thus, a key issue in developing such a computational means, is that of LCC knowledge modelling in order to support the utilization of the *right* knowledge at the *right* time during solution synthesis.

Need to consider knowledge maintenance issues

The fast pace of technological changes require knowledge to be updated regularly [Gupta et al. 1997]. Most computational means supporting providence fail to adequately support knowledge maintenance, which is more than being able to add/modify rules. This is an issue that needs to be addressed if knowledge intensive tools are to remain useful [Duffy 1997]. Some tools like MIDAS [Bonfield et al. 1997], go some way towards addressing this problem but most tools focus on supporting designers at the expense of knowledge maintenance.

Knowledge modelling needs a LCC phenomena model

As disclosed in Chapter 4, current means supporting providence basically require a candidate solution *prior* to providing an insight into LCCs. Further, knowledge revealed tends to be *generic, narrow* and *segmented.* This research considers this state of providence support as reflecting a lack of understanding of the phenomena by which LCCs are generated and propagated across *multiple* life-phases. Therefore, in correlation with the *computational means development framework* illustrated in Figure 1.3, developing a computational LCC knowledge intensive means to $D_sF\Sigma X$, requires first an understanding of the phenomena of how LCCs are generated.

Research Questions

Therefore, the research problem presented above introduces a number of research questions that have to be addressed in this research:

- how are LCCs generated and propagated from design decisions?
- what artefact and life-phase elements are relevant for a knowledge model whose purpose is to infer artefact LCCs?
- how are the relevant elements and LCCs related?
- how can such relationships be transformed into meaningful knowledge?
- how can this knowledge be operated upon by designers, to allow them to infer at the right time, LCCs co-evolving with their solution description?
- how can this knowledge be *organized* and *codified into computable form*, to cater for maintenance difficulties?

5.3 Ph.D. Research Boundary

Research boundary

In order to bound the established research problem, this thesis focuses on developing a computational framework providing designer support to $D_sF\Sigma X$ within the following context:

Component level

- during synthesis at the component level. A mechanical component is composed from a single material. This reduces the number of elements and relationships that need to be established. However, component design is still decision intensive and therefore an area where (component) life exploration is beneficial. The relevance of supporting component $D_sF\Sigma X$ is also evident from arguments made by others, such as "...qualities and problems of production are related (caused by) to the components...." [Andreasen et al. 1993a] and "Component designers can contribute significantly to the serviceability of the products that use their components" [Bralla 1996];

Design type

- during adaptive and variant (see section 3.1.1) mechanical component design scenarios;

Design stage

- during component conceptual design (see section 3.1.2), when the component solution is abstract, undetailed and still evolving;

Synthesis viewpoint

- during synthesis taking place mainly from a *constructional domain* (see section 3.1.3) viewpoint;

Individual support

- to 'individual' designers in foreseeing and exploring LCCs when generating mechanical component design solutions.

5.4 'Part A' Conclusions

Main problem is developing a framework for LCC knowledge modelling & utilization

This chapter, which concludes 'Part A' of this dissertation, has presented the research problem and boundary. The research problem concerns modelling LCC knowledge in such a way to allow it to be utilized during synthesis when the artefact solution is still being described, for pro-actively guiding designers in generating life-oriented design solutions. The boundary of this thesis is to supporting synthesis at the component level from a constructional point of view. This thesis will now proceed to Part 'B', during which the approach framework developed as a means for supporting component $D_sF\Sigma X$ is presented. As a solution foundation to the established research problem, the next chapter presents a phenomena model formally explaining how LCCs are generated from design decisions.

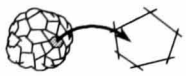
part B

Development of a 'KC' Approach to $D_sF\Sigma X$

chapter

6.0 'Life-Cycle Consequences' Phenomena Model

Scope



To address a question arising from the research problem established in Chapter 5, this chapter presents a phenomena model explaining *how* LCCs are *generated* from design decisions in two distinct ways during component life synthesis. For this purpose, section 6.1 provides a theoretical background to decision-making. Section 6.2 introduces the concept of artefact and life-phase synthesis decision commitments to describe a synthesis decision commitment model. Section 6.3 presents the LCC generation phenomena model established whilst section 6.4 discusses the significance of this model for developing a knowledge intensive approach framework for $D_sF\Sigma X$. Chapter conclusions are made in section 6.5.

6.1 Decision-Making Background

Decisions concern alternatives in a domain

As a foundation to explaining how LCCs are generated, this section presents relevant theoretical background to decision making. As argued in Chapter 3, the space of *alternatives* available in the solution space, make design, decision intensive irrespective of the design stage or synthesis viewpoint. Alternatives are known to be an essential feature of any decision – strictly speaking, without alternatives, there would be no decision [Holtzman 1988]. This is also the case with design decision-making [Mistree et al. 1993; Roozenburg et al. 1995]. In this Ph.D. research, a decision is therefore assumed to exist due to a selection between a number of *alternatives* related to some *domain* [Holtzman 1988]. Deciding on what material to use for a component is a decision in a different domain than say, deciding what assembly system to specify in the realization phase. In the case of components, the designer encounters alternatives due to degrees of freedom with respect to the manipulable characteristics [Tjalve 1979]. These include *reusable* PDEs such as form features [Nnaji et al. 1991], assembly features [Salomons et al. 1993], material and surface textures, a few of which are schematically illustrated in Figure 6.1. The alternative selected, a result of the decision-making process, is said to be the *decision commitment*.

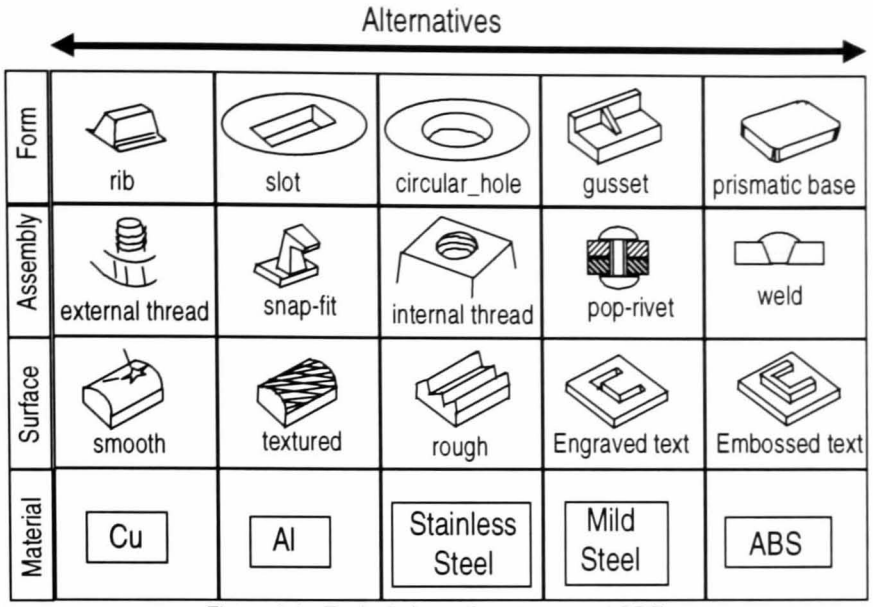


Figure 6.1 - Typical alternative component PDEs

Intended consequences

A decision commitment is subject to a corresponding *outcome* [Holtzman 1988]. That is, decision making is *consequential* in the sense that selecting an alternative (the commitment action) depends on anticipation of the future effects of alternatives being considered. A decision is made to *intentionally* achieve a *desired* consequence, termed the decision goal [Roozenburg et al. 1995]. Different decision commitments result in different consequences. Thus, alternatives are interpreted by the decision-maker, in terms of their *expected* consequences [March 1994].

Unintended consequences

Decision making theory assumes that decision-makers have perfect knowledge of alternatives and their expected consequences [March 1994]. However, studies of decision making in the *real* world suggest that decision-makers do not always know *all* the consequences of their alternatives [March 1994]. That is decision commitments also result in *unintended* consequences.

Uncertainty concerning consequences

In pure theories of choice, decision-makers are assumed to choose among alternatives on the basis of their expected consequences, but such consequences are not known with certainty [March 1994]. In some situations, the consequences depend not only on the chosen alternative, but also on factors outside the control of the decision-maker or on decisions of others [Roozenburg et al. 1995]. Some uncertainty therefore always exists in decision situations including design [Joshi 1991].

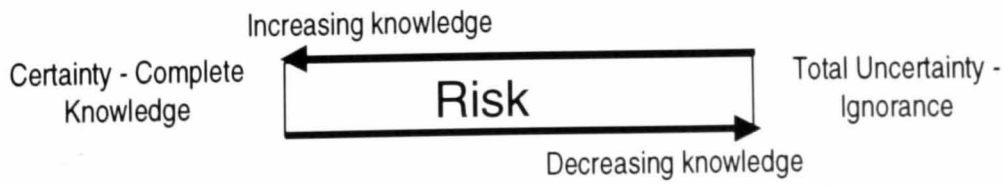


Figure 6.2 - Decision making zones - adopted from [Turban 1993]

Hence, the amount of knowledge concerning decision consequences available *during* decision-making, which ranges from *complete knowledge* to *ignorance* (Figure 6.2), results in [Turban 1993; March 1994]:

- *decision-making under certainty* – the (intended) consequence for each alternative is known, resulting in a deterministic situation;
- *decision-making under risk* – several possible consequences need to be considered for each alternative. The probability of such outcomes can be estimated, resulting in a probabilistic situation – a calculated risk;
- *decision-making under uncertainty* – insufficient information makes it difficult to estimate the probability for consequences of different alternatives.

Circumstances & preferences

Decision-makers take into consideration specific *circumstances* (e.g. constraints which restrict the possible set of alternatives) and *preferences* (e.g. designer's risk attitude) [Holtzman 1988]. Thus in decision-making, alternatives are compared in terms of how much their expected consequences serve the circumstances and preferences of the decision maker [March 1994].

Multiple criteria for comparing alternatives

To compare alternatives, one may consider either *single* or *multiple* criteria. In artefact design, the latter is always the case [Roozenburg et al. 1995; Sen et al. 1995], even more so in life-oriented design. For instance, the selection of an assembly feature needs to consider consequences of alternatives, such as ease of its realization, ease of artefact assembly, ease of dis-assembly during service and disposal. Multiple criteria decision-making techniques, which can be quantitative or qualitative, assume that criteria are known in advance of their use [Roozenburg et al. 1995]. However, due to the phenomena of propagation effects, different alternatives can co-evolve different requirements. Using these co-evolved requirements as criteria when selecting between alternatives is beneficial in DFΣX but poses a difficulty – knowledge of such co-evolved criteria needs to be revealed in order to be considered *during* decision making.

Decision-making influenced by number of actors involved

The foregoing arguments explain why the people (and number of) involved in decision-making can influence *which* alternative is chosen, due to possibly conflicting goals [Roozenburg et al. 1995], their *body of knowledge* and their risk attitude [Holtzman 1988].

Utility of LCC knowledge

In the context of design, many decisions made by a designer are choices that 'feel right' in the absence of evidence to the contrary [Bowen 1991]. As argued in Chapter 4, in the interim period between design review meetings, designers work in isolation. At the same time, as human beings, designers have limitations concerning knowledge possession and processing [Ellis et al. 1989; Kerr 1992]. This, highlights why explicitly providing designers with knowledge of unintended LCCs resulting with their commitments, is beneficial to supporting them in selecting life-oriented alternatives.

6.2 Synthesis Decision Commitments

Based on the decision-making background provided, it can be seen that every decision has a consequence - even selecting a designer who will execute the design process, or, say a design support tool (e.g. calculator).

Commitments reflected in evolving solution

However, this Ph.D. research focuses on consequences arising from decisions concerning alternatives about the design solution. These commitments, made during synthesis to an evolving solution, are termed in this thesis as *synthesis decision commitments* [Borg et al. 1998]. Thus, unlike other commitments (e.g. about the design process), synthesis decision commitments are reflected in the evolving artefact's solution model (Figure 6.3). As synthesis decision commitments are the result of decision-making taking place *during* design, they form a *link* between the 'design process' and the 'artefact solution model'.

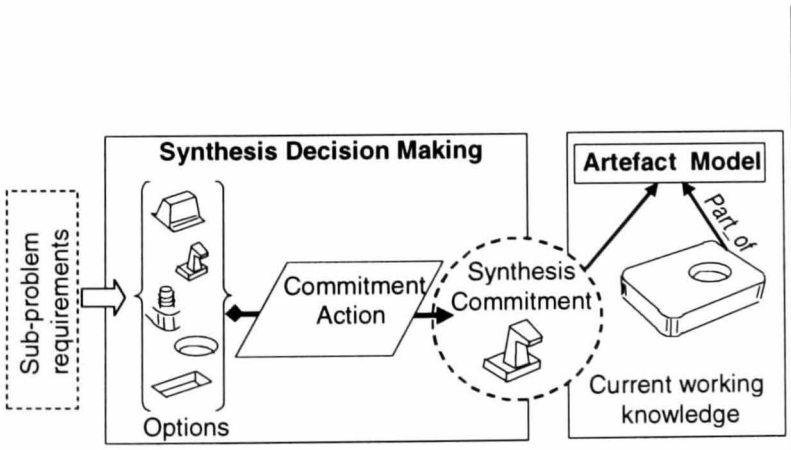


Figure 6.3 - Solution evolution with synthesis decision commitments

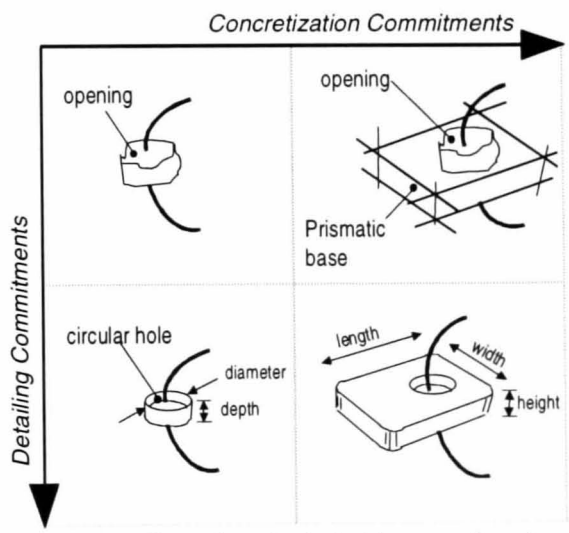


Figure 6.4 - Type of synthesis decision commitments

Concretization & detailing commitments

As the theory of domains (section 3.1.3) explains, during design, various navigations take place. Thus, as reflected in Figure 6.4, these commitments result in a more concrete solution if they are *concretization commitments* (e.g. committing an opening form feature) or a more detailed solution if they are *detailing commitments* (e.g. committing a value for the diameter). This

reflects that an expansion in the artefact's *current working knowledge* [Zhang 1998] is caused by synthesis decision commitments.

Significance of synthesis decision commitments

The same synthesis decision commitment can be made for *different* intentions. For example, intention 'a' (Figure 6.5) is to select a hole to permit a wire to pass through during the use phase, whilst intention 'b' is to select a hole for allowing a fastener to be inserted during the realization phase. That is, one alternative (a hole) in the option set, can be used for '*n*' ($n \geq 1$) different intended consequences. Since synthesis decision commitments become part of the artefact solution, then designers should be also concerned with *unintended LCCs* arising from such decision commitments.

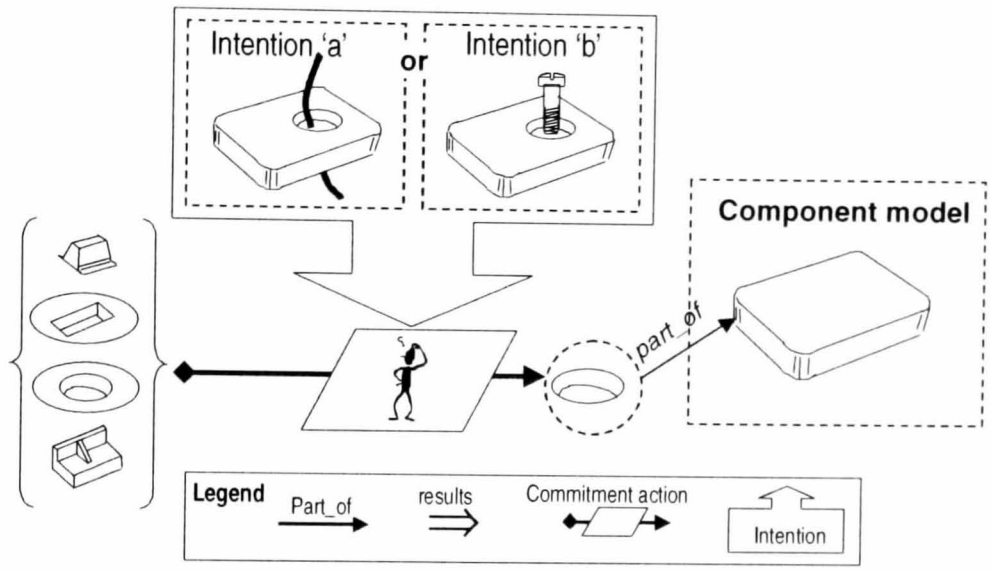


Figure 6.5 – Different intentions achieved with same synthesis decision commitments

6.2.1 Synthesis Decision Commitment Consequences

Scenarios arising from synthesis decision commitments

As argued in Chapter 2, the purposeful transformation brought about in a phase can cause both *desired* and *undesired* effects. One can thus visualize a number of scenarios resulting from *synthesis decision commitments* purposely made during the design phase. Consider Figure 6.6, where it is not shown how the set {} of solution alternatives is generated. Figure 6.6a depicts the case when given a set of alternative solutions, a designer makes a synthesis decision commitment to purposely achieve an *intended* good (✓) consequence. An example is when the designer selects a thermoplastic material for a component to ensure that it does not conduct electricity during its use phase. A different scenario is when a designer purposely makes a decision commitment to result in a problematic (✗) consequence (Figure 6.6b). This is a scenario which although possible, professional and ethical engineers should not resort to. Figure 6.6c demonstrates a scenario in which a designer purposely makes a decision commitment to achieve an intended, good consequence, but which also results in an *unintended* problematic

consequence. An example is the commitment to define an ultrasonic weld to join two thermoplastic components made from different materials. The good and intended consequence is to keep the number of parts in the sub-assembly to a minimum, thus abiding with a DFA guideline. The unintended (to this designer), problematic consequence is that the components will be permanently bonded, making dis-assembly difficult for maintenance and recycling purposes. A designer can also make a decision commitment to purposely achieve an intended, good consequence, but which simultaneously results in an additional, unintended good consequence (Figure 6.6d). An example is selecting a snap-fit for joining two thermoplastic components, making dis-assembly possible. This also results in a good, unintended (to this designer) consequence - the number of parts are kept to a minimum thereby abiding with a DFA guideline. Another possible scenario not illustrated in Figure 6.6 is a hybrid of (c) and (d).

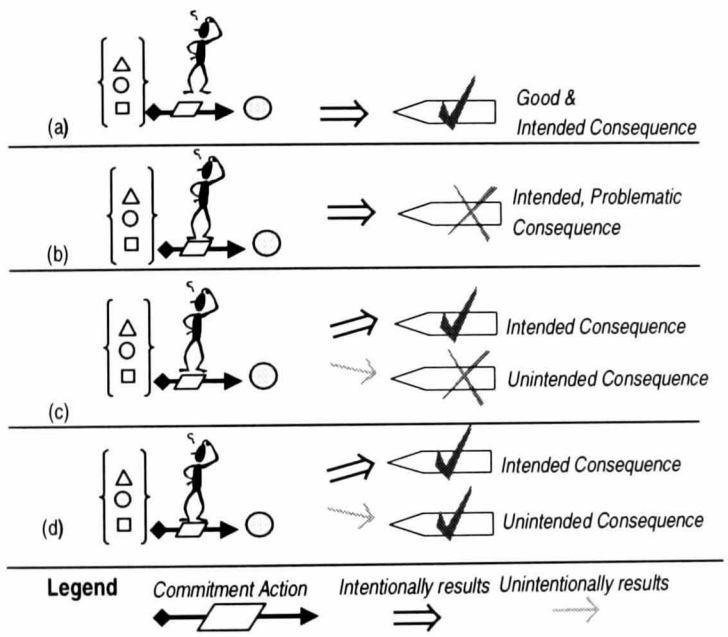


Figure 6.6 - Synthesis commitment consequences

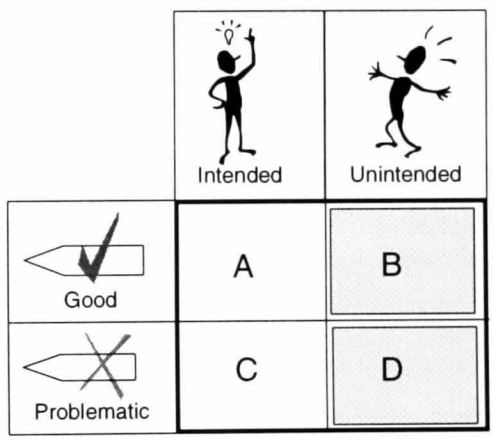


Figure 6.7 – Synthesis commitment, consequence dimensions

As these scenarios highlight, whether a consequence is intended or unintended depends on the designer's intention and knowledge of consequences associated with the different alternatives. Also, whether a consequence is considered good or problematic is subject to the criteria used by the designer.

Consequence Dimensions

Significance of revealing unintended consequences

The foregoing arguments mean that consequences arising from synthesis decision commitments can be classified in the *dimensions* shown in Figure 6.7. The *unintended consequences* ('B' and 'D') are those consequences of which the designer is *not explicitly aware* when making synthesis decision

commitments. Commitments intentionally made to the solution model, may thus result in unintended LCCs. Thus, it is such *unintended* LCCs, which a designer needs to be made aware. This is because, if an unintended LCC is *explicitly* made known to the designer and is then considered:

- *good*: it provides additional knowledge that may assist a designer to reinforce the commitment being made;
- *problematic*: it provides additional knowledge that may motivate a designer to explore other alternatives to avoid or relax LCCs revealed.

6.2.2 Artefact Synthesis Decision Commitments

As a basis for the phenomena model being introduced later in this chapter, this section presents a model of synthesis decision-making (Figure 6.8). This model, on which the 'KC' approach to $D_sF\Sigma X$ developed in this research is based, is founded upon decision-making theory [Holtzman 1988; March 1994] and the basic design cycle [Roozenburg et al. 1995]. The *notation* (e.g. D{O}) used for explanation purposes is defined in Appendix C.

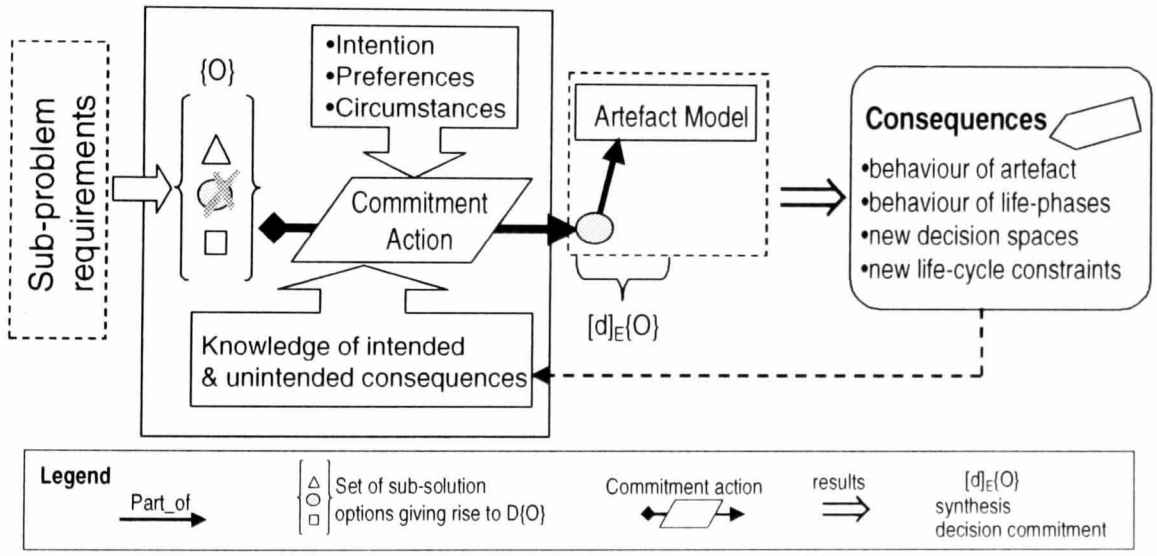


Figure 6.8 - Synthesis decision commitment model

During synthesis, the designer *generates* a set {O} of possible solution options to the sub-problem being tackled. This set gives rise to a decision proposal D{O}. Based on arguments made in Chapter 3, the set of options {O} consists of reusable PDEs. For variant and adaptive design, solution space options generated are known at the time of synthesis decision making. Thus, they are included in the set of the decision proposal D{O}. For original design, *new* alternatives (e.g. a new material), must be first generated and then included with the decision proposal D{O}. Nevertheless, the designer needs to select an *alternative* from the set of options available. Thus, the designer engages in a decision-making process (DMP) to make a selection. During the DMP, the

designer considers *intentions*, *preferences* and known *specific circumstances*. Following these considerations, the designer selects an alternative by making a synthesis decision commitment $[d]_E\{O\}$ to the evolving solution model.

Artefact minimum synthesis commitments

During early component design, due to a lack of complete information, the designer may only be in a position to make a *minimum commitment* [Guan et al. 1995]. For instance, the designer might not be sure or concerned with whether to select a snap-fit or screw as an assembly feature. Rather the designer may prefer to make a minimum commitment by specifying a 'non-permanent bond' to the artefact model. In this case, the decision commitment $[d]_E\{O\}$ is a minimum commitment rather than a specific commitment. For example, in defining a form feature, a typical scenario is:

D = which form feature type?

with { [F] } = { ⌊Opening⌋ ∨ ⌊Protruding⌋ }

and D { [F] } resulting in say $[d]_E\{[F]\} = \lfloor Opening \rfloor$. *Then,*

S= {D'} = { which type of opening feature?}

This will eventually result in the need for a more specific commitment such as:

D' = which type of opening feature?

with { [F'] } = { [hole] ∨ [rectangular_hole] ∨ [poly_hole] }

and D'{ [F'] } resulting in say $[d]_E\{[F']\} = [hole]$. *Then*

$S' = \{$ *what* $[hole]_{radius}$? $\wedge$ *what* $[hole]_{depth}$? $\wedge$ *what* $[hole]_{angle}$? $\wedge$ ... *what fabrication process to generate hole?}*

This example demonstrates how a minimum synthesis decision commitment can result in 'a new decision space'. A different synthesis commitment, say $[d]_E\{[F]\} = \lfloor Protruding \rfloor$, would eventually result in different consequences including a different new decision space.

Artefact Synthesis Commitment Exploration

Exploratory synthesis commitments

If the co-evolving *LCC knowledge* is explicitly revealed and utilized *during* synthesis decision-making (Figure 6.8), it provides *additional* knowledge that can be employed for comparing alternatives in terms of their LCCs. If made aware of unintended consequences, the designer may retract the synthesis

commitment to explore others. Thus, solution generation and exploration inherently involve the commitment and retraction of synthesis commitments. During exploration, an alternative is only *partially removed* from the set of options forming part of the decision proposal D{O}. This is because it may be found, after *exploring* the LCCs associated with other alternatives that it is after all the most feasible option. This means that it will be re-committed to the evolving solution.

6.2.3 Life-phase Synthesis Decision Commitments

Life-phase commitments

As argued in Chapter 3, $D_sF\Sigma X$ requires that designers, concurrently generate and model the *artefact solution* and *life-phase system solutions*. These models collectively form an *artefact life model*. During concurrent synthesis, synthesis decision commitments therefore concern both the *artefact model* and the different *life-phase models* such as those for realization and disposal. Life-phase synthesis decisions concern for instance the selection of technical processes (e.g. milling) and technical process parameters (e.g. feedrate). Life-phase synthesis decision commitments can also result in both intended and unintended, good and problematic consequences. For example, selecting an ultrasonic bonding system for the realization phase, keeps the number of components in the artefact structure to a minimum, but makes the permanently bonded components difficult to separate during disposal.

Life-phase solution exploration

The synthesis decision commitment model presented in Figure 6.8 is thus applicable to life-phase system synthesis and exploration. In this case, the model being generated and explored is that of a life-phase system and the decision proposal D{O} concerns alternative LCPEs.

Table 6.1 - Typical Minimum Synthesis Commitments

D	**D {O}**	**d_E {O}**	**Typical Constraints CN{O}**
Component material?	D {[M]} = D{⌊Ferrous⌋ ∨ ⌊Non-Ferrous⌋ ∨ ⌊Thermoplastic⌋ }	⌊Non-Ferrous⌋	CN{O}= ⌊Non-Magnetic⌋ => {O} = { ⌊Non-Ferrous⌋ ∨ ⌊Thermoplastic⌋ }
Fabrication Process?	D {[P]} = D {⌊Netshape⌋ ∨ ⌊Forming⌋ ∨⌊Profiling⌋ ∨ ⌊Material Removal⌋ }	⌊Netshape⌋	CN{O} = 3-dimensional process only => {O} = {⌊Netshape⌋ ∨⌊Forming⌋ ∨ ⌊Material Removal⌋ }

Minimum life-phase commitments

The principle of minimum commitments also applies to early life-phase solution synthesis. Typical examples of commitments made to the *artefact model* and the fabrication process forming part of the *realization phase model* are given in Table 6.1. As illustrated, constraints CN{O} can *restrict* the set of options {O} that can be considered in the respective decision proposal D{O} .

6.3 Life-Cycle Consequences Generation Model

As argued in Chapter 5, understanding *how* LCCs are generated and propagated during synthesis is essential for developing support to $D_sF\Sigma X$ from the perspective of providence. This is because synthesis decision commitments introduce LCCs, some of which reflect new unintended artefact life specific circumstances that have to be taken into consideration *during* synthesis decision making. As acknowledged in literature [Nowack 1997; Tichem 1997] decisions made during design result in consequences. Nowack explains that a *consequence* is produced by an *action* chosen by a designer when trying to resolve an *issue* (Figure 6.9). The resulting consequence is then used to check if it *fulfils* the issue being addressed. In this sense, the consequence referred to by Nowack is an intended consequence. Unintended consequences are referred to as *collateral consequences* and *new issues*.

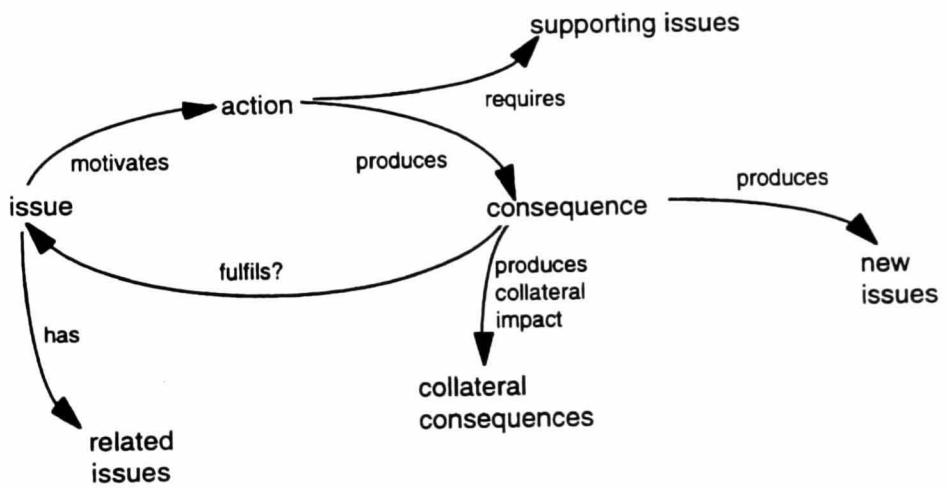


Figure 6.9 - Action-Centred Design Model - [Nowack 1997]

Tichem states that during DFΣX, all relevant life-cycle specifications should be taken into account. This is reflected in his *structure of a design decision* (Figure 6.10). Although in such work it is acknowledged that consequences arise from (i.e. related to) decisions, there seems to be a lack of an explicit explanation as to *how* life-cycle consequences are generated from synthesis decision commitments.

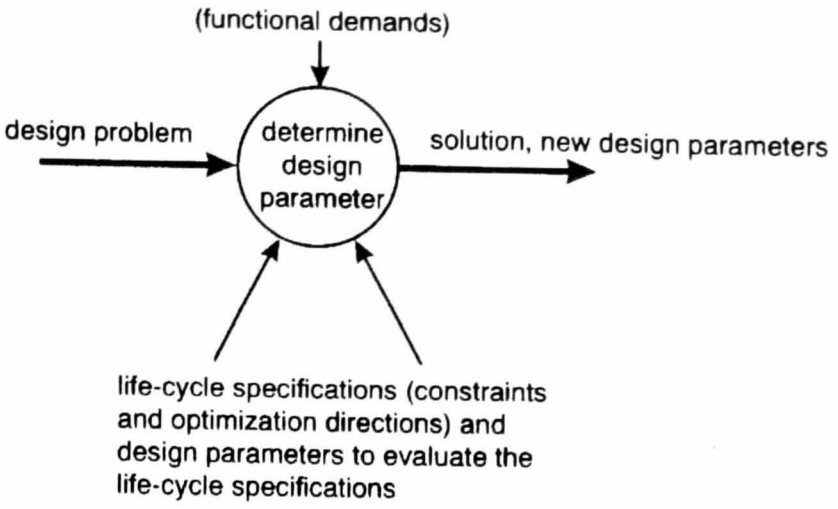


Figure 6.10 – Tichem's structure of a decision - [Tichem 1997]

Consequence generation model

Therefore, based on the theory of dispositions [Olesen 1992], research case-study observations (mainly from the domain of thermoplastic components) and building on synthesis decision commitment model presented in section 6.2, a phenomena model [Borg et al. 1998] describing *how* LCCs are generated, is now being introduced. As this model reveals, LCCs are generated from two fundamentally different conditions:

- individual, *non-interacting* synthesis decision commitments and
- multiple and *interacting* synthesis decision commitments.

6.3.1 Non-interacting Consequences

Generated from Individual synthesis decision commitments

Non-interacting consequences can result during the component's life due to the presence of individual elements (PDEs or LCPEs) in the artefact life model, introduced by synthesis decision commitments made *independent* of other elements (PDE or LCPEs) present in the artefact life model. That is, a LCC of this type is generated by *one* specific element commitment. Considering Figure 6.11, this means that given the set of elements $\{O_1\}$, the option selected by the designer in this case, o_1^1, introduces its associated consequence 'x' of the *non-interacting* type (LCC_{ni}). Formally, this is:

(6.1) $$([d]_E\{O_1\} = o_1^1) \Rightarrow (LCC_{ni} = \text{'x'})$$

A typical example is when a snap-fit (o_1^1) is selected from a set of assembly features $\{O_1\}$. This results in the consequence ('x') that the resulting bond will be weak:

(6.2) $$([d]_E \{[\text{snap-fit}] \vee \ldots[\text{screw}] \vee [\text{adhesive}]\} = [\text{snap-fit}]) \Rightarrow (LCC_{ni} = \text{'weak bond'})$$

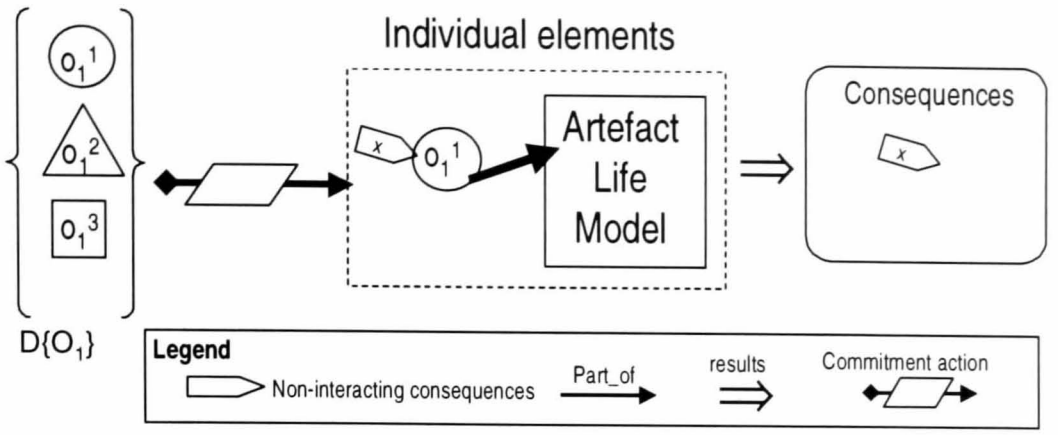


Figure 6.11 - Phenomena model for non-interacting consequences

Individual elements committed can also have other associated LCC_{ni}. For instance, for the snap-fit (6.2), another LCC_{ni} is 'not good for frequent, repetitive assembly and dis-assembly'. Independent of other synthesis decision commitments (e.g. a rib feature) made to the component, these LCC_{ni}s will be generated by a snap-fit. Due to the provision of such important life cycle knowledge, a designer can consider exploring alternative synthesis commitments. Typical examples of different LCC_{ni} are given in Table 6.2.

Table 6.2 – Examples of LCCs resulting from non-interacting commitments

Consequence type	Example	
	$d_E\{O\}$	LCC_{ni}
Artefact behaviour	$([d]_E \{[M]\} =$ ⌊Thermoplastic⌋)	Component does not conduct electricity in the use phase
Life-phase system behaviour	$([d]_E \{[F_a]\} =$ [bolt])	Little time & cost to dis-assemble components during e.g. disposal;
New decision space	$([d]_E \{[F]\} =$ [hole])	{"What value for the feature parameters?" ∧ "what fabrication process to generate the hole?"}
New life-phase constraints	$([d]_E \{[M]\} =$ ⌊Thermoplastic⌋)	Spark erosion (EDM) ∉ {Fabrication processes}

6.3.2 Interacting Consequences

Generated from a set of interacting synthesis decision commitments

Consequences can also result from the *interaction* between a *number* of specific synthesis commitments. In this case, LCCs depend on a set of '***n***' life synthesis commitments, where ***n >1***. As design proceeds, besides the element o_1^1, the designer makes another decision commitment (Figure 6.12) concerning a different set of elements $\{O_2\}$. Assume that in this case, the designer selects o_2^2, this introducing its associated LCC_{ni} 'y' :

(6.3) $([d]_E \{O_2\} = o_2^2) \Rightarrow (LCC_{ni} = \text{'y'})$

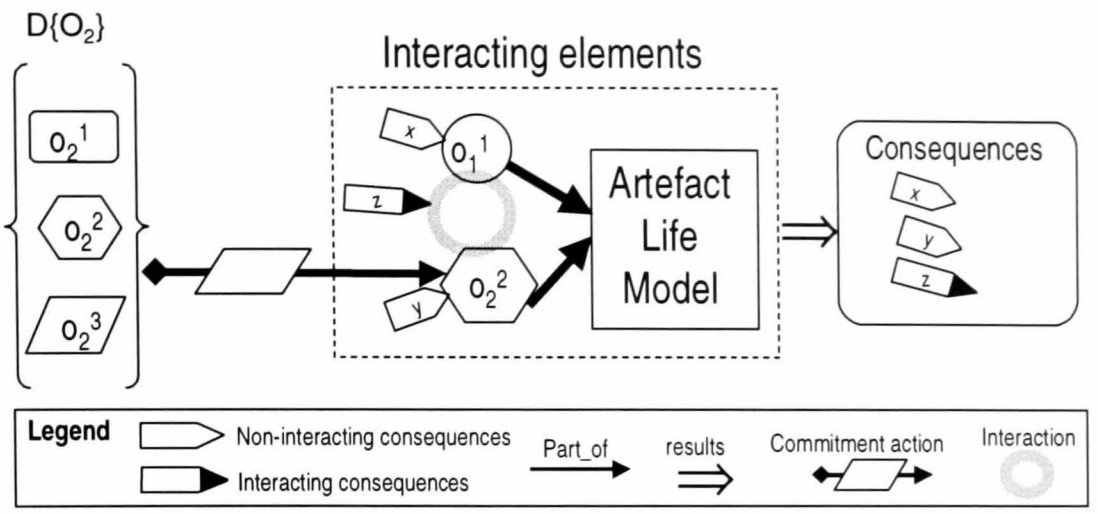


Figure 6.12 - Phenomena model for interacting consequences

An example concerning a set $\{O_2\}$ of *Life Cycle Phase Elements (LCPE)* is the commitment to opt for an injection moulding process (o_2^2). This individually results in the LCC_{ni} (y) that a 'mould is required' :

(6.4) $([d]_E \{[P]\} = [\text{Moulding}]) \Rightarrow (LCC_{ni} = \text{'mould is required'})$

The elements committed i.e. o_1^1 and o_2^2 can sometimes *interact* to give rise to a consequence 'z', this termed an *interacting consequence (LCC_i)*, formally:

(6.5) $([d]_E \{O_1\} = o_1^1) \wedge ([d]_E \{O_2\} = o_2^2) \Rightarrow (LCC_i = \text{'z'})$

A typical example is the interaction between the following set of component life synthesis commitments, which as illustrated in Figure 6.13, gives rise to a sink mark [Groover 1996] defect during the realization phase:

(6.6) $([d]_E \{ [F_a]\} = [\text{snap-fit}]) \wedge ([d]_E \{ [M]\} = [\text{ABS}]) \wedge ([d]_E \{ [P]\} = [\text{Moulding}])$

$\Rightarrow (LCC_i = \text{'sink mark defect'})$

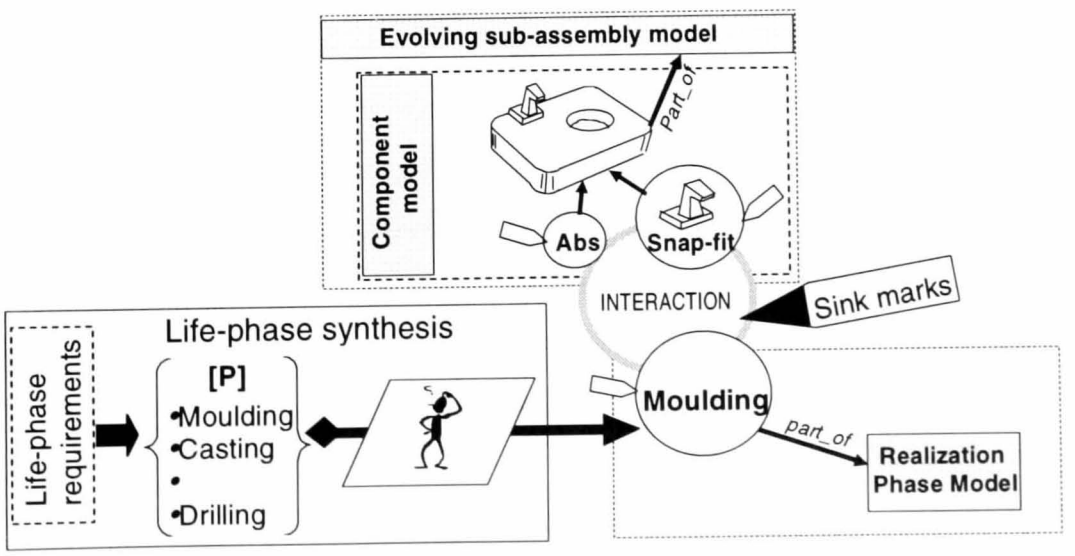


Figure 6.13 – Typical interacting life synthesis commitments

Highlights significance of concurrent synthesis

This phenomena of LCC_i resulting from *interacting* life synthesis commitments, highlights why *concurrent synthesis* of the artefact and life-phase systems is a necessity if designers are to take into consideration a host of total life issues. LCC_i knowledge revealed allows them to explore the avoidance/relaxation of such consequences during solution synthesis. For example, retracting any of the elements ([M], $[F_a]$ or [P]) committed in 6.6 will avoid this specific sink mark. Typical examples of LCC_i are given in Table 6.3.

Table 6.3 – Examples of LCCs resulting from interacting commitments

Consequence type	**Example**	
	Interacting $d_E\{O\}$	LCC_i
Artefact behaviour	($[d]_E$ { [F]} = [hole]) $\wedge$ ($[d]_E$ { [M]} = ⌊Thermoplastic⌋) $\wedge$ ($[d]_E$ {[P]} = [Moulding])	Weak component structure due to weld line defects
Life-phase system behaviour	($[d]_E$ { [F]} = [under-cut]) $\wedge$ ($[d]_E$ { [M]} = [Zinc]) $\wedge$ ($[d]_E$ {[P]} = [die_casting])	Die ejection difficult to achieve
New decision space	($[d]_E$ { [F]} = [slot]) $\wedge$ ($[d]_E$ {[M]} = ⌊Thermoplastic⌋) $\wedge$ ($[d]_E$ {[P]} = [Moulding])	{What mould core parameter values due to material shrinkage factor? $\wedge$ what slot draft angles?}
New life-phase constraints	($[d]_E$ {$[Base]_{radius}$}=0mm) $\wedge$ ($[d]_E$ {[M]} = ⌊Thermoplastic⌋) $\wedge$ ($[d]_E$ {[P]} = [Moulding])	Milling $\notin$ {Fabrication processes for mould cavity construction}

6.3.3 LCC Propagation Mechanism

Propagation mechanism example

Once generated through either non-interacting or interacting commitments, certain LCCs can *propagate* other consequences in the same or a different phase, as the examples presented in Chapter 2 reflect. Understanding how such propagations occur is necessary for generating a LCC knowledge model whose purpose is to infer LCCs propagating across *multiple* life-phases. The following simple example is being presented to explain *how* LCCs can propagate new consequences. Consider the case when *during* thermoplastic artefact solution synthesis, given a set of alternative assembly features {[external thread] $\vee$ [snap-fit] $\vee$ [screw] $\vee$ [adhesive_bond] }, a designer selects a *[screw]*. This commitment results in a number of LCC_{ni}s (Figure 6.14a):

(6.7a) $([d]_E \{[F_a]\} = [screw]) \Rightarrow (LCC_{ni} = \text{'Assembly slow'})$

(6.7b) $\wedge (LCC_{ni} = \text{'hole required'})$

(6.7c) $\wedge (LCC_{ni} = \text{'more parts'})$

(6.7d) $\wedge (LCC_{ni} = \text{'dis-assembly ok'})$

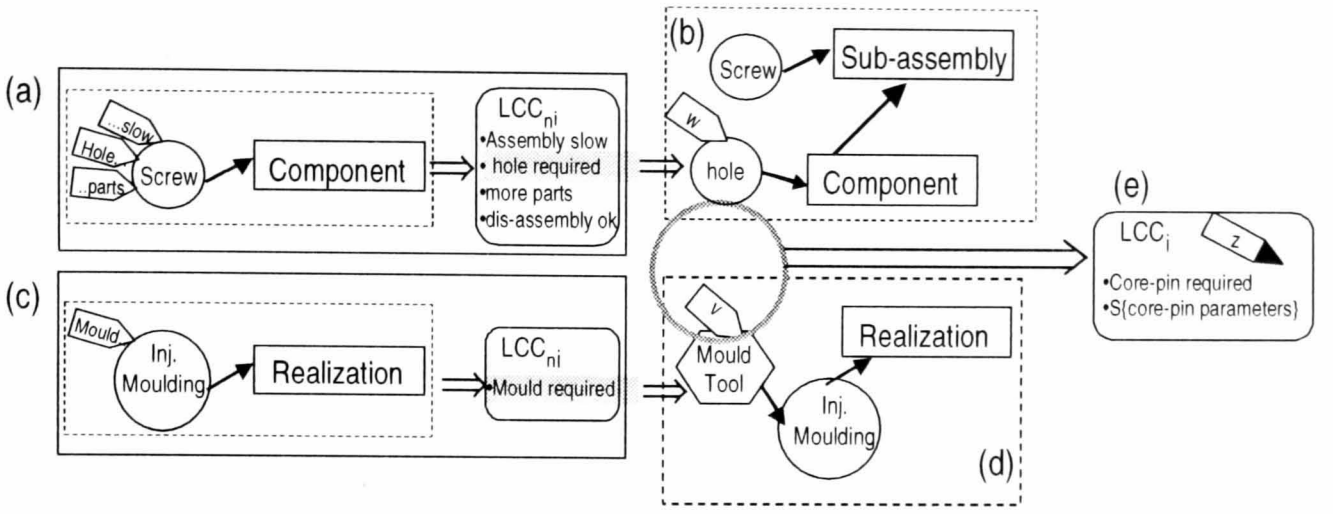


Figure 6.14 - Typical LCC propagation example

These LCC_{ni}s themselves propagate an influence across *multiple* life-phases/life-phase systems as reflected in Table 6.4.

Table 6.4 – Typical Propagation effects For A Screw

	Propagated Consequences	
Consequence	***Life-phase/systems effected***	***Influence***
Assembly slow	Assembly, Service	Time, cost, quality of service
Hole required	Design, Fabrication	Time, cost
More parts	Assembly, Service, Disposal	Time, cost, quality of service
Dis-assembly ok	Service, Disposal	Time, cost, quality of service

Typical propagations

As a result of relation 6.7b, a new PDE (a hole), a *kind of* form feature [F], is added to the component model. Also, due to 6.7c, a sub-assembly model evolves (Figure 6.14b). Assuming that the realizaiton process selected is injection moulding, then during concurrent synthesis, the designer commits an injection moulding system to the realization phase model (Figure 6.14c). From relationship 6.4, injection moulding introduces its own LCC_{ni}, that a '*mould is required*'. This LCC_{ni} results in the commitment of a new LCPE (a mould tool) to form part of the injection moulding system (Figure 6.14d). Now, the mould tool introduces its own set of LCC_{ni}s ('v'):

(6.8a) $(d_E\{[LCPE]\}=[\text{mould tool}]) \Rightarrow (LCC_{ni}=$ 'mould tool needs to be designed') $\wedge$

(6.8b) $(LCC_{ni}=$ 'mould tool needs to be constructed') $\wedge$

(6.8c) $(LCC_{ni}=$ 'form features require draft angle') $\wedge$

(6.8d) $(LCC_{ni}=$ 'component can have parting line defect").

Now, the hole (a PDE) forming part of the component, *interacts* with this mould tool (a LCPE), to propagate a new LCC_i (Figure 6.14e):

(6.9) $(d_E\{[LCPE]\}=[\text{mould tool}]) \wedge (d_E\{[F]\}=[\text{hole}]) \Rightarrow (LCC_i=$ 'core-pin required')

Not shown in Figure 6.14 are other propagations. For instance, the core-pin introduced in 6.9 will introduce a new decision space S = {what $[\text{core-pin}]_{\text{diameter}}$? $\wedge$ what $[\text{core-pin}]_{\text{length}}$? $\wedge$... which supplier etc. ?}. Also, the thermoplastic material *interacts* with the hole and the injection moulding system to propagate a new LCC_i influencing the component's behaviour – a weld line defect is generated during the realization phase.

Propagation mechanism highlights significance of exploration

As the propagation effect mechanism demonstrates, exploring alternative commitments will result in different LCCs and hence propagation effects. If such decision outcomes are unintended, making knowledge of such LCCs and their source commitments explicit, can help *guide* designers to which alternative to select in order to generate a life-oriented design solution. For example, committing an alternative fabrication process such as drilling to generate the hole, avoids the interaction in 6.9 and hence eliminates the core-pin and the LCCs this propagates, such as weld line defects. However drilling introduces different consequences – it is a secondary process and will therefore influence the realization phase's time and cost.

6.4 Significance of The LCC Phenomena Model

Explains how LCCs are generated in two different ways

The phenomena model (collectively shown in Figure 6.15) established is significant for developing a computational, LCC knowledge intensive approach framework to supporting $D_sF\Sigma X$, as it:

- contributes an explanation of *how* LCCs are generated under two fundamentally different conditions – by *non-interacting* and *interacting* component life synthesis decision commitments;

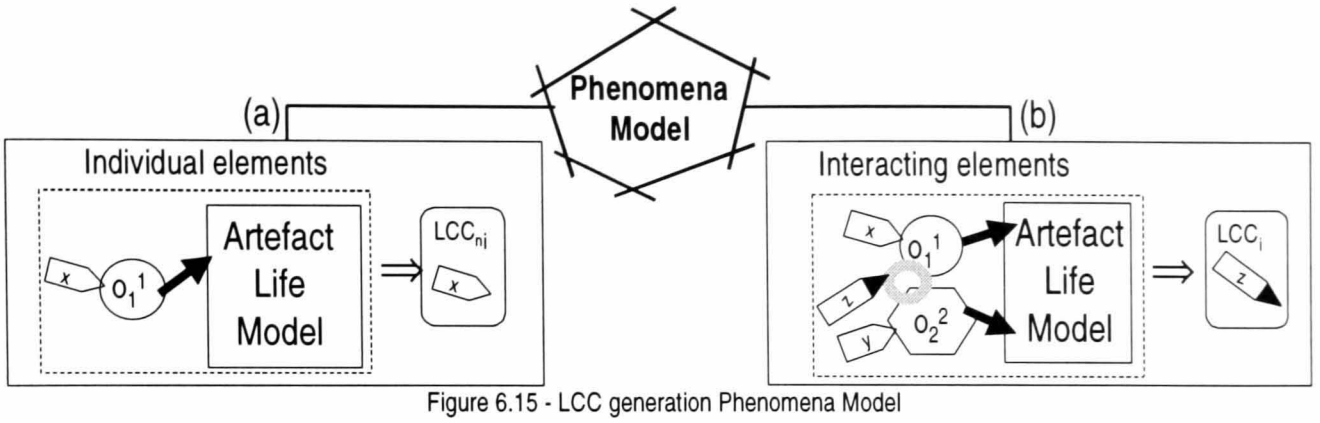


Figure 6.15 - LCC generation Phenomena Model

Highlights the need for concurrent synthesis

- highlights the necessity of concurrent synthesis – for generating a 'life-oriented design' solution, the design phase is *where* component life commitments should be made. Otherwise, it will be difficult to reveal and hence cater for LCC_i *during* component synthesis;

Explains how synthesis-based providence can be achieved

- explains how designers can be explicitly made aware of LCCs *during* synthesis. LCC_{ni} can be revealed with commitments made to an evolving solution. LCC_i can be revealed when *all* commitments of the relevant set of interacting commitments have been made to the still evolving solution.

Provides a foundation for LCC knowledge modelling and utilization

The phenomena model is thus significant for LCC knowledge *modelling* as it:

- provides a basis as to *what* should be captured to generate a LCC knowledge model – the *source* of LCCs are synthesis decision commitments. In the case of component concurrent synthesis, these concern PDEs and LCPEs;

- provides a basis for modelling *causal relationships* between synthesis decision commitments concerning PDEs or LCPEs and LCCs;

Further, it reflects how such a LCC knowledge model can then be *utilized* as a means for :

a) *LCC awareness*: depending on commitments made, LCCs (including those unintended) can be causally revealed during synthesis;

b) *solution synthesis guidance*: designers can utilize the knowledge model to identify which elements can be committed in order to result in intended consequences e.g for easy dis-assembly during the artefact's life, which assembly feature can be selected?

c) *avoiding/relaxing the formation of unintended LCCs:* designers can be guided to *which* LCC-specific commitments (interacting or non-interacting) can be retracted and explored to avoid/relax a LCC.

6.5 Chapter Conclusions

A phenomena model explaining LCCs generation

This Chapter presented a phenomena model explaining *how* LCCs are generated from two fundamentally different conditions - either from *non-interacting* or, from *interacting* synthesis decision commitments. By focusing on synthesis decision making, this chapter explained how both artefact and life-phase models become more concrete or detailed with synthesis decision commitments. Knowledge of the consequences associated with different alternatives, plays an important role in in selecting between alternatives. As in reality, not all the consequences associated with an alternative are known *during* decision making, synthesis decision commitments, result in both *intended* and *unintended* consequences. This explains *why* supporting designers in *explicitly* becoming knowledgeable of the co-evolving LCCs and their propagation *during* synthesis decision making is beneficial for $D_sF\Sigma X$.

Significance of phenomena model

The phenomena model highlights *why* concurrent artefact and life-phase synthesis is necessary if, LCCs of the interacting type, are to be taken into consideration *during* synthesis. Thus it provides a foundation to *what* should be modelled and related to generate a LCC knowledge model. In the case of components, concurrent synthesis decision commitments concern PDEs and LCPEs used in generating artefact and life-phase solutions respectively. Further, the mechanism explaining how LCCs are generated, provides a foundation to how co-evolving LCCs can be *explicitly* revealed when designers employ such a LCC knowledge model during synthesis decision making.

Exploiting the phenomena model to support $D_sF\Sigma X$

Therefore, as a means of supporting $D_sF\Sigma X$ from the perspective of providence, the next chapter exploits these explanations to develop the concept of an approach aimed at making designers explicitly aware of LCCs co-evolving and propagating with their synthesis decision commitments.

chapter 7

7.0 A 'Knowledge of LCC' Approach Framework To $D_sF\Sigma X$

Scope

The aim of this chapter is to disclose the foundation of how to model and utilize LCC knowledge for supporting $D_sF\Sigma X$. For this purpose, this chapter proposes a 'Knowledge of life-cycle Consequences (KC)' based approach to $D_sF\Sigma X$. To outline *why* a LCC knowledge intensive approach is being proposed, section 7.1 discusses the role of timely utilizing LCC knowledge co-evolving during synthesis decision making. Further, section 7.2 discusses why due to human mental processing limitations, designers have difficulties in the utilization of the co-evolving LCC knowledge. Using this background, section 7.3 presents the frames that collectively describe the 'KC' approach framework to $D_sF\Sigma X$. Section 7.4 discloses how the realization of the framework can be used for $D_sF\Sigma X$. Arguments as to why the framework needs to be realized as a *Knowledge Intensive CAD* tool so as to overcome designer limitations are presented in section 7.5. Chapter conclusions are made in section 7.6.

7.1 Timely Utilization of Co-evolving LCC Knowledge

Co-evolving Nature of LCC Knowledge

Current working knowledge in concurrent synthesis

As disclosed by the phenomena model, LCC knowledge is implicitly co-evolving with *concretization* and *detailing* synthesis decision commitments being made to the artefact life solution (Figure 6.8). Also, it was argued that concurrent synthesis is a necessity if LCC_i are to be revealed during synthesis. Hence, based on arguments made in section 3.1.4, then in the context of concurrent synthesis, the *Current Working Knowledge (CWK)* concerns knowledge being generated of both the *artefact* and the *life-phase systems* solutions. This *concurrent synthesis CWK* (Figure 7.1) would for example include descriptions of life-phase system composition. Hence, based on the phenomena of LCC_{ni} (section 6.3.1), implicitly co-evolving with this concurrent synthesis CWK is knowledge of solution specific LCC_{ni} (Figure 7.1). Similarly, based on the phenomena of LCC_i (section 6.3.2), implicitly co-evolving is solution specific LCC_i knowledge.

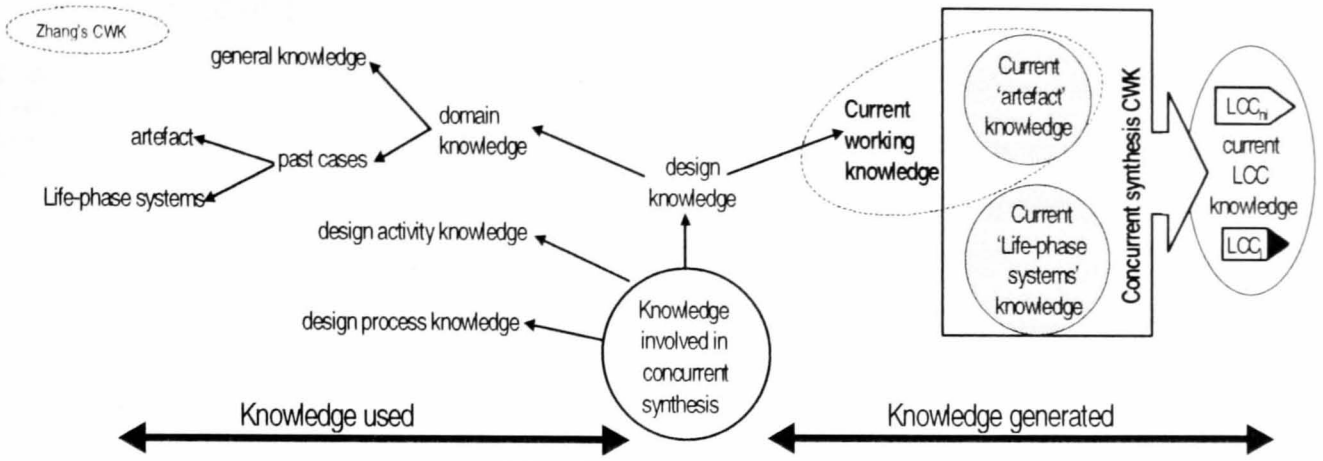


Figure 7.1 - Implicit LCC knowledge co-evolution - modified from [Zhang 1998]

LCC knowledge

The term *knowledge* has been used in this research without an explicit definition. The difficulty of defining it stems from the fact that knowledge exists in many ways [Reich 1992], resulting in many meanings [Giarratano et al. 1994]. It is not the scope of this thesis to investigate the philosophical aspects of *knowledge* by reference to *epistemology* – the study of philosophical problems surrounding knowledge. However, for the purpose of the approach being proposed in this chapter, the term *LCC knowledge* means information in any kind that makes explicit, solution related, unintended or intended LCCs. Based on discussions in section 3.1.4, as knowledge processed in design can vary in *type, source* and *characteristics,* Table 7.1 discloses from an artefact domain independent point of view, the nature of LCC knowledge.

Table 7.1 – Nature of LCC knowledge

Issue	**Comment**
Type	• implicit within concurrent synthesis *CWK*;
Source	• *private & public* (see Figure 3.19) • *distributed* internally & externally to an organization (Figure 3.19);
Characteristics	• *wide breadth dimension* – distributed amongst artefact life actors • *subject to uncertainty* - there is no certainty that commitments made will definitely result in certain LCCs during the artefact life. Rather, they are assumed to be true, under normal conditions. • *varies in permanence* - an example of permanent LCC knowledge is that fasteners require a hole to be generated in the realization phase; an example of static LCC knowledge is the production quantities above which automation is feasible; a piece of dynamic LCC knowledge is that of LCC arising from a material utilized in different temperature environments.

Role of Timely LCC Knowledge Utilization

LCC knowledge can influence mental processing

It is human designers that are involved in making synthesis decision commitments. Much of the design process is a *mental process* [Finger et al. 1989]. One characteristic of human designers is that they have a mind, in which mental processes taking place, can be more or less *influenced* [Roozenburg et al. 1995]. As discussed in section 3.1.5, knowledge is a factor that has a key influence on design, as it can guide design reasoning [Mohd-Hashim et al. 1994]. Further, as disclosed in section 6.1, during decision making, alternatives are interpreted by the decision maker in terms of their expected consequences. Hence, knowledge of the consequences associated with different alternatives plays an important role during the selection of alternatives. However, as the next section reflects, a number of human mental processing limitations influence the effective utilization of LCC knowledge at the *right time* during solution synthesis decision-making. As a result, in reality, not all the consequences associated with an alternative are known *during* decision making, this resulting in *unintended* LCCs.

7.2 Difficulties In Utilizing LCC Knowledge

To appreciate the difficulties designers have in utilizing LCC knowledge, section 7.2.1 provides an explanation of human mental processing operations this leading to the disclosure of designer LCC knowledge utilization difficulties in section 7.2.2.

7.2.1 Human Mental Processing Limitations

Human memory processing fundamentals

In the case of the human brain, so little is known that the researchers ideas about *what might* be going on are based on models [Klatzky 1980; Ellis et al. 1989; Bjork et al. 1996]. As illustrated by the model in Figure 7.2 adopted from [Ellis et al. 1989], human memory processing fundamentals can be explained from an *information processing* perspective. This model is divided into three processing components: *input processing*, *storage* and *output* described below:

Input processing

Input via receptor cells

Information from the external real world is received by *receptor cells* - these are the *eye*, *ear*, *nose*, *tongue* and *skin*. For example, during synthesis, the visual input would include information represented on say a paper or computer screen of the evolving artefact life solution.

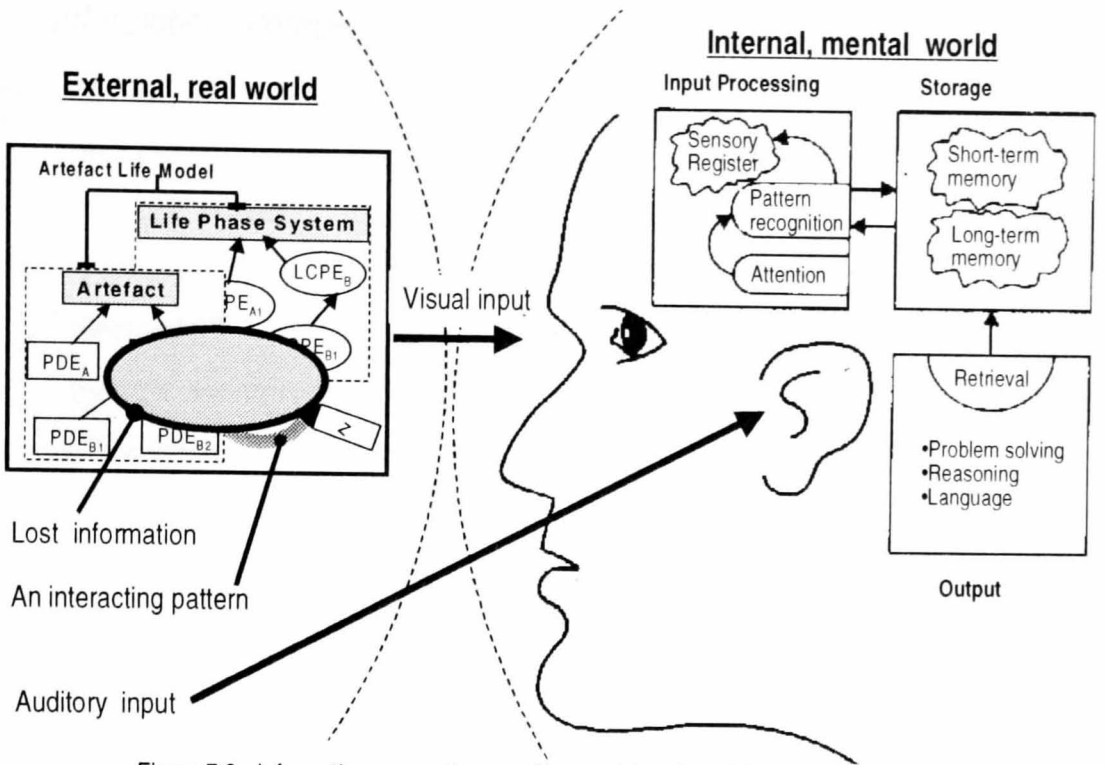


Figure 7.2 - Information processing cognitive model - adopted from [Ellis et al. 1989]

Meaningless information in sensory register

The information received by *receptor cell* is stored in the *sensory register.* This information, at this stage having no meaning, remains on the sensory register only for a *brief time* as information in the sensory register *decays very rapidly.* Thus, much of what is available, as input will not reach meaningful processing. Hence, at any one time, a portion of the information available in the real world, external to the human brain, will be lost (Figure 7.2).

Pattern recognition for deriving meaning

The meaningless information in the sensory register is then given *meaning* through a *pattern recognition (PR)* process. Deriving the meaning of sensory information involves matching that information with a representation in long-term memory. Human beings also have difficulties in organizing, summarizing and using information to form inferences about the causal connections of events and about relevant features of the world [March 1994]. Thus although they often receive relevant information, they may fail to see its relevance.

Attention on a 'portion' of the information

Since the time and capabilities for attention of the human being are limited, not all information reaching the sensory register can be processed in PR. *Attention* is the process which determines which information will be *selected* to be processed in situations where not all the available information can be processed. Thus, the decision to select particular information for pattern recognition is critical. In the context of this research, this means that the designer's attention needs to be rapidly attracted to causal and interacting patterns reflecting LCCs not being attended to by designers, if they are to cater for a host of total life issues during synthesis.

Information Storage

Storage capacity & retention time

At any given time, immediate experiences/events occupy our thought processes but at the same time, many other facts are also known but not thought about. This distinguishes between *working memory* and *inactive memory.* In this cognition model, working memory is ascribed to *short-term memory (STM)* and inactive memory to *long-term memory (LTM).* The human being's short-term memory capacity generally ranges from five to nine items [Ellis et al. 1989; Kerr 1993; Pahl et al. 1996]. Similar to a computer's random access memory (RAM), information in STM is retained for a very short time interval. On the other hand, like a computer's hard disk, LTM allows the *permanent retention* of knowledge acquired formally and through experience by the individual over a long period [Ellis et al. 1989; Boston et al. 1996].

Output

Utilization of retrieved meaningful information

Once information is accessed in memory, it may be used in a variety of ways such as for problem solving, concept generation, reasoning and language processing (Figure 7.2). For instance, in forming the concept of 'dog', young children learn to classify a variety of specific instances as members of a set. They learn that the label 'dog' may be applied to specific instances but more importantly they learn that 'dog' represents a class of instances which have certain properties in common. When used for reasoning, information retrieved is used to derive a conclusion from a given set of facts.

7.2.2 Designer Difficulties

Missing LCC knowledge

Due to the profound distribution of LCC knowledge, it is difficult for such knowledge to be possessed by designers. At the same time, as argued in section 4.2.1, for most of the time, design team members work in isolation. Hence, little if any, such LCC knowledge is therefore acquired and stored in the designer's *long term memory* for supporting them to foresee during synthesis, unintended LCCs co-evolving with their decision commitments.

Difficulty in recognizing LCC patterns

As information in the sensory register decays, a portion of the information input is lost. At the same time, the working memory (STM) is limited to handling an average of 7 items at any one time. Thus, even if designers possess LCC knowledge, these limitations hinder their ability to detect within their mental world, the many *causal* and *interacting* patterns between the commitments made to the artefact life model in the real world (Figure 7.2).

Difficult to achieve timely attention

As argued in section 3.3, to be effective, $D_sF\Sigma X$ requires artefact life exploration to be rapid. The human being's mental processing speed is known to be slower than a computer's [Sharples et al. 1989]. With missing LCC knowledge and a difficulty in revealing LCC patterns, the designer's attention is not rapidly attracted to unintended LCCs co-evolving with the solution, at the *right time*, during synthesis decision making.

Scope of amplifying designer's LCC knowledge processing capability

These designer LCC knowledge processing limitations highlight the scope of:

- capturing distributed LCC knowledge and storing it for future reuse;
- supporting designers in utilizing the stored LCC knowledge to attract their attention at the *right time* to patterns of unintended LCCs co-evolving with their solution description.

7.3 'Knowledge of LCCs' Approach Framework

Approach concept

To support designers, human mental processing limitations have to be taken into account in engineering design so that procedures and methods recommended match the way humans think [Pahl et al. 1996]. This section presents the *'Knowledge of life-cycle Consequences (KC)'* approach framework to $D_sF\Sigma X$ that takes into consideration the difficulties designers have in handling co-evolving *current LCC knowledge* (Figure 7.1).

Considering the right things at the right time

By exploiting the explanations provided by the phenomena model, the concept of the 'KC' approach is to explicitly reveal the *current LCCs* co-evolving with the concurrent synthesis CWK, so that the designers foresee unintended multiple artefact life issues *during* synthesis. The argument is that for solution specific $D_sF\Sigma X$, the synthesis decision making process will be better accomplished by reasoning with *current LCC* knowledge because the designers' attention can be *timely* attracted to consider and explore unintended LCCs. By revealing solution specific problematic LCCs, designers can be motivated to explore alternative commitments, thereby being guided in the generation of life-oriented design solutions.

Approach framework

In order to adopt this approach, there is a need to describe *what* needs to be captured and modelled to generate a suitable LCC knowledge base. Also, to timely support the designer's thinking process, designers need principles disclosing how to describe evolving artefact life solutions from which current LCCs can be inferred. For this purpose, the *'KC'* approach to $D_sF\Sigma X$ is based

on three frames (Figure 7.3), an *Artefact life modelling frame*, a *Knowledge Modelling frame* and an *Operational frame*, that collectively form the *'KC' approach framework*.

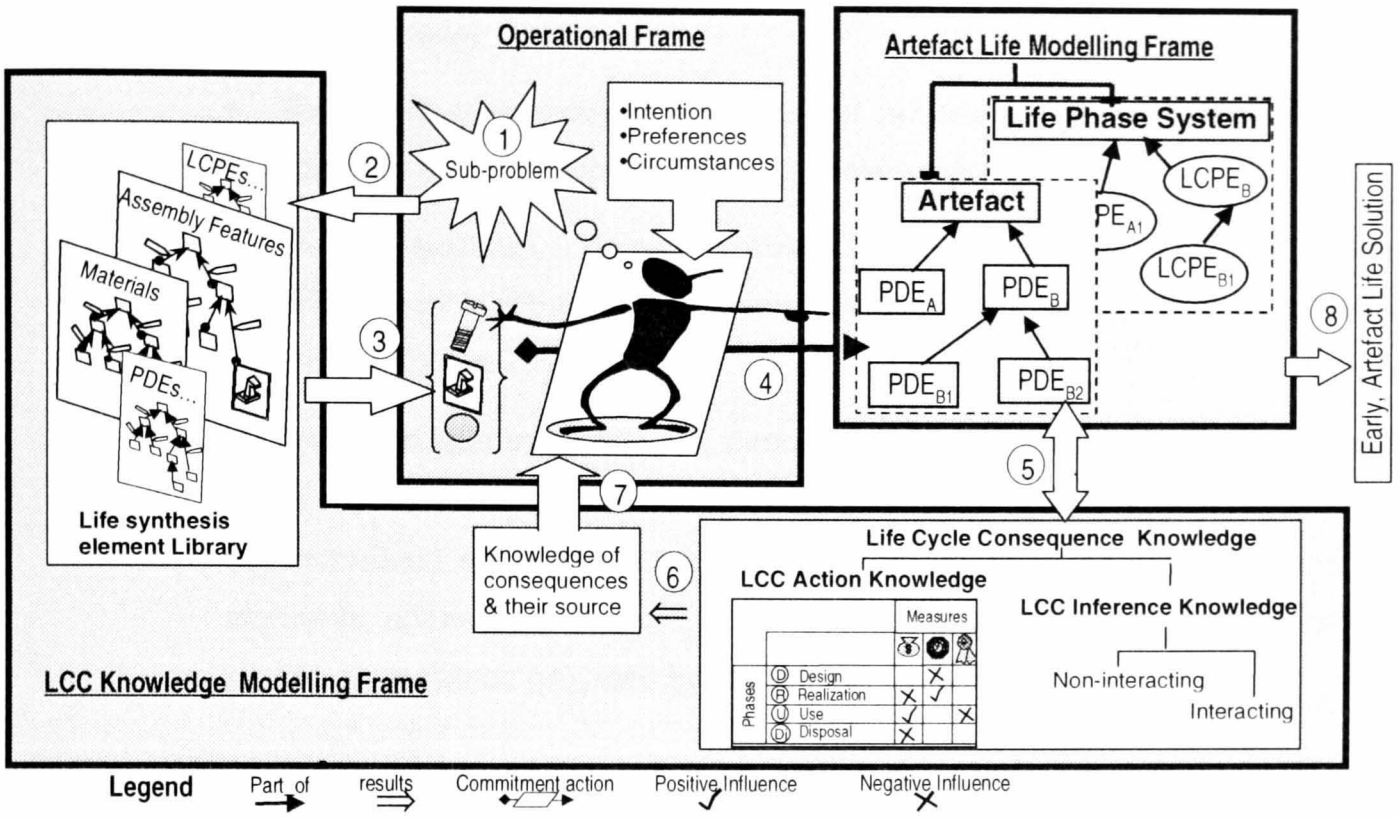


Figure 7.3 - The 'KC' Approach Framework

7.3.1 Operational Frame

By exploiting the LCC phenomena model explanations, the *Operational Frame* provides :

i. a model of the *$D_SF\Sigma X$ working environment* and

ii. a description of the designer's *$D_SF\Sigma X$ mode of operation* so that designers can systematically utilize modelled LCC to infer, at the right time, co-evolving LCCs.

$D_SF\Sigma X$ working environment

The *$D_SF\Sigma X$ working environment* is composed of:

- the human designer, who is engaged in synthesis decision making;
- a set of *reusable* domain specific, well developed solution elements (PDEs) and well known life cycle phase elements (LCPEs);
- an LCC knowledge model for the domain specific elements in the S.E.L;
- a means by which the designer can search for synthesis elements resulting in an intended consequence;
- a means by which a designer can interact with and evolve an artefact life solution model;
- a communication medium providing designers with:
  - awareness of LCCs co-evolving with their solution description;
  - guidance to the avoidance/relaxation of detected LCCs.

$D_sF\Sigma X$ mode of operation

The *$D_sF\Sigma X$ mode of operation* is that:

- *synthesis decision making* takes place by designers as in the *model* of Figure 6.8; commitments can be made in *any order* and at a *least* or *specific* level;
- the designer is engaged in concurrent synthesis - the *artefact model* and the *life-phase models* are *what* the designer generates;
- during synthesis, designers interact with a synthesis elements library (PDEs and LCPEs), the latter formally described through LCC knowledge model (Figure 7.3).

As described later in section 7.4, through this *$D_sF\Sigma X$ working environment* and *mode of operation*, the designer can search for synthesis elements, commit/retract elements to evolve/modify the artefact life model, specialize elements, utilize LCC knowledge and explore alternative commitments based on design guidance provided by the knowledge model.

7.3.2 LCC Knowledge Modelling Frame

What versus how?

As outlined in section 5.2, a problem in modelling LCC knowledge for use by designers is concerned with *what* to capture and model, rather than *how* to represent knowledge. The knowledge modelling frame (Figure 7.3) is like an instrument to the 'KC' approach as it discloses *what* to *acquire*, *model* and *relate* for an application domain, together with *how* to transform and structure the established relationships into meaningful LCC knowledge to support the inference of both LCC_{ni} and LCC_i relevant to the application domain.

Distinction between LCC inference and LCC action knowledge

The phenomena model presented in Chapter 6, reveals that *concretization* or *detailing* synthesis decision commitments generate LCCs (e.g. a sink mark defect) that can in turn influence the behaviour of multiple life-cycle phases with respect to performance measures e.g. cost, time and quality (Figure 7.4). Modelling LCC knowledge is therefore concerned with the description of *LCC inference knowledge* (Figure 7.4), this concerning *causal relationships* between the evolving *artefact life model* and the resulting *LCCs.* It also concerns modelling *LCC action knowledge.* This knowledge describes actions that arise from the generated LCCs such as fluctuations in life-phase performance measures. Thus, this frame (Figure 7.3), detailed in Chapter 8 for the mechanical component domain, provides a *formalism* for:

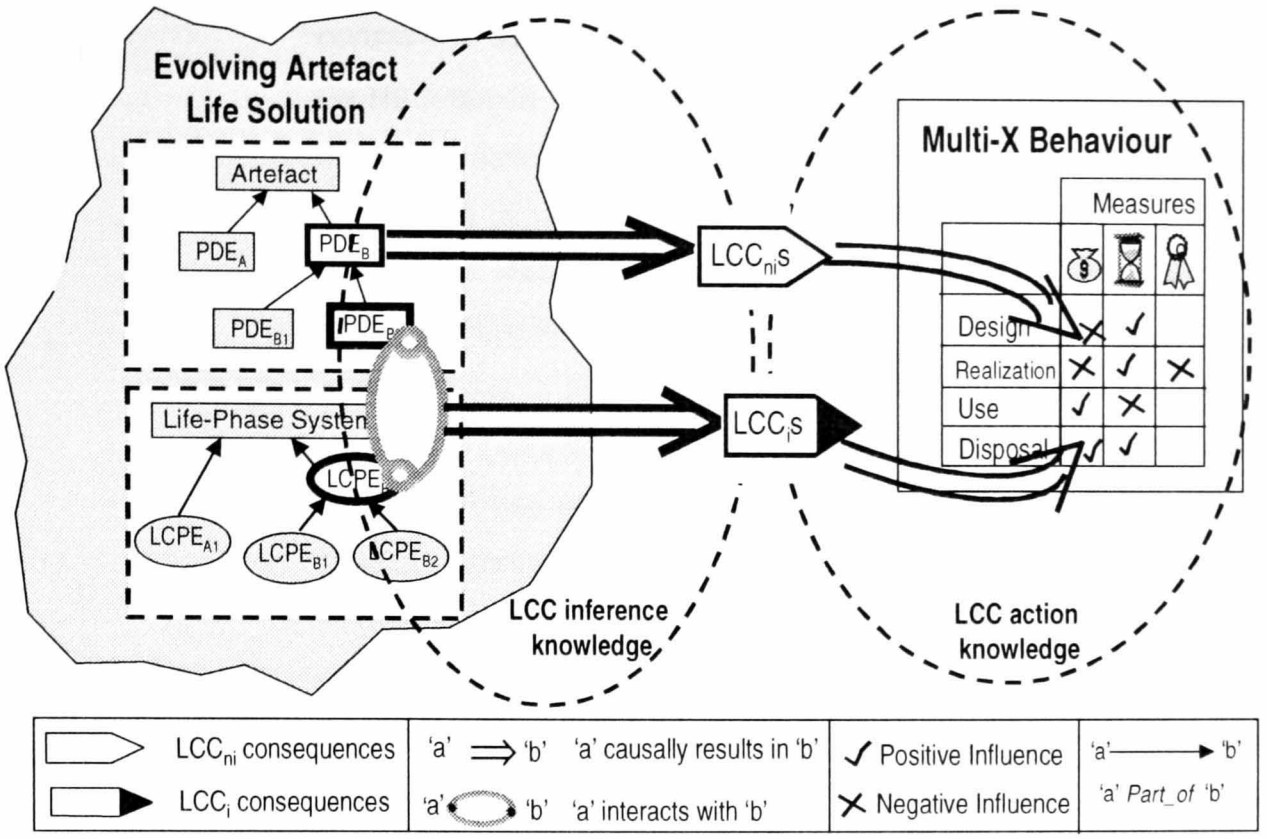


Figure 7.4 - Relationships embodying LCC knowledge

Reuse library

i. ***a life synthesis element library:*** this consists of a structured library of various models of synthesis elements (PDEs and LCPEs) frequently reused within the design and life of an artefact's domain. These elements are used to generate the artefact life model. In the case of components, these include models of form features, assembly features, materials and LCPEs such as fabrication systems and assembly systems. Elements in the library can be generic (e.g. a hole form feature) or domain specific (e.g. mould core-pin). This library amplifies the designer's knowledge base with synthesis elements that can be reused to generate domain artefact life solutions.

ii. ***relationships*** between PDEs/LCPEs and LCCs describing how to model:

(a) *LCC inference knowledge*: these are descriptions logically relating artefact domain synthesis elements to LCC_{ni} and LCC_i (Figure 7.3). Based on the phenomena model, in the case of LCC_i, interacting relationships can be between different PDEs, different LCPEs or between PDEs and LCPEs.

(b) *LCC action knowledge* this describing:

- *mappings between LCCs and performance measures* - a LCC (e.g. sink mark) can cause a change (δ) to a *performance measure (PM)* e.g. cost, of a technical process [P], forming part of that phase (e.g. time of moulding);

- *concurrent synthesis patterns:* in some cases, the LCC is a decision space 'S' with a set of decision proposals D{O} having a set of options {O}. Sometimes, the option set {O} has associated default commitments. Describing such situations provides a pattern allowing commitments to be automatically made to evolve a specific <model>;
- *explanations of specific LCC and providing guidance to their* avoidance/relaxation: having inferred a LCC (e.g. a sink mark), designers need explanation of what the LCC means and what it effects together with guidance to its avoidance/relaxation. Through the LCC phenomena model, knowledge of *which* commitment(s) give rise to a detected LCC can be made explicit, thus guiding designers to a list of commitments that can be explored.

7.3.3 Artefact Life Modelling Frame

Viewpoint dependent solution modelling

As discussed in section 2.1, the synthesis of mechanical artefacts can take place from different viewpoints (e.g. functional or constructional) and for different system levels (e.g. component or sub-assembly). Irrespective of the synthesis viewpoint or system level, as argued in section 3.2.1, exploring alternatives generates LCC knowledge that is useful for reasoning taking place during the selection of elements. Thus, depending on the artefact system level and the synthesis viewpoint $D_sF\Sigma X$ is concerned with, designers need relevant life synthesis elements (e.g. functional elements versus constructional elements). In this sense, the *Artefact Life Modelling Frame* is like an instrument for the framework. It provides for a specific synthesis viewpoint, a formalism for describing an evolving *'artefact life' model* that allows relevant LCCs to be inferred.

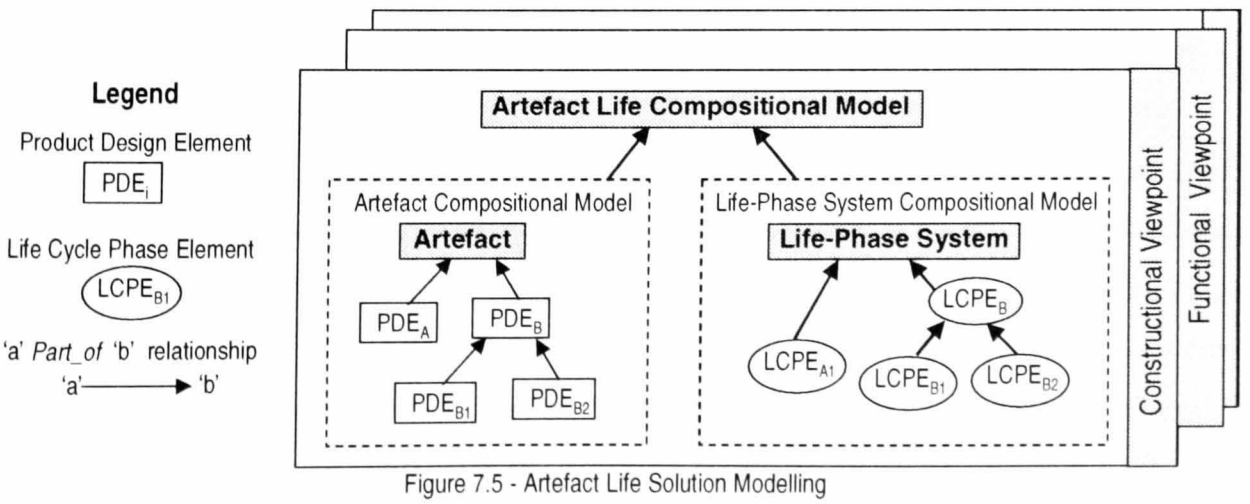


Figure 7.5 - Artefact Life Solution Modelling

Synthesis elements & their relationships

The models described by the artefact life-modelling frame are the *artefact model* and *life-phase system models* (Figure 7.5). For a constructional viewpoint, the elements describing the artefact life solution are PDEs (e.g. a hole form feature) and LCPEs (e.g. a mould core-pin). From a constructional viewpoint perspective, these elements are linked with *part_of* (→) relationships to form an evolving conceptual artefact model and evolving conceptual life-phase system models. These collectively form the evolving conceptual *artefact life compositional model* (Figure 7.5).

7.4 $D_sF\Sigma X$ Through The 'KC' Approach Framework

By realizing the 3 frames presented making up the 'KC' approach framework, $D_sF\Sigma X$ can take place as described in this section.

Search for synthesis elements

As illustrated in Figure 7.3, when encountering a sub-problem (step 1), the designer interacts (step 2) with a *life synthesis elements library* to search for a feasible set of elements {O}. For example, designers can search for assembly elements that result in non-permanent bonds.

Decision proposal concerning alternative elements

Due to the set of *alternative* synthesis elements retrieved, the designer reaches a decision proposal point D{O} (step 3). As the *synthesis decision commitment model* (Figure 6.8) explains, by taking into consideration *intentions*, *preferences* and *circumstances* known at that time, the designer commits (step 4) the chosen element to evolve the artefact life model.

LCC inference

Through monitoring (step 5) taking place with captured LCC knowledge, relevant LCC_{ni} and LCC_i are inferred (step 6) *independent* of the designer's cognitive capacity.

Pro-active exploration

The inferred LCC knowledge is explicitly provided to the designer (step 7). This makes it possible for the designer's *attention* to be attracted during synthesis decision making, to LCCs co-evolving with their solution. This inferred LCC knowledge gives rise to a new state of *circumstances* that are *now* known by the designer. As LCC knowledge is based on the explanations provided by the phenomena model, designers can be also provided with knowledge of *which* commitments give rise to the inferred LCCs. The aim of providing this knowledge is to *rapidly* and *pro-actively guide* the designer's reasoning to *which* synthesis commitments could be explored in order to avoid or relax the detected LCCs.

Designer controlled process

As reflected by the *operational frame* (Figure 7.3), the approach is designer controlled. It is the designer who is responsible for making synthesis commitments; for judging if the LCCs revealed are acceptable or not for the given situation; for deciding whether to explore or not; for deciding whether to accept the recommendations being provided by the knowledge model or not; for deciding when to terminate (step 8) the synthesis and exploration activities. In this sense, the LCC knowledge model provides *design assistance* by 'amplification' of the designer's mental processing capabability without hindering the designer's control of the design process.

7.5 KICAD Based Framework Realization

Need for computational approach realization

Adopting the established *'KC'* approach to $D_sF\Sigma X$ is however impeded due to designers' *memory storage capacity* and *knowledge processing* limitations discussed in section 7.2. Then, based on arguments made in section 4.4, a computer based means offers a better avenue to providing direct support to life-oriented design from the perspective of providence. As discussed in Chapter 4, a strength of *knowledge based systems (KBS)* is their ability to retain and rapdily process captured, traditionally distributed knowledge, so that it can be explicitly reused in new design situations. Also, due to their inference engine, KBSs have a predictive power that makes them suitable as a means to infer LCCs. A sensible avenue is to therefore realize the 'KC' approach framework in computational form as a *Knowledge Intensive CAD (KICAD) tool* [Tomiyama et al. 1995; Mantyla et al. 1996]. A KICAD tool does not only support designers in building up solution models but also exploits KBS technology to provide a means of how captured knowledge can be retained and processed at fast processing speeds, to be reused in new design situations at the right time.

7.6 Chapter Conclusions

'KC' Approach framework

By exploiting the LCC phenomena model explanations and disclosing designer mental processing limitations, this Chapter proposed the concept of a 'KC' approach to $D_sF\Sigma X$ in order to *timely* attract the designers *attention* to unintended *current LCCs* co-evolving with synthesis decision commitments. As argued, making this LCC knowledge explicit during synthesis can motivate and guide designers to explore alternative synthesis commitments so as to generate life-oriented design solutions. To develop this 'KC' approach, this

chapter presented three frames, an *Artefact life modelling frame*, a *LCC Knowledge modelling frame* and an *Operational frame*. The resulting 'KC' approach framework collectively describes how to model LCC knowledge and the principles of how to *timely* reuse the captured knowledge to proactively guide designers in $D_sF\Sigma X$.

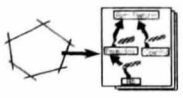
KICAD based framework

Due to the designers' mental processing limitations, it has been argued that a sensible avenue is to realize the 'KC' approach framework as a *KICAD tool*. To develop such a KICAD tool, an application oriented LCC knowledge model is required. In such a model, the structure and the behaviour of the world for an artefact domain, is stored as a collection of concepts, their interconnections, and their mutual causality. This chapter did not specifically present *what* concepts should form part of such a LCC knowledge model for mechanical component $D_sF\Sigma X$. In addition, no formal explanation has been provided as to how LCC knowledge can be structured to infer LCC knowledge *specifically relevant* to the evolving artefact life solution at the *right time* during solution synthesis. These LCC knowledge modelling issues will be specifically addressed in the next chapter.

chapter 8

8.0 Formalism of A LCC Knowledge Model

Scope



To codify LCC knowledge in a KICAD tool, an application domain LCC knowledge model is required. The scope of this chapter is to present a formalism of a LCC knowledge model by disclosing what synthesis elements and generic relationships describe LCC knowledge. For this purpose, section 8.1 first provides a background to knowledge modelling. Section 8.2 then presents the established generic LCC knowledge modelling formalism. Given the specified research boundary, section 8.3 discloses a number of relationships established for the mechanical component domain. To enable relevant LCC knowledge modelled to be reused at the *right time* during $D_sF\Sigma X$, section 8.4 discloses LCC knowledge structuring concepts, this as discussed also suitable for addressing knowledge management issues. Chapter conclusions are made in section 8.5.

8.1 Knowledge Modelling Background

Recognized and computable knowledge

Knowledge can be *recognized* and *unrecognized* [Tomiyama et al. 1995] (Figure 8.1). Further, knowledge can be *codified* in a form that is described in any human-recognizable form, this ranging from being non-computable (e.g. textbook knowledge) to computable (e.g. a database). A pre-requisite to codifying LCC knowledge in a KICAD tool is that it is therefore made *recognizable and computable*. This chapter exploits the explanations provided by the phenomena model (Chapter 6) to advance a further step in this codification process by generating a *LCC knowledge model* that can be then codified into computable form (Figure 8.1).

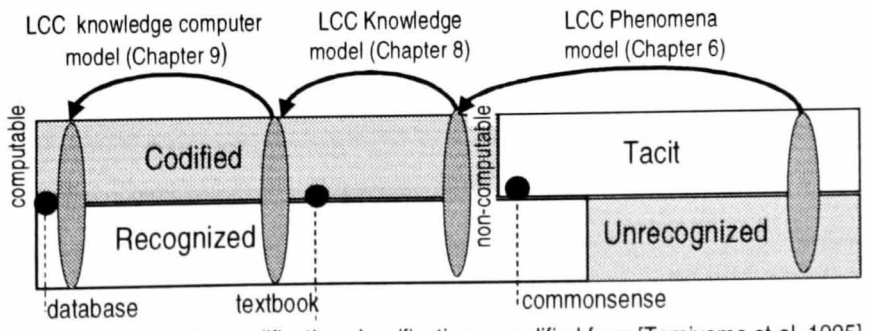


Figure 8.1 - Knowledge codification classification - modified from [Tomiyama et al. 1995]

Modelling background

To discuss LCC knowledge modelling, it is necessary at this stage to provide some generic background to *modelling*, derived from [Schut et al. 1996]. Models refer to something i.e. a *referent.* For example a map refers to some area in the world. Models adopt a *perspective* to match their purpose. For example a map does not refer to biological processes in an area but to topographical issues. The *purpose* of a model thus determines which aspects are relevant for modelling purposes and which can be neglected [Xia et al. 1996]. Certain aspects are *input* to a model whilst others are *output*. Models are therefore useful to *infer* information. For instance, maps may be used to infer distances between cities. However, due to the perspective adopted a road map cannot be used to infer how many trees are in a particular street.

LCC knowledge modelling

In the case of LCC knowledge modelling, the referent is the phenomena of LCCs. The purpose of a LCC knowledge model is to support knowledge engineers in capturing and codifying relevant knowledge for inferring LCCs from the model's input – an evolving artefact life model. For the purpose of this research, the knowledge model perspective adopted is one focusing on supporting mechanical artefact design at the component level. Therefore, the aspects considered relevant are *component life-synthesis elements* and *relationships* between them that describe *component LCC inference knowledge* and *LCC action knowledge.* It neglects focusing on detailing the internal structure of component life-synthesis elements.

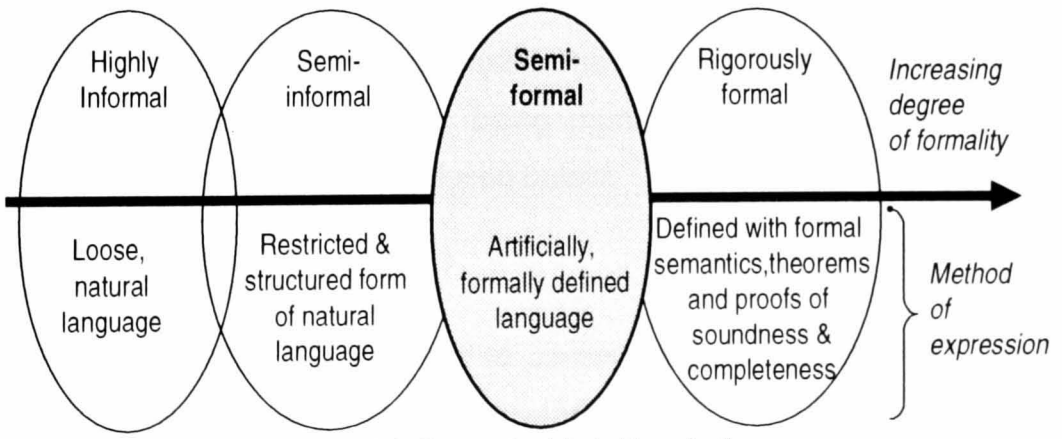


Figure 8.2 - Degree of ontological formalization

Formalism motivation

To *promote a shared understanding* of a LCC knowledge model, to the design research community made up of people with different backgrounds, viewpoints and different research thrusts, some form of *ontology* is required. An ontology [Uschold et al. 1996] embodies some sort of world view with respect to a given domain *(e.g. mechanical components).* The world view is often conceived as a set of *concepts* (e.g. PDEs), their *definitions* and their *inter-relationships.* An ontology thus includes a *vocabulary* of terms and some specification of their meaning. The degree of formality by which the vocabulary is created and

meaning specified varies considerably [Uschold et al. 1996] (Figure 8.2). As LCC knowledge is very domain specific and covers multiple artefact life-phases, no claim is being made that the model presented in this dissertation is rigorously formal. Rather, the model is presented in a *semi-formal* way using the notation in Appendix C. Nevertheless, it provides a basis by which LCC knowledge modelling can be explained.

8.2 LCC Knowledge Modelling Concept

From material presented in section 7.3.2 concerning the LCC knowledge modelling frame, the required model concerns formalizing *relationships* between elements in the *life synthesis elements library* and LCCs. Section 8.2.1 therefore presents generic relationships describing LCC knowledge whilst section 8.2.2 presents a formalism related to mechanical component life synthesis elements

8.2.1 Generic Relationships Describing LCC Knowledge

This section presents generic relationships between different PDE, LCPEs and LCCs describing *LCC inference knowledge* in section 8.2.1.1 whilst section 8.2.1.2 presents descriptions of *LCC action knowledge.*

8.2.1.1 LCC Inference Knowledge

Non-interacting Consequence Knowledge

Based on the LCC phenomena model, the basic *causal relationship* between synthesis decision commitments being made to an artefact life model and LCC_{ni}s is given from (6.1), reproduced below:

(6.1) $([d]_E\{O_1\} = o_1^{\,1}) \Rightarrow (LCC_{ni} = \text{'x'} \quad)$

Generic LCC_{ni} knowledge model

Causal relationship (6.1) reflects that consequence *'x'* can be associated with the *committed option* $o_1^{\,1}$. That is, knowledge of LCC_{ni}s can be modelled as:

(8.1) $([d]_E\{O\}) \Leftarrow \{ LCC_{ni} \}$

read as:

A synthesis commitment $[d]_E\{O\}$ has ($\Leftarrow$) an associated set $\{LCC_{ni}\}$ of non-interacting life-cycle consequences.

The synthesis commitment can be a ⌊class⌋, a specific [PDE] or [LCPE] if it is a concretization commitment, or a parameter value if it is a detailing commitment (section 6.2). Set $\{LCC_{ni}\}$ can have one or more members of LCC, the latter defined in section 2.3.5.1.

Interacting Consequence Knowledge

The basic causal relationships between synthesis decision commitments and interacting LCC_i is given by the phenomena relationship (6.5) reproduced below:

(6.5)

$$([d]_E\{O_1\}= o_1{}^1) \wedge ([d]_E\{O_2\} = o_2{}^2) => (LCC_i = \text{'z'})$$

Interacting Relationships

Causal relationship (6.5) depicts that it is the *interacting relationship* between elements $o_1{}^1$ and $o_2{}^2$ (i.e. the synthesis decision commitments) which *causes* the interacting LCC 'z'. An interacting relationship $(IR)_j$ between a set of synthesis elements is a *consequence-specific* conjunctive set of $[d]_E\{O\}_j$ *($j \geq 2$)* synthesis commitments formally:

(8.2)

$$(IR)_j = \{ [d]_E \{O\}_1 \wedge [d]_E \{O\}_2 \wedge \ldots [d]_E \{O\}_j \}$$

An $(IR)_j$ can exist between commitments made to the same model (e.g. component) or between commitments made to the different models forming part of the artefact life model.

Generic LCC_i knowledge model

Then from the above, LCC_i knowledge can be modelled by causally relating a LCC_i to an *interacting relationship $(IR)_j$*, formally:

(8.3)

$$(IR)_j \Leftarrow \{ (LCC_j)_i \}$$

read as

An interacting consequence $(LCC_j)_i$ is associated with an interacting relationship $(IR)_j$ describing a LCC-specific conjunctive set of synthesis commitments.

LCC_i knowledge modelling example

A specific example is the piece of knowledge concerning the formation of *sink mark defects* in thermoplastic components:

(8.4)

$$\{([d]_E\{[F]\}=[\text{rib}]) \wedge ([d]_E\{[M]\}=[\text{ABS}]) \wedge (([d]_E\{[P]\}=[\text{moulding}]) \rightarrow <\text{phase}>_{\text{realization}})\} \Leftarrow \{ (LCC_i = \text{'sink mark defect'} \rightarrow <\text{model}>_{\text{component}}) \}.$$

8.2.1.2 LCC Action Knowledge

As disclosed through the LCC knowledge modelling frame (section 7.3.2), LCC action knowledge concerns *performance measure mapping*, *concurrent synthesis patterns* and *LCC explanations/guidance*.

Performance Measure Mapping

An LCC can cause a change (δ) to a performance measure (PM) of a technical process [P], formally:

(8.5) $LCC_{ni} => \delta(PM)[P]$ or $LCC_{i}=> \delta(PM)[P]$

Relation (8.5) allows a change (δ) caused in the respective performance measure to be causally related to the respective LCC. Using *relative values* for changes in performance measures on a scale –10 to +10, the *performance measure mapping* for the LCC_{ni}= 'assembly slow' of a screw commitment (6.7a) can for instance be modelled using (8.5) as follows:

(8.6) $(LCC_{ni}='assembly\ slow')=>(Quality=-3)([P]=[Assembly]) \wedge (Time=+6)([P]=[assembly]) \wedge (Cost=+5)([P]=[Assembly])$

Performance propagation

Changes in performance measures propagate to life-cycle phases involving that [P]. For instance, 'assembly slow' will influence any phase involving an assembly process. This propagation can be explained through the following example. Assume that the following current knowledge on the artefact life composition is modelled:

(8.7) $([P] = [Assembly]) \rightarrow <Phase>_{Realization}$

(8.8) $(([P] = [Dis\text{-}Assembly] \wedge ([P] = [Assembly])) \rightarrow ([P]=[Service])$

(8.9) $([P]= [Service]) \rightarrow <Phase>_{Use}$

Propagation sensitivity

Therefore the fluctuations in the performance measures for LCC_{ni}= 'assembly slow' given in (8.6), will through (8.7), (8.8) and (8.9) be propagated onto $<Phase>_{Realization}$ and $<Phase>_{Use}$ since ([P] = [Assembly]) is *part_of* ($\rightarrow$) these phases. The sensitivity of this propagation mechanism therefore depends on the resolution of the respective life-phase compositional model. An assembly process can for instance be decomposed into sub-processes such as *gripping bonding, orienting* and *inserting* (Willemse and Storm 1995). Therefore, if say a snap-fit influences the *bonding time,* then through such a detailed assembly process compositional model, the overall assembly process is influenced. Thus from 8.7, 8.8 and 8.9, this propagates onto the realization and the use phases.

Virtual artefact life behaviour modelling

Due to performance propagation, a LCC can cause fluctuations in *multiple* life-phases. Mapping these fluctuations provides a means of estimating the relative *artefact life behaviour* resulting with a synthesis decision commitment. As discussed in section 2.2.3, the transformation in a $<Phase>_i$ is the result of a set process $\{[P]_j\}$ effects delivered by a number of systems forming *part of* that phase. Then, assuming technical processes [P] are executed sequentially within a life-phase, an *estimate* of a specific performance measure $(PM)_x$ of that

phase is given by the summation (Σ) of the current value of that performance measure, for the different processes $[P]_j$ involved in that life-phase, formally:

(8.10) Since ($\{[P]_j\} \rightarrow$ <Phase>$_i$) then $(PM)_x$<Phase>$_i = \Sigma\ (PM)_x[P]_j$

Mapping behaviour as a DFΣX Matrix

Relationships (8.5) and (8.10) collectively allow *relative* performance measures for different life-phases to be estimated to generate a DFΣX *matrix* (see section 2.3.5.2). This matrix provides a means by which a designer can foresee and *monitor* the virtual behaviour of the *artefact's life* due to co-evolving LCCs and their propagations across *multiple* life-phases.

Concurrent Synthesis Patterns

As discussed in section 7.3.2, a concurrent synthesis pattern describes situations where a LCC gives rise to some decision proposal D {O} concerning elements that need to be committed to some <model>$_k$ which however has a default commitment ($[d]_E\{O\} = o'$). This pattern is formally described as:

(8.11) If (LCC => D {O}) and D {O} has a default commitment o',

then ($[d]_E\{O\} = o'$)→ <model>$_k$.

For example, Figure 8.3a schematically describes a *concurrent synthesis pattern* arising from the screw commitment example in (6.7) :

(8.12) (LCC_{ni}='hole required') => ($[d]_E\{[F]\}$= [hole])→ <model>$_{component}$

Further, due to the new state arising from 8.12, the LCC_i = 'core-pin required' in (6.9) gives rise to the following concurrent synthesis pattern (Figure 8.3b):

(8.13) (LCC_i='core-pin required') => ($[d]_E\{[LCPE]\}$= [core-pin])→<model>$_{mould\ tool}$

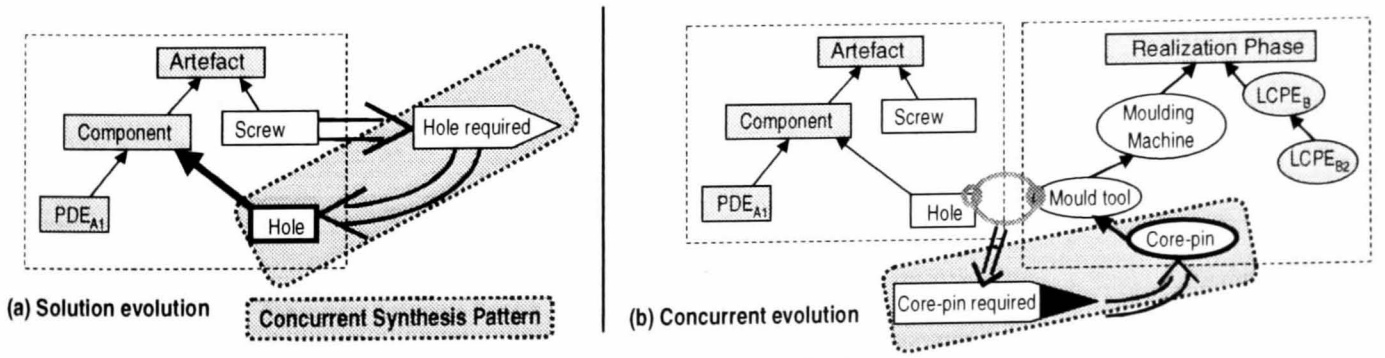


Figure 8.3 - A typical concurrent synthesis pattern

Significance of modelling concurrent synthesis patterns

By explicitly utilizing modelled concurrent synthesis patterns such as (8.12) and (8.13), *new* commitments automatically made to the artefact life model due to these patterns can therefore propagate their own LCC_{ni} and LCC_i across *multiple* life-phases.

Guidance/Explanation Knowledge

As discussed in section 7.3.2, a type of LCC action knowledge is the provision of designer *guidance* and *explanations* of detected LCCs.

LCC guidance knowledge

LCC guidance knowledge describes:

(a) which commitments, from possibly many others already made to the evolving artefact life model, should be explicitly explored to avoid a detected LCC. This guidance knowledge is derived from (8.1), (8.2) and (8.3). For instance, the source commitments resulting in the sink mark defect example in (8.4), are the [rib] form feature, the [ABS] material and the injection [moulding] process.

(b) specific action that can be taken beyond the design phase (e.g. changing the injection moulding pressure in the realization phase) to relax a consequence (e.g. sink mark) if the source commitments cannot be avoided.

LCC explanation knowledge

LCC explanation knowledge is highly informal. It is a description of a detected consequence e.g. describes what a sink mark is, how it looks, why it is considered problematic etc.

8.2.2 Mechanical Component Synthesis Element Formalism

Detailing the internal structure of synthesis elements is not within the scope of this research – this can be found elsewhere e.g. in STEP *(STandard for Exchange of Product data)* for say form features [Shah et al. 1995]. However, *LCC inference* and *LCC action knowl*edge depends on domain specific life synthesis elements (PDEs and LCPEs) having certain structure. This section therefore discloses a basic formalism of component PDEs and LCPEs for supporting the inference of LCCs from *artefact life models* described with these elements.

8.2.2.1 PDE Models

Component level PDEs

Based on discussions made in section 3.1.5, for design at the component level, the basic properties manipulable by the designer are *form*, *material*, *surface finish, dimensions* and *tolerances*. Further, of relevance to mechanical component synthesis are assembly features e.g. a snap-fit. A PDE does not necessarily have geometric shape (e.g. material). In this sense, the definition of a feature found in [Weber 1996] fits in with the way PDEs are treated for component synthesis in this research - "A feature is an information unit

(element) representing a *region of interest* within a product". Therefore, PDEs relevant for a *synthesis elements library* for designing mechanical components are form features, assembly features, material, surface finish, dimensions and tolerances (Figure 8.4). These are discussed next.

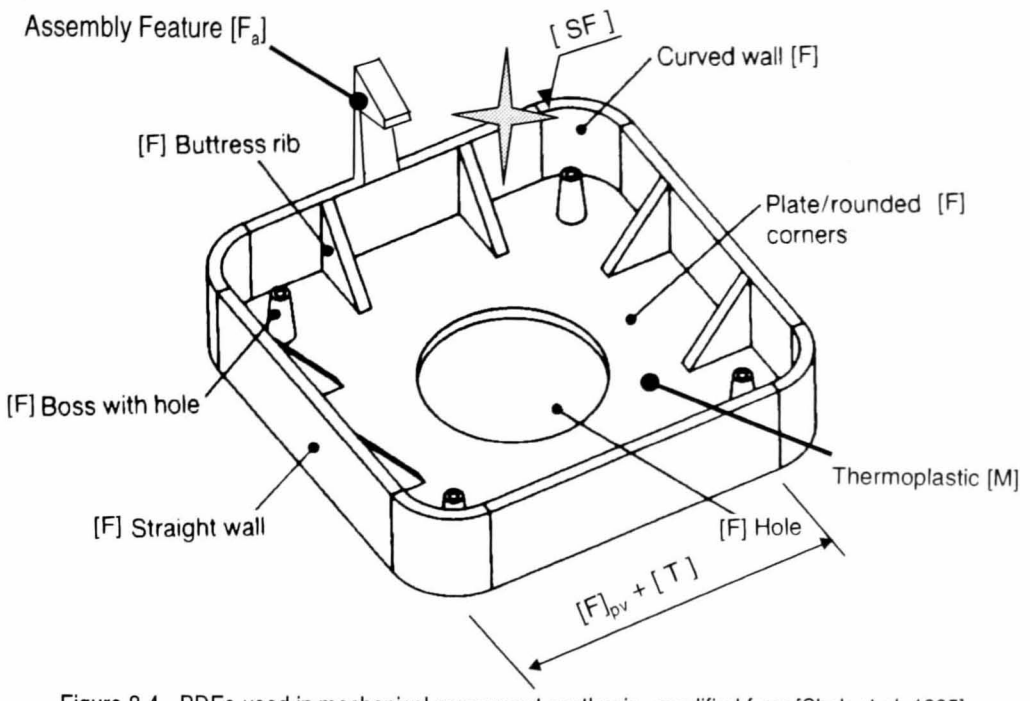


Figure 8.4 - PDEs used in mechanical component synthesis - modified from [Shah et al. 1995]

Form features

Influence

A component can be described from a constructional viewpoint in terms of *form features [F]* [Salomons et al. 1993; Shah et al. 1995] (Figure 8.4). A form feature e.g. a 'boss', allows commonly used geometric shapes to be associated with a set of properties relevant to an application. Of relevance to this research is that form has an influence on other life-phases e.g. a gear tooth profile influences the use phase or the form of an opening (circular versus prismatic) influences the fabrication process in the realization phase.

Form feature model

For this research, a form feature model [F] is considered to consists of a set of feature parameters:

(8.14)

$$[F] \Leftarrow \{ [F]_p \}$$

$$\text{e.g. } [\text{hole}] \Leftarrow \{ [\text{hole}]_{\text{centre\_point}} \wedge [\text{hole}]_{\text{radius}} \wedge [\text{hole}]_{\text{depth}} \wedge [\text{hole}]_{\text{angle}} \}$$

A feature is related to a component model with a *part_of* (→) relationship. One component can have *n (n ≥ 1)* features (Figure 8.4).

Dimensions

Influence

Dimensions are feature parameter values $[F]_{pv}$ (Figure 8.4). Dimensions can influence different phases (e.g. case 7, chapter 2) as argued in [Hubka et al. 1988]. For instance, the thickness dimension influences a component's

stiffness and the ease of assembly of mating components in the realization phase. Each feature parameter $[F]_p$ has an associated feature parameter value $[F]_{pv}$, formally as:

(8.15) $[F]_p \Leftarrow [F]_{pv}$ e.g. $[hole]_{radius} \Leftarrow 5.0$, written as $[hole]_{radius=5.0}$.

During early component design, designers may prefer to make minimum commitments, this meaning that form features [F] can be specified *vague* or *nil* parameter values for the feature parameters $[F]_P$.

Material

As case examples in Chapter 2 reflect, a material [M] can have consequences on different component life-phases. From the definition in section 2.1, one component has exactly one material. A material [M] constitutes a set {m} of mechanical and physical properties [Groover 1996] such as conductivity, magnetism and Young's modulus. Properties can be *qualitative* (e.g. non-magnetic) or *quantitative* (e.g. density = 0.25Kg/m^3). Therefore for the purpose of this research, a material PDE model [M] is :

(8.16) $[M] \Leftarrow \{m\}$ e.g. [ABS] $\Leftarrow$ { non-conductive$\wedge$...non-magnetic }

Assembly Features

An *assembly feature [F_a]* e.g. a snap-fit (Figure 8.4), is a PDE by which a component can be physically connected to another component. One component can have *n (n ≥ 1)* assembly features. They are intentionally used to allow a component to be assembled/dis-assembled during the realization and use (service system) phases. For this reason, assembly features are treated separately from form features. This research identified a number of assembly feature characteristics that generate $LCC_{ni,}$ as follows:

Joint strength
- Different assembly features result in different tensile, transversal and fatigue strengths [Beitz 1993] for the joint. For example, a snap-fit has an average tensile strength whilst a bolt/nut has a good tensile strength.

Joint dynamism
- An assembly feature influences the *degrees of freedom* between components joined, resulting in either a *static* or *dynamic* joint. For example a bolt/nut results in a static joint and a hinge in a dynamic joint.

Integrity
- Some assembly features are physically an integral part of the component (e.g. snap-fit) whilst others are not (e.g. fasteners) the latter thus increasing the number of parts in a sub-assembly.

Joint permanence

- Assembly features have both a joining and detaching behaviour [Beitz 1993]. This can result in a permanent, non-permanent or semi-permanent joint e.g. a weld feature, a snap-fit and a pop-rivet respectively.

Repetitivity

- Non-permanent assembly features have different assembly repetition characteristics. For example, a self-tapping screw is not suitable for applications where assembly and dis-assembly repetition is high.

$[F]_a$ model

Thus, an assembly feature model $[F_a]$ has an associated set $\{ [F_a]_c\}$ of assembly characteristics, formally:

(8.17)

$$[F_a] \Leftarrow \{ [F_a]_c\}$$

where {strength, repetitivity, dynamism, integrity, permanence} $\in$ $\{ [F_a]_c\}$

e.g. [bolt] $\Leftarrow$ {'assembly slow' $\wedge$ 'requires hole' $\wedge$ 'dis-assembly easy' $\wedge$....'non-permanent bond'}

Surface Finish

The 'use' failure of a component can be attributed to its surface [Robinson et al. 1993]. The designer can commit a specific surface finish [SF] to a form feature. A surface finish can have a qualitative (e.g. smooth, rough, textured) or quantitative (in terms of R_a [Haslehurst 1985]) value. That is, [SF] is associated with a form feature [F], formally:

(8.18)

$$[F] \Leftarrow [SF] \quad \text{e.g. } [\text{hole}] \Leftarrow ([SF]=\text{smooth})$$

Influence

A surface finish [SF] value can influence a number of component life-phases. For example, it influences the fabrication processes that can be employed in the realization phase, the aesthetic appeal, strength and component wear in the use phase [Hubka et al. 1988; Galea 1997].

Tolerances

To achieve component interchangeability, designers specify both linear and geometric feature tolerances [Haslehurst 1985]. Tolerances [T] are restrictions on the range, a feature parameter value $[F]_{pv}$ can posses once the feature is realized. Tolerances can have consequences on different life-phases e.g. they can influence the ease of assembly in the realization phase or ease of servicing in the use phase. Tolerance values are associated with feature parameter values (Figure 8.4), formally:

(8.19)

$$[F]_{pv} \Leftarrow [T] \qquad \text{e.g. } [\text{hole}]_{\text{radius}=5.0} \Leftarrow ([T] = \pm 0.01\text{mm})$$

8.2.2.2 LCPE Models

Different life-phase system model synthesis viewpoint

As disclosed by the artefact life solution modelling frame (Figure 7.5), LCPEs are employed for *life-phase system* compositional modelling. An LCPE model describes reusable, well-known transformation systems (e.g. milling system) that are encountered in an artefact's life. Life-phase systems can be viewed from *different perspectives*, this giving rise to *different life-phase system models* such as a 'mode of action model' [Mortensen et al. 1996], a 'process sequence model' [Andreasen et al. 1996b] and a 'system's physical structure model' [Mortensen et al. 1996]. Hence as with artefacts, the synthesis of a life-phase system can take place from *different viewpoints*. For example, Olesen [Olesen 1993] argues that the theory of domains used for describing an artefact solution is applicable for describing production systems. He states that the systems represented by the domains are: the *process system* i.e. those processes which transform materials, energy and information. The *function system* i.e. those functions which are necessary in order to realize the processes. The *organ system* i.e. the overall solutions and solution principles which realize the functions by exploitation of physical effects. The *constructional system* i.e. the equipment and the machines which form the physical fabrication system. This demonstrates that during life-phase synthesis, designers employ 'viewpoint-specific' system characteristics to describe different life-phase system models.

Viewpoint-specific life-phase modelling characteristics

As discussed in section 2.3.3, during its life, a mechanical artefact *interacts* with life-phase systems. These systems are the units (e.g. moulding machine) and elements (e.g. mould tool) that make up a *physical* (i.e. constructional) system delivering the *processing* effects that transform the artefact from one state to another during this interaction. Hence, as argued in Chapter 2, a life-phase $<\text{Phase}>_i$ is composed of a set of *transformation systems* delivering *transformation process [P]* effects. Thus, in this thesis, the concurrent synthesis of a life-phase system takes place from the process and constructional viewpoint. Hence, for life-phase system modelling, designers are concerned with manipulating process and constructional viewpoint design characteristics.

LCPE model foundation - Hubka & Eder's technical process model

The LCPE model developed in this research is based on Hubka & Eder's [Hubka et al. 1988] system model of a transformation process as this involves both process and constructional viewpoint characteristics. In this model (Figure 8.5), the input *operand* Od^1 receives effects based on a *process technology* $[P]_t$ (e.g. erosion by sparks) to be transformed into *operand state* Od^2. These transformation effects are delivered through *human beings* (e.g. machining operators) and *technical systems [TS]*, the latter basically decomposable into

executing systems (e.g. tooling) and *control systems*. As reflected by non-exhaustive examples in Table 8.1 derived from [Bonello 1997; Cutajar 1997; Galea 1997], Hubka & Eder's model applies to processes encountered in different mechanical artefact life-phases.

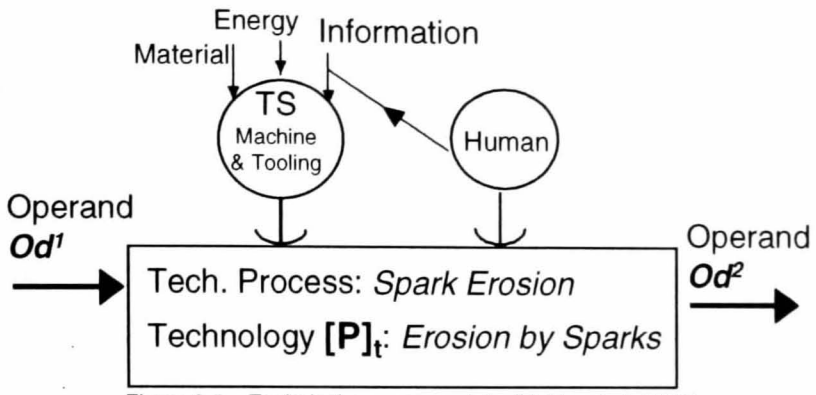


Figure 8.5 – Technical process model – [Hubka et al. 1988]

Table 8.1 Typical life-phase transformation processes

Life-Phase	[P]	$[P]_t$	Od^1	Od^2	Technological properties {t}
Realization	Spark Erosion (fabrication)	Material erosion by electric sparks	Initial component form	Different component form	Electrical conductor; low melting point; low hardness;
Realization	Sand casting (fabrication)	Material solidification	Raw material	Cast component	Achievable melting point; low molten fluidity;
Realization	Arc welding (assembly)	Electric power material fusion	Separate components	Bonded components	Electrical conductor; achievable melting point;
Use	Grit blasting (maintenance)	Chip formation by material abrasion	Dirty artefact surface	Clean artefact surface	Operand hardness < abrasive hardness
Disposal	Magnetic separation	Magnetic attraction	Mixed components	Separated components	Magnetic material

LCPE model characteristics

This section proceeds by presenting the characteristics used in declaring a *Life-Cycle Phase Element (LCPE) model* that supports the inference of LCCs from an evolving *component life model* (this inference is described later in section 8.2.3). Due to the boundary of this research, certain characteristics are omitted from this LCPE model e.g. the material of the elements making up a [TS], the human operator (e.g. a novice versus an experienced machining operator), the assembly features of elements making up the [TS], the control system of a [TS] and the environment in which the [TS] operates. LCPE characteristics that are considered sufficient in this thesis for supporting the inference of LCCs from the interaction of a conceptual mechanical component solution and a life-phase system conceptual solution are: *process technology, technological properties, process parameters, process parameter values, process minimum economic quantity, process technical (physical) system,* and *technical system parameters.*

Process Technology

Based on Hubka & Eder's model, we can say that a technical process [P] has (⇐) an associated process technology ($[P]_t$ by which an input operand is transformed, formally:

(8.20) $[P] \Leftarrow [P]_t$

e.g. ([P] = [Spark Erosion]) ⇐ ($[P]_t$ = Erosion by electric sparks)

Technological Properties

To deliver transformation effects, a process technology ($[P]_t$) requires (<>---) an operand (component) to have a compatible set of technological properties {t} :

(8.21) $[P]_t$ <>--- {t}

e.g. ($[P]_t$ = Erosion by electric sparks)<>---{conductive ∧... ∧ low hardness }.

Process Parameters

A transformation process [P] has an associated set of process parameters $[P]_p$:

(8.22) $[P] \Leftarrow \{ [P]_p \}$

e.g. ([P]=[milling]) ⇐ { $[milling]_{feed\ rate}$ ∧ ... $[milling]_{spindle\ speed}$ }.

Process Parameter Values

Each process parameter $[P]_p$ will eventually have a parameter value $[P]_{pv}$ specified, formally:

(8.23) $\forall [P]_p \in \{ [P]_p \} \Rightarrow [P]_p \Leftarrow [P]_{pv}$.

e.g. $[milling]_{feed\ rate} \Leftarrow [milling]_{feed\ rate=100}$.

Process Minimum Economic Quantity

A transformation process [P] is economically feasible for transforming a minimum quantity $(Q)_e$ of operands. Thus, in addition to Hubka's model we can say that a process [P] has an associated minimum economic quantity $(Q)_e$:

(8.24) $[P] \Leftarrow (Q)_e$

e.g. ([P]=[milling]) ⇐ ($(Q)_e \geq 1$) whereas due to the cost and time of preparing CNC part programs, for ([P]=[CNC_milling]) ⇐ ($(Q)_e \geq 50$).

Process Technical System

As reflected in Figure 8.5, the transformation effects of a process [P] are realized by a physical technical system [TS], the latter being a description of the transformation system from a constructional viewpoint:

(8.25) [P]<>--- { [TS] }

e.g.([P] = [moulding]) <>--- { [moulding machine] ∧ … ∧ [mould tool]] }

e.g.([P] = [automated assembly]) <>--- { [3-axis robot] ∧ … ∧ [gripper] }.

Technical systems [TS] (e.g. moulding machine) can be decomposed into *n (n ≥ 1)* finer constructional systems some of which have to be specifically designed & realized *(e.g. a mould tool)* for the operand being transformed by the process.

Process Technical System Parameters

Process [TS]s have their own set of parameters {$[TS]_p$}:

(8.26a) $[TS] \Leftarrow \{ [TS]_p \}$

e.g. $[\text{mould tool}] \Leftarrow \{ [\text{mould tool}]_{\text{length}} \wedge \ldots [\text{mould tool}]_{\text{breadth}} \}$.

Process Technical System Parameter Values

(8.26b) Each $[TS]_p$ requires during concurrent synthesis, the definition of a parameter value $[TS]_{pv}$, formally:

$\forall [TS]_p \in \{ [TS]_p \} \Rightarrow ([TS]_p \Leftarrow [TS]_{pv})$

e.g. $[\text{mould tool}]_{\text{length}} \Leftarrow [\text{mould tool}]_{\text{length}=100}$.

8.2.3 Component Life Model Based Reasoning

As this section now explains, through the established generic LCC knowledge models, an evolving *component life model* described by the component PDE and life-phase system formalism presented, supports the inference of co-evolving LCC_{ni}s and LCC_is.

Component modelling formalism

Building on the foundation provided by the *artefact life modelling frame* (section 7.3.3), the component PDEs (section 8.2.2.1) can be related as in Figure 8.6a to describe a component compositional model. Figure 8.6b illustrates an example.

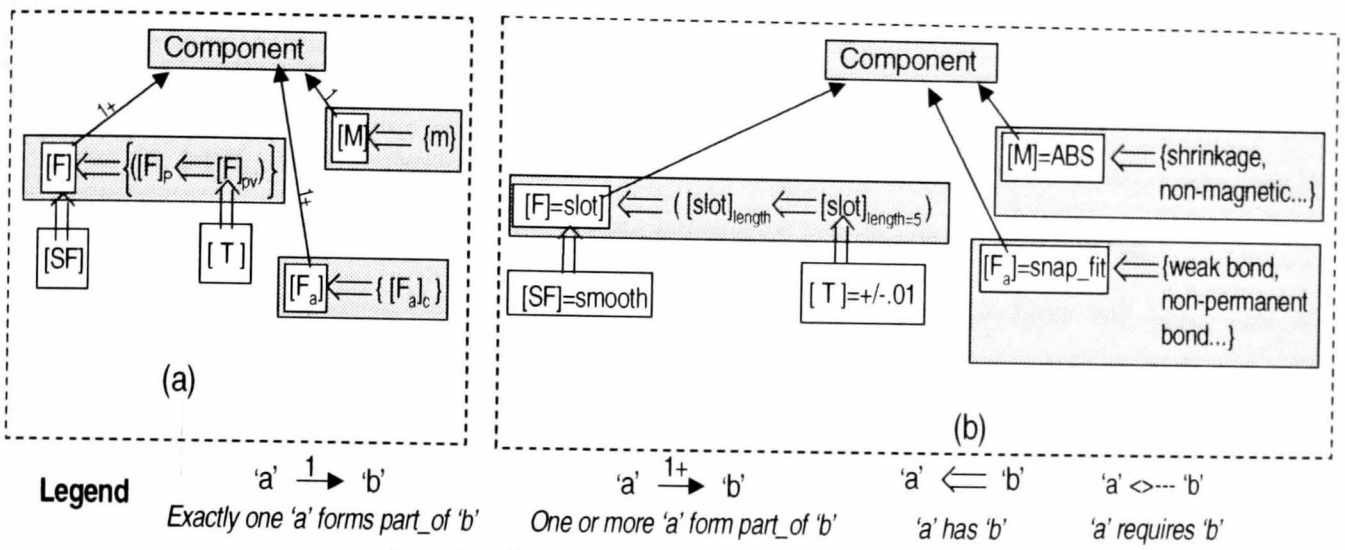


Figure 8.6 - Component compositional modelling formalism

Life-phase system modelling formalism

Similarly, using the foundation provided by the artefact life modelling frame (section 7.3.3), transformation processes forming part of a life-phase system compositional model can be described through the LCPE formalism presented in section 8.2.2.2 as in Figure 8.7a. Figure 8.7b illustrates an example.

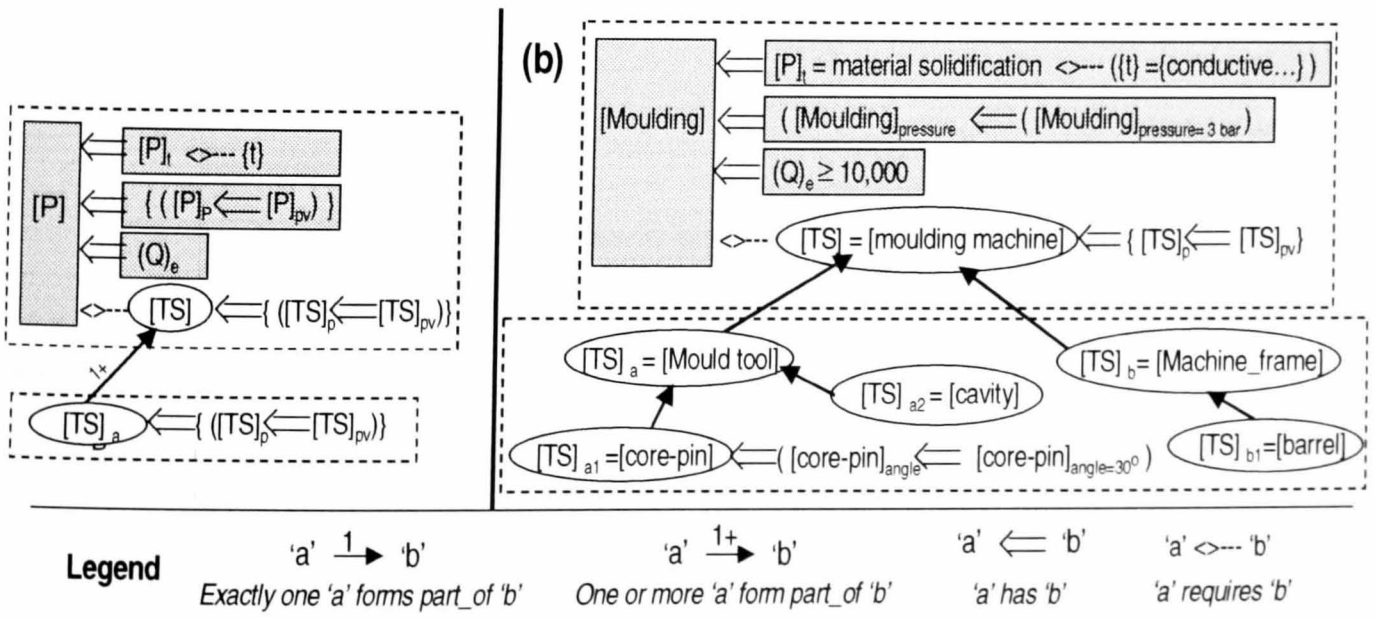


Figure 8.7 – Formalism of a transformation process for life-phase compositional modelling

Inferring LCC_{ni}

By modelling LCC_{ni} knowledge as in (8.1), LCC_{ni} can be explicitly inferred through formal relationship (6.1), when a synthesis decision commitment is made to the evolving *artefact life model.* The following are a few examples.

Inferring artefact behaviour

From (8.16) for a component made from ABS, it can be inferred that it will not conduct electricity in any life-phase since from (8.1)

$([d]_E\{[M]\}=[ABS]) \rightarrow <model>_{component}$

$\Rightarrow ([ABS] \Leftarrow \{\underline{\text{non-conductive}} \wedge \ldots \text{ non-magnetic }\}) \rightarrow <model>_{component}$

Inferring life-phase system behaviour

From (8.17), an artefact with a bolt assembly feature influences the behaviour of the realization phase since a hole generating process is required::

$([d]_E\{[Fa]\} = [bolt]) \rightarrow <model>_{artefact} \Rightarrow ([bolt] \Leftarrow \{\text{assembly slow} \wedge \underline{\text{requires hole}} \wedge \text{dis-assembly easy} \wedge \ldots.\text{non-permanent bond}\}) \rightarrow <model>_{artefact}$

Inferring new decision spaces

From (8.14), the minimum commitment of a circular hole will give rise to a decision space S= {what $[hole]_{centre\_point}$? $\wedge$ what $[hole]_{radius}$? $\wedge$ what $[hole]_{depth}$? $\wedge$ what $[hole]_{angle}$?} since:

$([d]_E\{[F]\}=[hole]) \rightarrow <model>_{component} \Rightarrow \quad ([hole] \Leftarrow \{[hole]_{centre\_point} \wedge [hole]_{radius} \wedge [hole]_{depth} \wedge [hole]_{angle}\}) \rightarrow <model>_{component}$

Inferring new life-phase constraints

From (8.24), if ([P] = [moulding]) $\Leftarrow$ (Q_e > 10,000), then the commitment of a moulding process to the realization phase co-evolves a restriction on the quantities [Q] that can be economically realized, to be over 10,000 since:

$([d]_E\{[P]\}=[moulding]) \rightarrow <phase>_{realization} \Rightarrow \underline{([moulding] \Leftarrow (Q_e > 10{,}000))} \rightarrow <phase>_{realization}.$

Inferring LCC_i

Assuming that the *LCC_i knowledge model* described by (8.3) is deterministic, then if at any time during solution synthesis,

(8.27)

$(\ (IR)_j \subseteq <life\ model>\ ) \Rightarrow (LCC_j)_i$

read as:

If the interacting relationship $(IR)_j$ between a set of synthesis commitments exists as a sub-set of the evolving artefact life compositional model <life model>, then the interacting LCC $(LCC_j)_i$ co-evolves.

LCCi inference example

Assume $(IR)_j = \{ [d]_E\{O\}_1=[A] \wedge [d]_E \{O\}_2=[B] \wedge [d]_E \{O\}_3=[C] \}$. This $(IR)_j$ does not exist in state 1 (Figure 8.8) but, when [C] is committed in state 2, based on relationship 8.27, $(LCC_j)_i$ will co-evolve. Relationship (8.27) therefore provides the basis for inferring a LCC_i *during* synthesis decision making.

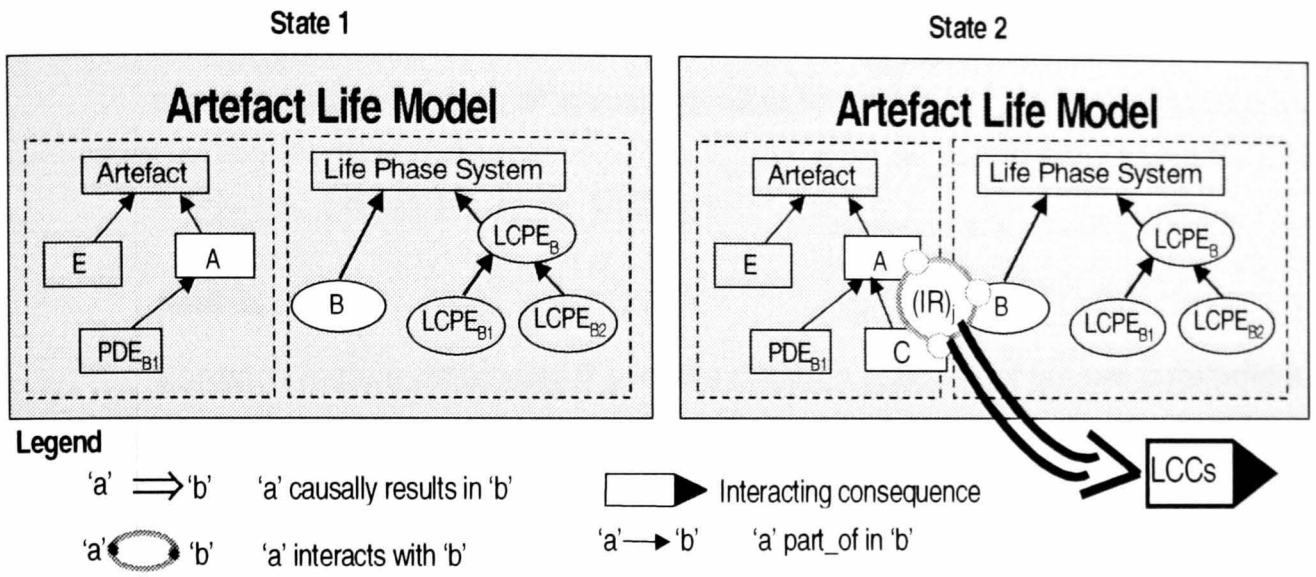


Figure 8.8 - Concept of inferring LCC_i during synthesis

For instance, from the example in (8.4), it can be inferred that for a still evolving component model, currently consisting of an ABS material and a rib form feature, together with a realization phase model currently consisting of an injection moulding process, the formation of a sink mark defect will occur.

8.3 Established Mechanical Component Domain Relationships

Based on design decision commitment examples and the generic LCC knowledge models disclosed in section 8.2.1, this research has identified a number of relationships concerning mechanical component PDEs and LCPEs, which describe domain LCC_{ni} and LCC_i knowledge. These relationships, presented next, provide patterns that can be customized for specific mechanical component domains.

8.3.1 Form Feature Commitment Consequences

Decision space

Based on (6.1), a LCC_{ni} resulting from a form feature synthesis commitment is a decision space S, formally:

(8.28)

$$([d]_E\{ [F] \} \rightarrow <model>_{component}) \Rightarrow (LCC_{ni} = (S= D\{ [F]_p \}))$$

read as:

A synthesis commitment concerning a form feature i.e. $[d]_E$ [F] made to the component model, gives rise to a decision space (S) concerning feature parameters $[F]_p$.

e.g.($[d]_E${[F]}=[hole])=> (LCC_{ni}= (S={ D {$[hole]_{radius}$?} ∧ D{$[hole]_{depth}$?} ∧ D{$[hole]_{angle}$?} ∧D{$[hole]_{centre\text{-}point}$?} }))

Decision proposal

Further, each feature parameter $[F]_p$ (e.g. $[hole]_{angle}$) can be assigned a value from a set of possible values (e.g. {0 to 360°}). Therefore, a LCC_{ni} is that each feature parameter in the decision space S propagates a decision proposal concerning the selection of a specific *value*, formally:

(8.29)

$$\forall [F]_p \in (S= D\{ [F]_p \}) => D\{ [F]_{pv} \}$$

read as:

For each feature parameter $[F]_p$ in the decision space S of feature parameters, there exists a decision proposal $D\{[F]_{pv}\}$ with a set of options for the value $[F]_{pv}$.

8.3.2 Material Commitment Consequences

From (8.16), a material [M] has an associated set of properties {m}. A designer commits a specific material [M] due to a number of intended properties $\{m_k\}$, with $\{m_k\}$ a subset {m}. However, the material commitment also results in a set $\{m_j\}$ of unintended properties (Figure 8.9). An example is the commitment of a thermoplastic material due to its low-density and non-conductive properties. However, the component will however also be non-magnetic.

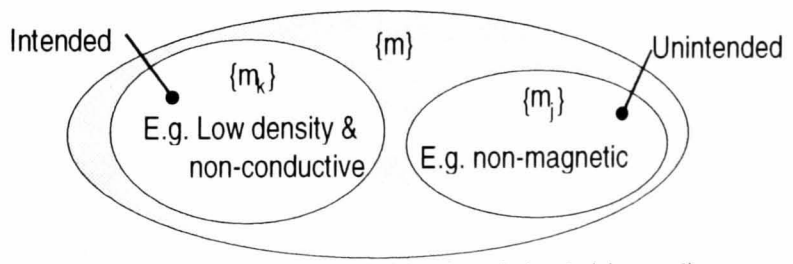


Figure 8.9 – Intended & unintended material properties

Further, a LCC propagating from this unintended property is that it will be difficult to explicitly separate the component with a magnetic separation process in the disposal phase. That is {m} is a LCC_{ni} introduced by a material commitment:

(8.30)

$$([d]_E\{ [M] \rightarrow <model>_{component}) => LCC_{ni}= ((\{m_k\} + \{m_j\}) \rightarrow <model>_{component})$$

read as:

A non-interacting consequence (LCC_{ni}) resulting with a material commitment $[d]_E\{ [M]\}$ is the set of intended $\{m_k\}$ and the set of unintended $\{m_j\}$ material properties.

8.3.3 Material: Technical Process Interaction

The commitment of a material [M] influences the set of technical processes (e.g. spark erosion, arc-welding, magnetic separation) that a component can interact with during its life. Therefore, from (6.1), a LCC_{ni} of a material commitment is a decision proposal concerning a set of compatible processes {[P]}, formally:

(8.31) $([d]_E \{[M]\} \rightarrow <model>_{component}) \Rightarrow (LCC_{ni} = D \{[P]\})$

read as:

The synthesis commitment of a material $[d]_E$ {[M]} to a component gives rise to a decision proposal D {[P] } concerning compatible technical processes.

(8.32) e.g. $([d]_E \{[M] = [mild\_steel]\} \rightarrow <model>_{component}) \Rightarrow (LCC_{ni} = D\{ [milling] \vee [turning] \vee [arc\_welding] \vee [magnetic\ separation] \})$

Thus relationship (8.31) results in a *restriction* on the members in the set of {[P]} options to those that are only compatible. This provides guidance to the space of feasible options that can be explored during concurrent synthesis. However, utilizing relationship (8.31) as in (8.32) for each material available on the market is not computationally practical e.g. (8.32) will have to be updated to cater for compatibility of a *[mild_steel]* material with say a *[spark erosion]* process. Rather, the material PDE formalism of (8.16) and the technical process model formalism (Figure 8.7), collectively allow compatible life-synthesis elements to be inferred for an evolving *artefact life model* through a relationship established in this research:

(8.33) $(\{t\} \subset \{m\}) \Rightarrow ([P] >\text{----} [M])$

read as:

if the set {t} of process technological properties is a sub-set of the component's material properties {m}, then the process [P] is suitable for material [M].

Without the need to explicitly model relationships between specific material instances and specific technical process instances, relationship (8.33) allows compatible processes [P] to be inferred when a material [M] is committed. This *material:process* compatibility relationship applies to different processes employed during the component life such as 'arc welding' for the realization phase and 'magnetic separation' for the disposal phase. The following example explains how this works.

Compatibility example

Based on (8.16), assume that the following material PDE model is known:

([mild_steel])⇐ ({m}={magnetic∧ (melting_point=1600^{O}C) ∧ conductive ∧ ductile ∧ low_hardness}).

From relation (8.21), it is also known that:

([P]$_t$ = Erosion by electric sparks)<>---({t}={conductive ∧...∧ low hardness}).

Then from the above and relationship (8.33) we see that:

({t}={conductive ∧...∧ low hardness}) ⊂ {m}.

That is, ([P]= [Spark Erosion]) is compatible with [mild_steel], this being a consequence revealed with the material commitment in (8.31). Similarly, other compatible processes are revealed.

Non-compatibility example

Assume now that a *new* different material [M'], a *kind of* thermoplastic is introduced on the market, modelled as:

[M'] ⇐ ({m'} = {non-magnetic ∧ (melting_point=400^{O}C) ∧ non-conductive ∧ ductile ∧ low_hardness}).

Then, from relationship (8.33) we see that:

({t}= {conductive ∧...∧ low hardness}) ⊄ {m'}.

In this case, spark erosion is *not* compatible with the [M'] thermoplastic , since:

(8.34)

$$\{t\} \not\subset \{m'\} \Rightarrow \neg ([P] >\text{----} [M'])$$

read as:

If the process technological properties {t} are not a subset of the material properties {m'}, it implies that technical process [P] is not (¬) compatible with material [M'].

Since artefact life commitments interact, from relationships (8.28) and (8.31) we can collectively say that:

(8.35)

$$([d]_E\{ [F]\} \rightarrow <\text{model}>_{\text{component}} \wedge ([d]_E \{[M] \} \rightarrow <\text{model}>_{\text{component}}) \Rightarrow D'\{[P]\} \wedge (S= D\{[F]_P\})$$

read as:

A form feature commitment $[d]_E\{[F]\}$ interacts with a material commitment $[d]_E\{[M]\}$, giving rise to a decision proposal D'{[P]} concerning feature-material compatible technical processes together with a feature parameter decision space ($S = D\{ [F]_P\}$).

It should therefore be noted that *D' {[P]}*, which is a type of LCC_i, is a sub-set of *D{[P]}* in (8.31). For example for a mild steel material, (8.31) can result in *D{[P]} = D { [drilling]* $\vee$ *[milling]* $\vee$ *[spark erosion]}* but for a *rectangular slot* in mild steel, the option set is restricted, *D' {[P]} D { [milling]* $\vee$ *[spark erosion]}*, this illustrated in Figure 8.10. Relationship (8.35) is thus important as exploring alternative form features may result in a drastic change of compatible processes.

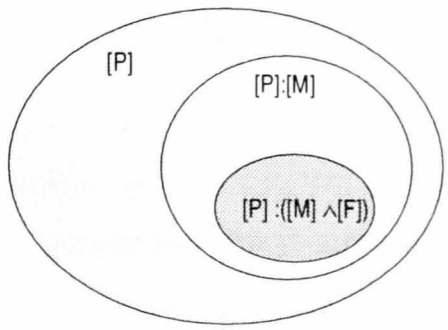


Figure 8.10 - Restricted space of compatible technical processes

8.3.4 Technical Process Commitment Consequences

Once a technical process [P] is committed to a life-phase model, a number of consequences can result, formally:

(8.36) $([d]_E\{[P]\} \rightarrow <phase>_i) => \{ LCC \}$

read as

A technical process commitment $[d]_E\{[P]\}$ *results in a number of possible life cycle consequences.*

New decision space

For example, from relationship (8.36) and (8.22), an example of LCC_{ni} is the decision space concerning process parameters, formally:

(8.37) $([d]_E\{[P]\} \rightarrow <phase>_i) =>(S= D\{ [P]_P \})$

New decision proposal

Now, each process parameter $[P]_P$ can be assigned a value from a set of alternative values. For instance, the parameter $[milling]_{feed\ rate}$ has a range of possible values. That is, each parameter in the decision space S eventually requires the commitment of a specific *value*, formally:

(8.38) $\forall [P]_P \in (S= D\{ [P]_P \}) => D\{ [P]_{pv} \}$

New constraints

Since from (8.25), [P] <>---[TS], a specific technical system [TS] introduces process restrictions. For example, using (8.36), different circular hole generating processes can result in different restrictions on $[hole]_{diameter}$ ranges:

i. (([d]$_E${ [P]} = [twist_drilling])→ <phase>$_{realization}$) => (LCC_{ni}) = { 5mm < [hole]$_{diameter}$ <55 mm}

ii. (([d]$_E${[P]}= [laser_drilling]) → <phase>$_{realization}$) => (LCC_{ni}) = {0.05mm< [hole]$_{diameter}$ <25mm}

Restriction on processing quantities

From (8.24), a technical process has an associated economic quantity $(Q)_e$. Therefore, from (8.36), the commitment of a process results in a restriction:

($[d]_E${[P]} → <phase>$_i$) => (LCC_{ni} = (d_E{[Q]} > $(Q)_e$))

read as

The commitment of a technical process $[d]_E${[P]} restricts the commitment of use-phase quantity [Q] to be greater than the process' economic quantity $(Q)_e$.

Otherwise, the commitment by the designer of a smaller number of components [Q] required in the use phase, results in a non-economically feasible process:

(8.39) ($[d]_E${[Q]}< $(Q)_e$) => ¬ $[d]_E$[P]

read as:

If the component quantity commitment $[d]_E${ [Q] } is less than the process' economic quantity $(Q)_e$, then a consequence is that the commitment $[d]_E$[P] of that technical process is not (¬) feasible.

Also, from (8.25), the commitment of a technical process introduces a technical system formally:

(8.40) ($[d]_E${[P]} → <phase>$_i$) => (LCC_{ni} = ([P]<>---{[TS]})→ <phase>$_i$)

read as

The commitment of a technical process $[d]_E${[P]} to a life-phase <phase>$_i$ is associated with the commitment of a technical system [TS] to the same life-phase.

(8.41) e.g. ($[d]_E${[moulding]} → <phase>$_i$) => (LCC_{ni} = ([moulding]<>---{[mould tool]})→ <phase>$_i$)

8.3.5 Assembly Feature Commitment Consequences

From (8.17), an assembly feature introduces a number of LCC_{ni}, formally:

(8.42) ($[d]_E$ {[F_a]→ <model>$_{artefact}$) => (LCC_{ni} ∈ {Joint strength, repetitivity, dynamism, integretiy, permanence})

read as:

The commitment of a specific assembly feature ($[d]_E$ $\{[F_a]\}$) results in a set of non-interacting consequences concerning the joint strength, repetitivity, dynamism, integrity and permanence.

e.g. ($[d]_E\{[F_a]\}$ = [bolt]) → $<model>_{artefact}$ => (LCC_{ni} = ([bolt] ⇐ {assembly slow ∧ requires hole ∧ dis-assembly easy ∧....non-permanent bond}) → $<model>_{artefact}$)

Also, the commitment of an assembly feature gives rise to a LCC_{ni} concerning compatible assembly process, resulting in a decision proposal, formally:

(8.43) $([d]_E\{[F_a]\} \rightarrow <model>_{artefact}) \Rightarrow (LCC_{ni} = D\{[P]\})$

read as:

The commitment of an assembly feature ($[d]_E$ $\{[F_a]\}$) results in a decision proposal concerning assembly process [P] for use during the realization phase.

Some examples, derived from [Bonello 1997] are provided in Table 8.2. The set of assembly processes {[P]} frequently has only one option, this being a *concurrent synthesis pattern* thereby making the commitment automatic.

Table 8.2 – Assembly features and required assembly processes

Assembly Feature $[F]_a$	**Required assembly process [P]**
Weld	Welding
Rivet	Rivetting
Snap-fit	Insertion
Screw	Fastening

Since an assembly feature also has form, its commitment results in a LCC_{ni} concerning a decision space of form feature parameters, formally:

(8.44) $([d]_E\{[F_a]\} \rightarrow <model>_{artefact}) \Rightarrow (LCC_{ni} = (S=D\{[F]_p\}))$

read as:

The commitment of an assembly feature ($[d]_E$ $\{[F_a]\}$) gives rise to a decision space 'S' concerning assembly feature form parameters $[F]_P$.

(8.45) e.g. $([d]_E\{[F_a] = [screw]\} \rightarrow <model>_{artefact}) \Rightarrow$

$(LCC_{ni} = (S= \{ D\{[screw]_{diameter}?\} \wedge D\{[screw]_{length}?\} \ldots \wedge D\{[screw]_{pitch}?\}\}))$

A LCC_{ni} with some assembly features is the commitment of form features into the artefact model, this described through the concurrent synthesis pattern:

(8.46) $([d]_E\{[F_a]\} \rightarrow <model>_{artefact}) \Rightarrow ([d]_E\{[F]\} \rightarrow <model>_{artefact})$

e.g. $(([d]_E\{[F_a]\} = [rivet]) \rightarrow <model>_{artefact}) \Rightarrow (([d]_E\{[F]\}=[hole]) \rightarrow <model>_{artefact})$.

8.3.6 Surface Finish Commitment Consequences

The commitments of a surface finish and a material interact to restrict the set of possible fabrication processes, formally:

(8.47) $([d]_E\{[SF]\} \rightarrow <model>_{component}) \wedge ([d]_E\{[M]\} \rightarrow <model>_{component}) \Rightarrow (LCC_i = D\{[P]\})$

read as

The commitment of a surface finish $[d]_E\{[SF]\}$ together with the material commitment $[d]_E\{[M]\}$ give rise to a decision proposal $D\{[P]\}$ of feasible technical processes.

Alternatively, depending on the sequence of commitments made:

(8.48) $([d]_E\{[P]\} \rightarrow <phase>_{realization}) \wedge ([d]_E\{[M]\} \rightarrow <model>_{component}) \Rightarrow (LCC_i = ([SF] \rightarrow <model>_{component}))$

read as:

The component material commitment $[d]_E\{[M]\}$ interacts with the fabrication process $[d]_E\{[P]\}$ to restrict a component surface finish [SF].

Examples based on public knowledge found in [Groover 1996] illustrate how during concurrent synthesis, different processes [P] can be explored to reveal the [SF] range for a component made from a material [M]:

$([d]_E\{[P]\}=[green\_sand\_casting]) \wedge ([d]_E\{[M]\}=[Aluminium])$

$\Rightarrow (LCC_i = ([SF] = [6 - 25\mu m]) \rightarrow <model>_{component})$

whereas:

$([d]_E\{[P]\}=[investment\_casting]) \wedge ([d]_E\{[M]\}=[Aluminium])$

$\Rightarrow (LCC_i = ([SF] = [0.75 - 2.5\mu m]) \rightarrow <model>_{component})$.

8.3.7 Tolerance Commitment Consequences

The commitments of a tolerance and a material interact to restrict the set of possible fabrication processes, formally:

(8.49) $[d]_E\{[T]\} \wedge (\,[d]_E\{[M]\} \rightarrow \langle model\rangle_{component}\,) \Rightarrow (LCC_i = D\{[P]\})$

read as:

The commitment of a tolerance $[d]_E\{[T]\}$ of some feature parameter value for a component with material commitment $[d]_E\{[M]\}$ gives rise to a decision proposal of tolerance feasible fabrication processes $D\{[P]\}$.

A tight tolerance value can for instance restrict the set of feasible processes {[P]} to finishing processes such as grinding and honing. Alternatively, based on 8.19, depending on the sequence of commitments made:

(8.50) $([d]_E\{[P]\} \rightarrow \langle phase\rangle_{realization}) \wedge ([d]_E\{[M]\} \rightarrow \langle model\rangle_{component})$

$\Rightarrow (LCC_i = ([F]_{pv} <= [T]))$

read as:

The component material commitment ($[d]_E\{[M]\}$) interacts with the process commitment ($[d]_E\{[P]\}$) to restrict tolerances [T] for feature parameter values.

Examples of (8.50) based on *public knowledge* in [Groover 1996] are:

$([d]_E\{[P]\}=[green\_sand\_casting] \rightarrow \langle phase\rangle_{realization}) \wedge ([d]_E\{[M]\}=[Cast\_Iron] \rightarrow \langle model\rangle_{component}) \Rightarrow (LCC_i = ([F]_{pv} <= ([T]=\pm 1.0mm)))$

whereas

$([d]_E\{[P]\}= [investment\_casting] \rightarrow \langle phase\rangle_{realization}) \wedge ([d]_E\{[M]\}=[Cast\_Iron] \rightarrow \langle model\rangle_{component}) \Rightarrow (LCC_i = ([F]_{pv} <= ([T]=\pm 0.25mm)))$.

8.3.8 Parameter Value Commitment Consequences

The commitment of parameter values to an artefact or life-phase system model can interact with other life-phase commitments, this giving rise to LCC_is. Typical examples encountered in this research concern interactions with technical processes, formally :

(8.51) $[d]_E\{[F]_{pv}\} \wedge ([d]_E\{[P]\} \rightarrow \langle phase\rangle_{realization}) \Rightarrow LCC_i$

read as:

The detailing synthesis commitment ($[d]_E\{[F]_{pv}\}$) of a form feature parameter value can interact with the commitment of a technical process ($[d]_E\{[P]\}$), to give rise to an interacting LCC.

Example of detailing commitment consequences

The case example in Figure (2.19) and formally illustrated in Figure (8.11) shows how a value ($\neq 0$) of the parameter $[hole]_{angle}$ in a thermoplastic component influences the parameter value of *a core-pin,* this an LCPE part of the mould tool system, formally :

(8.52)

$$([d]_E \{ [hole]_{angle}\} = 30^o) \wedge ([d]_E \{[P]\} = [moulding] \rightarrow <phase>_{realization}) \Rightarrow$$

$$LCC_i = ([d]_E \{ [core\text{-}pin]_{angle}\} = 30^o)$$

where from figure 8.11, [core-pin]→ [mould tool].

Further, this $[core\text{-}pin]_{angle}$ propagates affects to various performance measures (see case 7 in section 2.3.4.3). From (8.5), this can be modelled as:

$$LCC_i = ([d]_E \{ [core\text{-}pin]_{angle}\} = 30^o)$$

$$\Rightarrow ((cost)<phase>_{(mould)\ design} = +8$$

$$\wedge (time)<phase>_{(mould)\ realization} = +5 \quad \wedge (time)<phase>_{(component)\ realization} = +3)$$

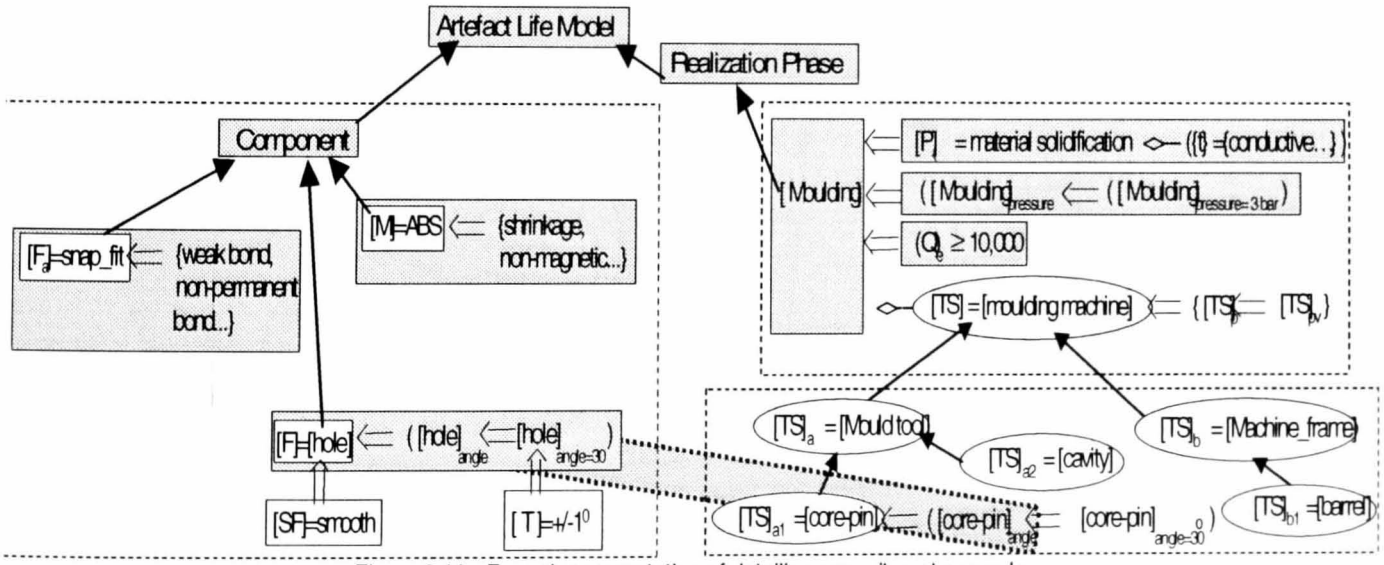


Figure 8.11 - Formal representation of detailing commitment example

8.4 LCC Knowledge Structuring Concept

Relevance of structuring

The generic LCC knowledge models presented in (8.1) and (8.3) are useful to capture and model LCC knowledge as demonstrated in section 8.3. However, for a relevant piece of LCC knowledge to be utilized at the *right time* during synthesis, a LCC knowledge base for a domain specific *synthesis elements*

library needs to be appropriately structured. LCC knowledge structuring is thus important to support providence with both least and specific synthesis decision commitments and also to address LCC knowledge management issues. Since a number of different synthesis elements can result in a similar LCC for which action knowledge is the same, knowledge structuring is concerned with LCC inference knowledge, as discussed next.

8.4.1 Structuring of LCC_{ni} Inference Knowledge

Classification

Rationalization mechanisms [Kerr 1993a] can be employed to organize different PDEs and LCPEs stored in a *synthesis elements library* (Figure 7.3) into sets of elements having similar characteristics. For example, Figure 8.12a demonstrates a non-exhaustive set of assembly features organized into sub-sets by the type of their permanence characteristic. These sets can be clustered into further sub-sets. Thus, such a *class* rationalisation process enables synthesis elements to be organized into *kind_of* taxonomies (Figure 8.12b).

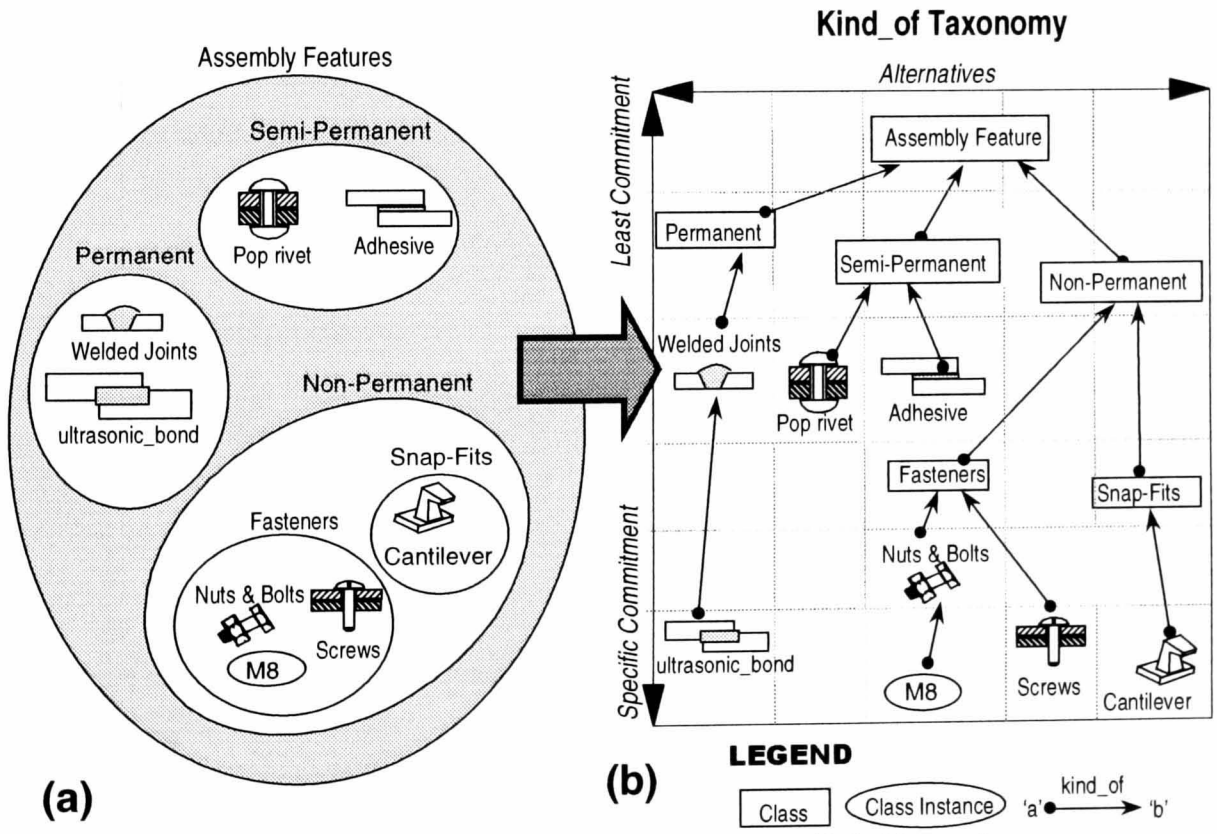


Figure 8.12 – Concept of organizing synthesis elements into *Kind_of* taxonomies

Kind_of taxonomies

A *kind_of* taxonomy allows synthesis elements at the top of the hierarchy to be more abstract than those lower down. For instance, we can say instance [M8] •→ ⌊Nuts&Bolts⌋. A *class* is a generalized set of elements. For instance the class ⌊Fasteners⌋ is a generalization of the set {⌊Nuts&bolts⌋, ⌊Screws⌋}. Similarly, the class ⌊Non-Permanent⌋ is the generalization of the set {⌊Fasteners⌋, ⌊Snap-Fits⌋}. This also means that reusable elements can be organized into classes and sub-classes. Sub-classes are different from a

mathematical sub-classes - they represent a specialization of the concepts but also an expansion of the information about the element [Walters et al. 1988].

Exploration benefits

Through *kind_of* taxonomies (Figure 8.12b), designers can explore *alternative* synthesis elements (e.g. pop-rivet versus M8 bolt) and make commitments at the desired *level of commitment* (e.g. M8 bolt versus ⌊Fasteners⌋).

Inferential Structure

The significance of these *kind_of* taxonomies is that the generic LCC_{ni} knowledge model in (8.1) allows LCC_{ni} knowledge to be associated to different abstraction levels in the taxonomy (Figure 8.13). For instance,

(8.53) ⌊Non-Permanent⌋ ⇐ (LCC_{ni} = 'dis-assembly easy').

At a more specific level in the taxonomy, more specific LCC_{ni} knowledge can be associated, as in the following examples:

(8.54) ⌊Fasteners⌋ ⇐ (LCC_{ni} = 'assembly slow' ∧ 'requires hole' ∧ 'more parts')

(8.55) ⌊Snap-Fits⌋ ⇐(LCC_{ni} = 'weak bond' ∧ 'assembly fast' ∧ 'poor repetitiveness')

LCC_{ni} knowledge inheritance

This concept of associating LCC_{ni} knowledge with relevant classes reduces knowledge duplication, making it attractive from a knowledge management point of view. For example, there is no need to explicitly associate *'requires hole'* with instance [M8], as this will be *inherited* from (8.54) since from Figure 8.13:

(8.56) [M8] •→⌊Fasteners⌋

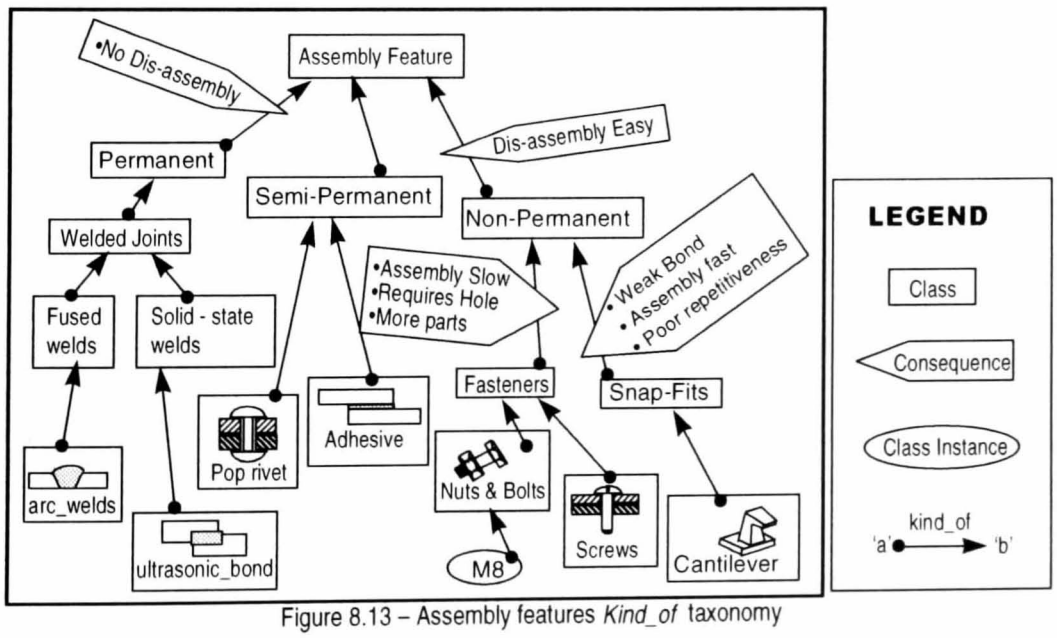


Figure 8.13 – Assembly features *Kind_of* taxonomy

PDE taxonomies

Thus, *kind_of* taxonomies provide a suitable way of how PDEs and LCPEs can be organized to result in a structure with LCC_{ni} inferential capabilities. This concept of structuring PDEs with relevant LCC_{ni} knowledge is illustrated for *form features* and *engineering materials* in Figure 8.14a and 8.14b respectively.

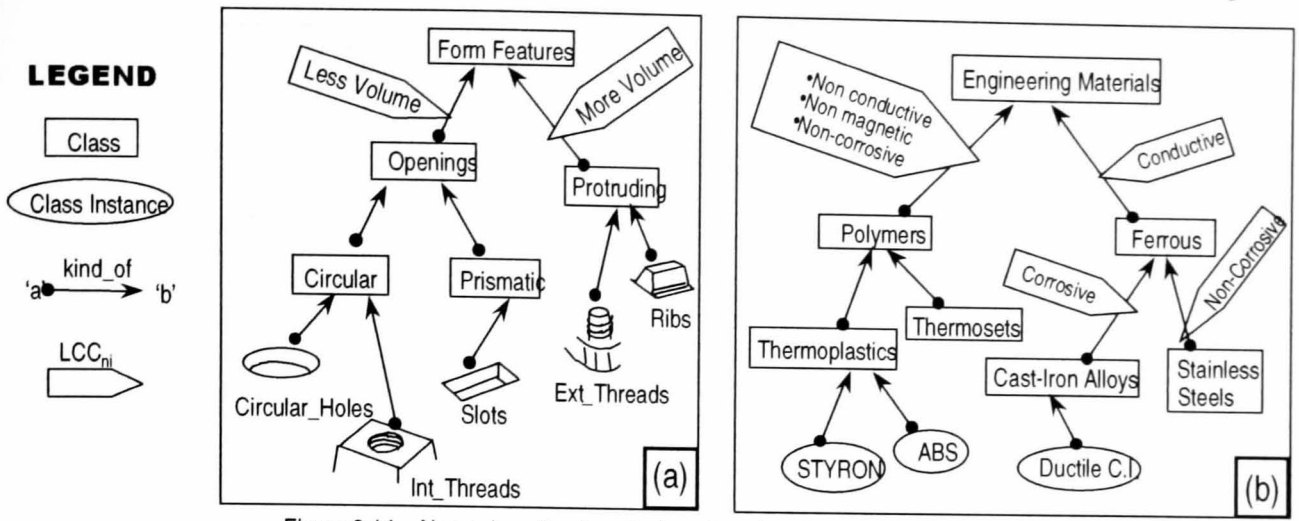


Figure 8.14 – Non-exhaustive form feature & engineering material taxonomies

LCPE taxonomies

LCPEs and relevant LCC_{ni} knowledge can be similarly associated to *kind_of* taxonomies. For example, Figure 8.15a and Figure 8.15b provide non-exhaustive taxonomies for fabrication LCPEs and disposal LCPEs respectively.

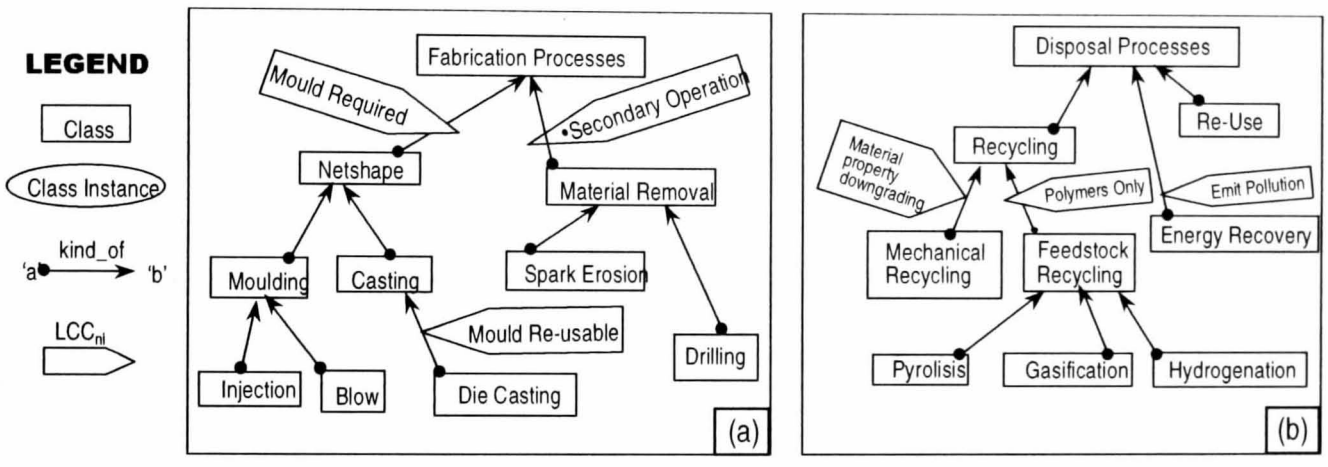


Figure 8.15 - Non-exhaustive fabrication and disposal process taxonomies

Union Inheritance

In situations where a class or instance has more than one-parent class belonging to either the same or a different taxonomy, *Kind_of* taxonomies can exploit *union inheritance* [Walters et al. 1988] (Figure 8.16). An example of a class having parent classes in *different* taxonomies is ⌊Snap-fits⌋ in Figure 8.16,

(8.57) (⌊Snap-fits⌋•→⌊Non-Permanent⌋) ∧ (⌊Snap-fits⌋•→⌊Protruding⌋)

(8.58) Now from Figure 8.16,

⌊Protruding⌋ ⇐ (LCC_{ni} = 'more volume')

Thus, from (8.57), the class ⌊Cantilever⌋, a sub-class of ⌊Snap-fits⌋ (Figure 8.16), will inherit from the assembly features' taxonomy the knowledge as described in (8.53) and (8.55) together with *'more volume'* from (8.58). ⌊Pop rivet⌋ in Figure 8.16 is an example of a class having two parents within the *same* taxonomy, that is:

(8.59) (⌊Pop rivet⌋ •→ ⌊Semi-Permanent⌋) ∧ (⌊Pop rivet⌋ •→ ⌊Fasteners⌋.

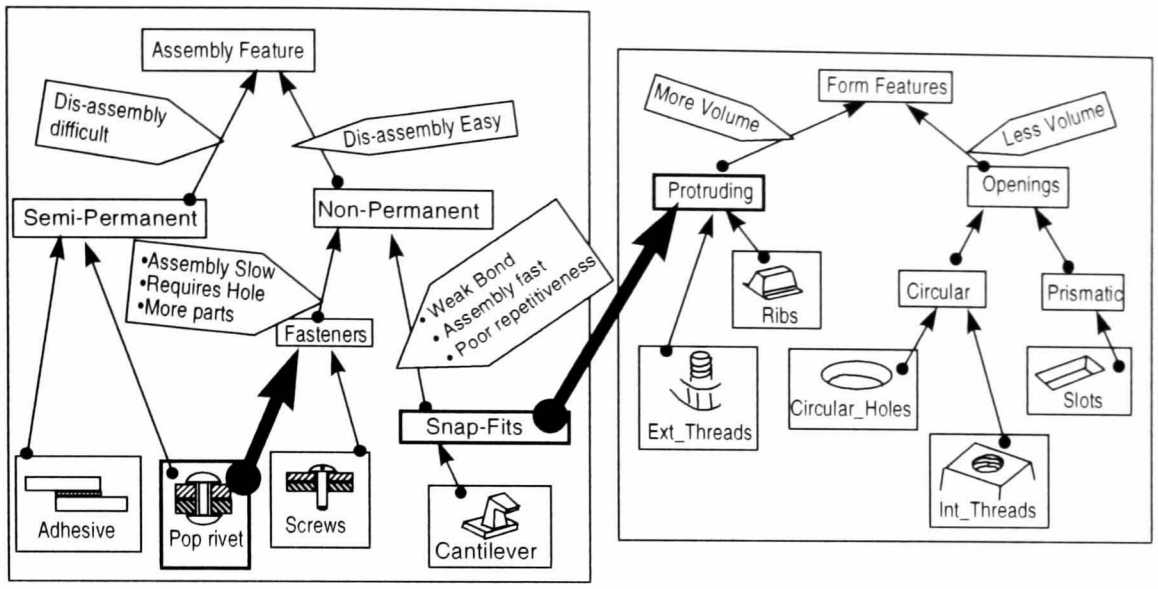


Figure 8.16 - Exploiting the concept of union inheritance

Careful use to avoid conflicting inheritance

Whilst taxonomies and union inheritance reduce knowledge duplication, they require careful use and the *validation* [Rich et al. 1991] of inherited LCC_{ni} knowledge . For example, ⌊Pop rivet⌋ inherits LCC_{ni}s as described in (8.54), and 'dis-assembly easy' as described in (8.53). However, the latter contradicts 'dis-assembly difficult' inherited from ⌊Semi-Permanent⌋ (Figure 8.16).

8.4.2 Structuring of LCC_i Inference knowledge

Structuring concept

Relationship (8.3) reflects that *LCC_i knowledge* can be modelled by the causality between an interacting relationship and a LCC_i. From section 8.4.1, an interacting relationship therefore concerns commitments about PDEs or LCPEs that belong to different *classes* in *kind_of* taxonomies. Thus *LCC_i knowledge* can be generalized by defining interacting relationships $(IR)_j$ for sets of *class-based* decision commitments, rather than specific commitments, formally:

(8.60)

$$(IR)_j = \{ ([d]_E\{O\}_1 \bullet\!\!\rightarrow \lfloor class \rfloor_1) \wedge \ldots ([d]_E \{O\}_j \bullet\!\!\rightarrow \lfloor class \rfloor_j)$$

Utilizing the interaction relationship structure of (8.60) with (8.3) allows interaction relationships forming between commitments of sub-classes and instances to also be a valid $(IR)_j$. For instance, the sink mark example in (8.4) can be generalized as in Figure 8.17, formally:

(8.61)

$$(\lfloor Protruding \rfloor \wedge \lfloor Thermoplastics \rfloor \wedge \lfloor Solidification\_based \rfloor) \Leftarrow$$

$$\{ (LCC_i = \text{'sink mark defect'} \rightarrow <model>_{component}) \}$$

Thus, the LCC_i knowledge in (8.61) is equally applicable to a component made from Styron *(a kind of ⌊Thermoplastics⌋ material)* having a snap-fit assembly feature *(from 8.57, this also a kind of ⌊Protruding⌋ form feature)* that is going to be injection moulded *(a kind of ⌊Solidification_based⌋ fabrication process)*.

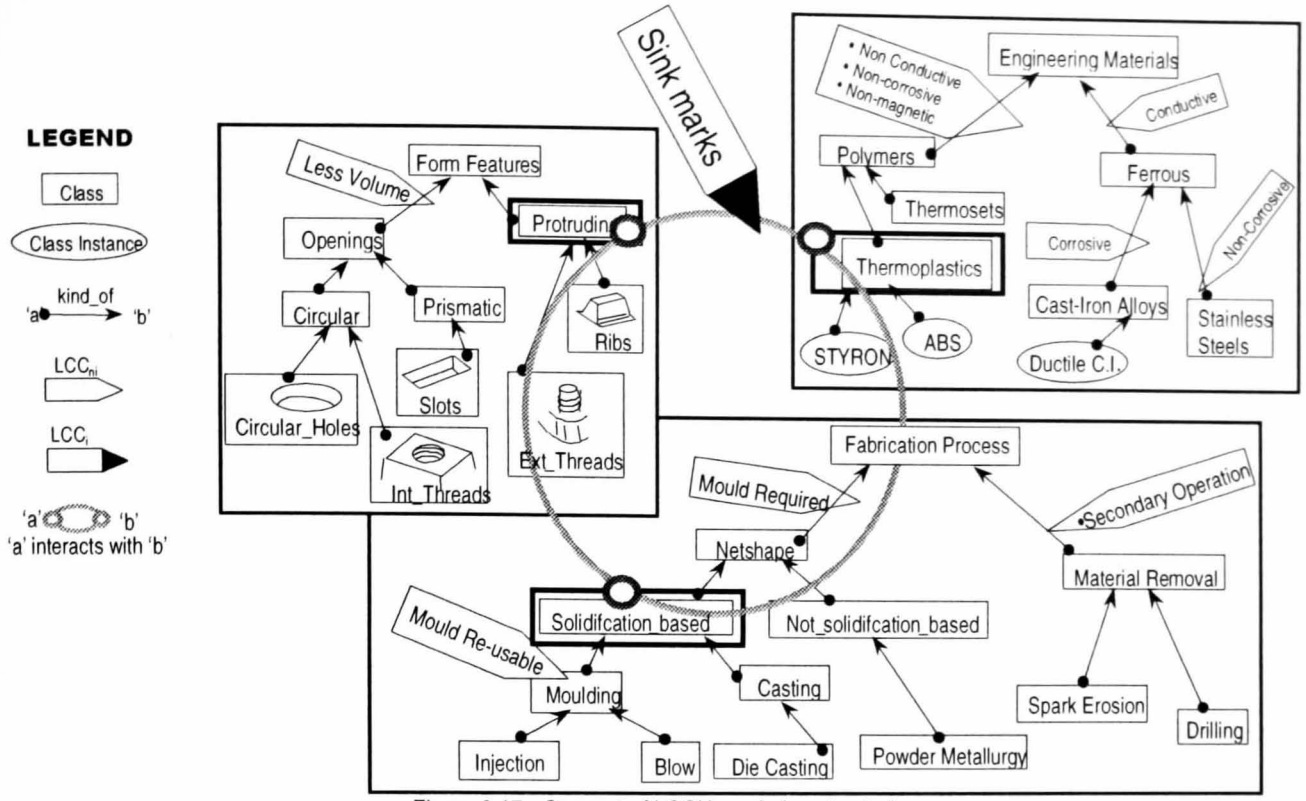


Figure 8.17 - Concept of LCCi knowledge structuring

Structure's strengths & limitations

Thus such a LCC_i knowledge structure reduces LCC_i knowledge duplication. Further, from (8.27), it allows designers to be supported in foreseeing and exploring LCC_i with either *specific* or *least* commitments. However, for generalizing an IR, classes selected can have an influence on the LCC_i knowledge inherited. For instance, if the relationship in 8.61 is modelled as stated on ⌊Solidfication Based⌋, then even blow moulding instances can give rise to sink mark defects which in reality is not the case. Thus, as with LCC_{ni} knowledge structuring, validation is required to ensure that the LCC_i knowledge has not been modelled at a too general level.

8.4.3 Towards A Customizable LCC Knowledge Structure

Beyond a fixed LCC knowledge perspective

The organization of elements into *kind_of* taxonomies is implicitly based on some *perspective* (Kerr et al. 1992). The work presented in this research does not employ the concept of customized perspectives – rather *kind_of* taxonomies used are based on a fixed perspective. This is sufficient to explore the 'KC' approach concept in this research, because at any one time, the designer only uses one perspective. However, in practice, a 'fixed perspective' *kind_of* taxonomy is insufficient, if artefact life specific LCCs are to be foreseen, explored and catered for *during* design. For example, different suppliers can provide elements (e.g. fasteners, end-mill cutters or standard mould parts) in different size ranges and material compositions. Further, performance measures (e.g. delivery time, cost, quality) associated with an element will differ from one supplier to another [Swift et al. 1994]. Thus, a customizable structure is required if designers are to explore LCCs associated with such *bought out items*.

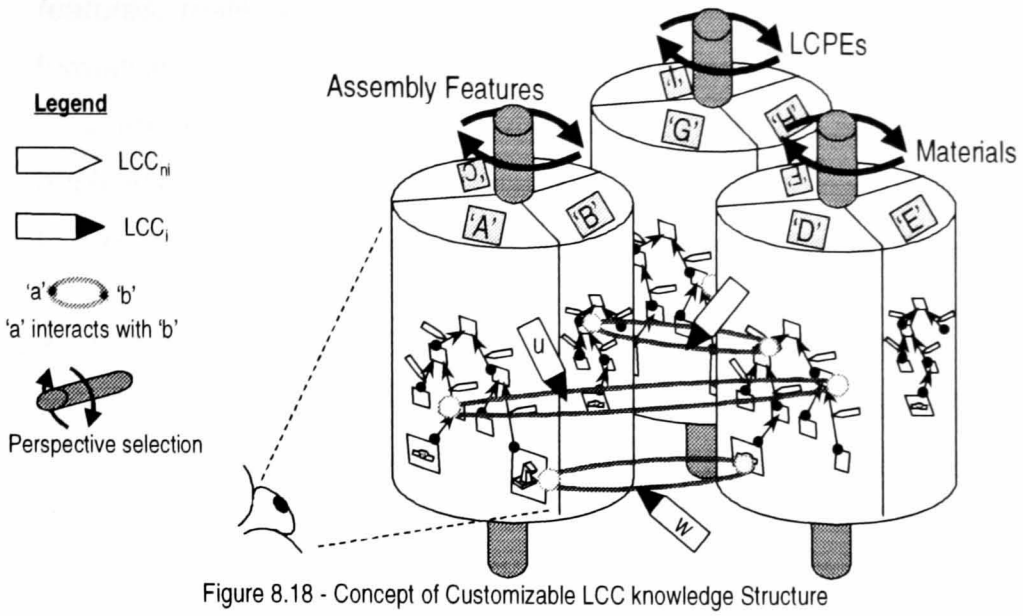


Figure 8.18 - Concept of Customizable LCC knowledge Structure

Concept of an explorable LCC knowledge structure

The *concept* of a customized LCC knowledge structure is demonstrated in Figure 8.18. A designer can for instance explore committing assembly features, provided by different suppliers, by first selecting a supplier perspective (e.g. supplier 'A' or 'B' or 'C'). This perspective provides a supplier-specific library of elements, the latter modelled and structured as discussed in this chapter. Similarly, perspectives on other elements (e.g. materials and LCPEs) could be specified. For the perspectives selected, an assembly feature will therefore introduce its specific LCC$_{ni}$ and specific LCC$_i$.

8.5 Chapter Conclusions

Formal relationships

The purpose of this chapter was to disclose in non-computational form the elements and relationships that make up a mechanical component domain LCC knowledge model. The chapter did not discuss *how to represent* LCC knowledge in computational form. Section 8.2.1 thus presented generic formal relationships that model :

a) LCC inference knowledge for
   - *non-interacting consequences (relationship 8.1);*
   - *for interacting consequences (relationship 8.3);*

b) LCC action knowledge for
   - *consequence to performance measure mapping (relationship 8.5);*
   - *concurrent synthesis patterns (relationship 8.11);*
   - *explanation/guidance.*

Mechanical component life synthesis elements

Section 8.2.2 than presented a basic formalism of relevant synthesis elements to specifically generate a LCC knowledge model for the mechanical component domain. The vocabulary in this formalism consists of *form features*, *assembly*

features, *materials*, *dimensions*, *tolerances* and *surface finish*. In addition, a formalism for LCPEs has been presented to allow life-phase systems to be concurrently synthesized. The vocabulary presented can be tailored for specific applications where *specific* elements are employed. For example, through the LCPE formalism, company specific technical processes can be modelled.

Formalism supports model based reasoning

As shown in section 8.2.3, the resultant element formalism can be employed to generate *component life models* that can be reasoned upon through modelled LCC knowledge, to infer co-evolving LCCs. To demonstrate the utility of the generic formal relationships in modelling LCC knowledge, section 8.3 presented a number of mechanical component domain relationships.

LCC knowledge structuring concepts

To support the retrieval of relevant LCC knowledge at the *right time*, with both least and/or specific commitments, LCC knowledge structuring concepts have been presented in section 8.4. As discussed, this structure also reduces LCC knowledge duplication. However, further work is required to provide a knowledge structure with a customizable perspective.

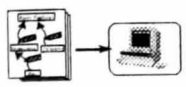
Computational model in FORESEE prototype

The issue of how to codify into computational form, LCC knowledge utilizing the established knowledge modelling and structuring concepts, will be disclosed in the next chapter where an implementation of the 'KC' approach framework is realized as a KICAD prototype named *FORESEE*.

chapter 9

9.0 FORESEE - A KICAD Prototype Implementation

Scope



This chapter presents how the 'KC' approach framework can be implemented into computational form as a KICAD prototype system. For this purpose, section 9.1 presents a set of relevant system requirements for realizing the framework as a KICAD tool. Implementation issues including LCC knowledge representation and application domain selection are discussed in section 9.2. The architecture of the implemented prototype, FORESEE, is then presented and discussed in section 9.3. Chapter conclusions are made in section 9.4.

9.1 KICAD Tool Requirements

Based on the frames making up the 'KC' approach framework presented in section 7.3 together with the LCC knowledge modelling and structuring concepts presented in Chapter 8, this section discloses with reasons, the general requirements of a KICAD tool for supporting $D_sF\Sigma X$.

Flexible synthesis sequence

The KICAD system should not restrict the order in which a designer works since observations made in this research correlate with arguments by [Nilsson 1997] that there is no definite answer to the *sequence* to which comes first in component design: form synthesis, material selection or process selection. The tool should provide designers the freedom to design in any order.

Concurrent synthesis

Due to the operational frame principles, the tool should support designers in the concurrent generation of the artefact and life-phase compositional models.

Concurrent & integrated solution modelling

Based on the 'artefact life modelling frame', the artefact and life-phase system models should not only be generated concurrently, but, the co-evolving models should be integrated (Figure 9.1) to enable LCC_i to be revealed.

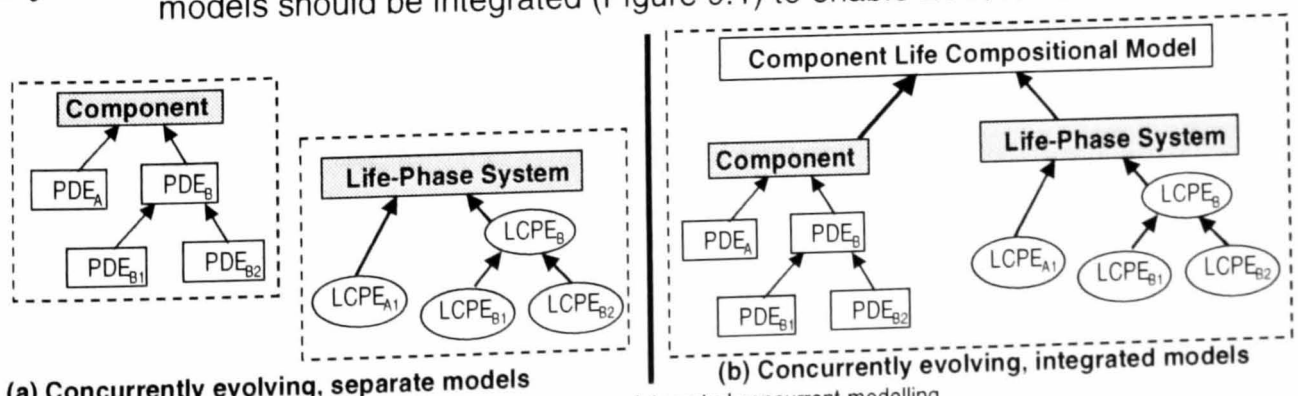


Figure 9.1 – Separate versus integrated concurrent modelling

Supports evolutionary modelling

Based on the operational frame principles, the KICAD system should allow designers to make least commitments (e.g. material [M] is a *kind_of* Thermoplastic) or specific commitment (e.g. [M] = ABS) to both the artefact and life-phase system models. This is necessary if the tool is to *continuously* support component life synthesis and exploration from the early design stages. Thus, it should support *evolutionary solution modelling* (Figure 9.2) due to both concretization and detailing commitments (section 6.2) being made during synthesis.

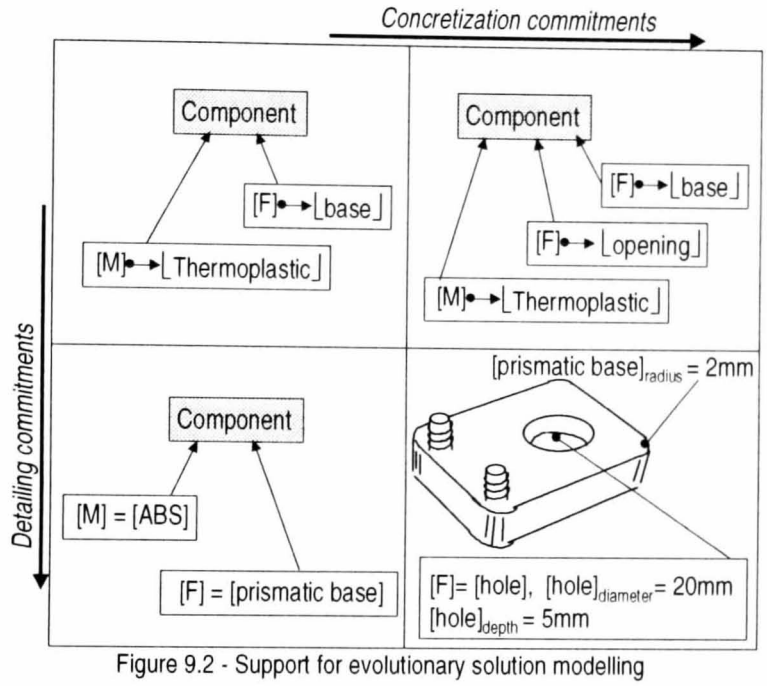


Figure 9.2 - Support for evolutionary solution modelling

Providence with inaccurate and incomplete models

To overcome a limitation of existing means (section 4.4), the KICAD tool should support synthesis based providence by inferring LCCs from an inaccurate solution model (e.g. component has a 'hole' without parameter values) and also with an incomplete solution model (e.g. component consists of a hole only). This is applicable to both artefact and life-phase models.

Supports monitoring multi-X behaviour

To overcome a limitation of current means that provide a narrow and segmented insight into LCCs (section 4.4), it is necessary that the behaviour of *multiple* life-phases in terms of performance measures be *continuously* and *concurrently* estimated. Further, the estimates should be presented to the designer for multi-X behaviour monitoring purposes.

Should provide pro-active support

As with current means, foreseeing artefact life interactions is difficult to achieve (section 4.4) and due to human LCC knowledge processing limitations (section 7.2.1), the KICAD tool needs to take a significant role in component $D_sF\Sigma X$. The tool should therefore provide:

- *guidance in the retrieval of relevant and feasible option sets:* e.g. the tool should help designers retrieve relevant assembly features that result in a non-permanent bond;

- *timely awareness of co-evolving LCCs*: during synthesis the tool should work independently from the designer, to check for complex interactions between the numerous life commitments made to the model, reporting back any solution specific LCCs *co-evolving* with the decisions committed;
- *explanation of LCCs revealed*: suitable and relevant explanations need to be provided if a designer is to be motivated in avoiding/relaxing detected LCC that relate to artefact life issues about which a designer may not be knowledgeable. For example, simply informing a designer that due to some commitment made to a thermoplastic component, a 'side-core' will need to be used in a mould tool, may not allow a designer to appreciate the severity this has in terms of design or construction, time and costs;
- *LCC avoidance support*: designers should be guided to *those* commitments which could be explored in order to avoid or relax a specific LCC. Thus, the distinction between LCC_{ni} and LCC_i *should* be exploited.
- *concurrent synthesis support:* by exploiting captured *concurrent synthesis patterns,* the KICAD tool should work in the background, independently from the designer, to evolve relevant solution models and where necessary propagate co-evolving LCCs.

Synthesis element reuse library

Due to the operational frame's reuse principle, the KICAD tool should provide designers with an application domain library of well-developed *synthesis element* models. Due to concurrent synthesis, the library should therefore consist of application specific PDEs (e.g. form features) and LCPEs (e.g. assembly systems, mould tools).

Static and dynamic synthesis element models

Certain elements do not change in behaviour with a change in time or environmental conditions e.g. a model of a circular hole form feature. However, some elements do change and thus need to be modelled to reflect their dynamic nature e.g. a material, whose properties like stiffness change with exposure to different temperatures. If artefact life-specific aspects are to be catered for, such changes with PDEs/LCPEs need to be foreseen.

Company specific knowledge re-use

To support company specific $D_sF\Sigma X$, the tool must posses *public* and also *private* LCC knowledge concerning *company-specific* :

- *synthesis elements:* e.g. a model of a specific milling machine, MILL_01;
- *LCC inference knowledge:* e.g. knowledge that milling more than 20 cast-iron components per day on machine MILL_01 causes a significant dimensional inaccuracy due to machine vibration unlike with milling the components on machine MILL_02;

- *LCC action knowledge* – *e.g.* life-phase system performance measures will vary from one company to another. Designers need this specific knowledge to foresee company specific *mutli-X behaviour* in order to generate a 'company life-oriented' artefact solution.

Knowledge management facilities

Since LCC knowledge has a dynamic nature (section 7.1) then to avoid obsolescence of captured knowledge, this a weaknesses of KBS means (section 4.3.4), the KICAD tool should provide management facilities allowing users to update and/or tailor captured LCC knowledge.

Customizable knowledge perspective

As discussed in section 8.4.3, different synthesis elements (e.g. materials) can be acquired from different suppliers. To support life-specific synthesis and exploration, the KICAD tool should allow users to select a perspective (e.g. knowledge about materials acquired from supplier 'A').

Maintains truth

As reflected by the operational frame (section 7.3.1) both decision commitment and retraction are involved. Therefore, if designers are to explore qualitatively as many alternatives as possible within a given project time frame, the KICAD tool should take care of maintaining/updating dependencies when a designer commits or retracts synthesis decision commitments. The KICAD system should therefore support the *truth maintenance* of causal relationships [Yan et al. 1995] for:

- *consequences* - retracting a synthesis decision commitment should be associated with the retraction of a causally related LCC_{ni} (Figure 9.3);

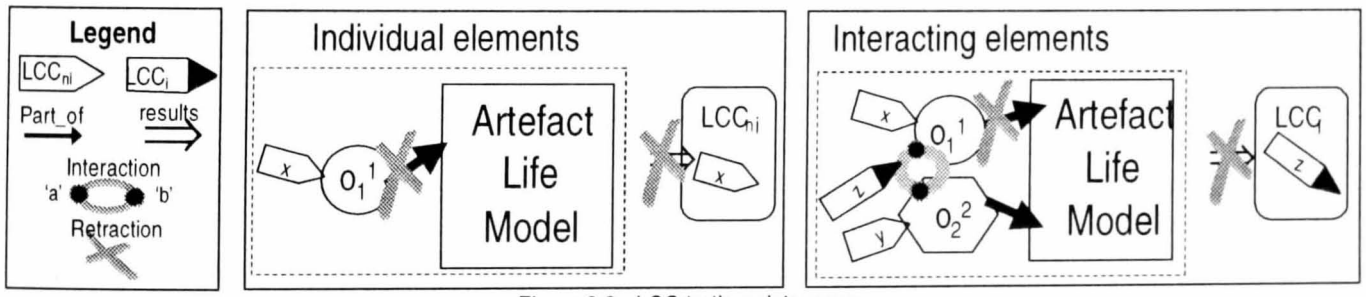


Figure 9.3 - LCC truth maintenance

- *component and life-phase system models*: a change to a component model (e.g. different thermoplastic material) may require a life-phase system model to be updated e.g. mould's core-pin diameter (Figure 9.4).

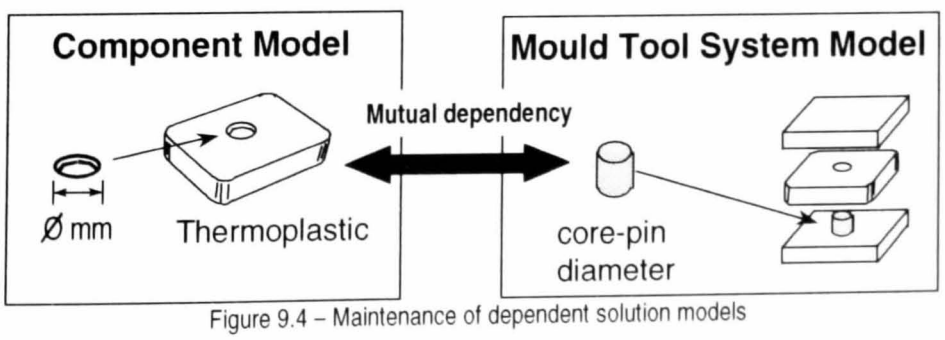


Figure 9.4 – Maintenance of dependent solution models

- *performance measures* - the retraction of a LCC should result in an 'inverse' update of relevant performance measures influenced by that LCC as illustrated in Figure 9.5;

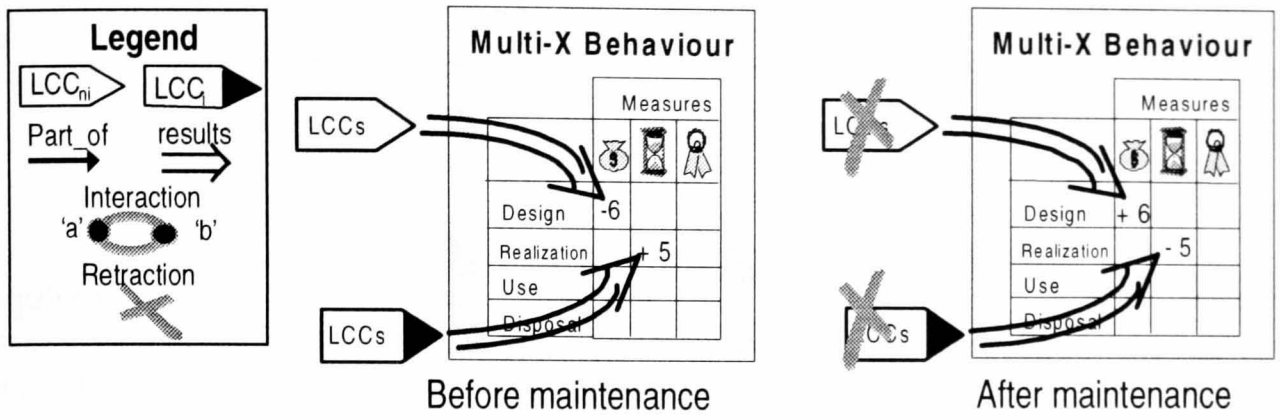


Figure 9.5 - Performance measure value truth maintenance

Maintains a design session history

Maintaining a history of synthesis decision commitments made and/or retracted with the related fluctuations in life-phase behaviour is relevant for design review meetings, for recording the solution alternatives explored, for learning and for communicating the concurrent synthesis intent. Automating this record keeping task avoids wasting a designer's time and effort.

A $D_sF\Sigma X$ user interface

To provide a $D_sF\Sigma X$ environment that creates a *partnership* with the designer the KICAD tool's user interface is critical. In essence, the interface should facilitate the *acquisition* of information 'from the designer' and the *presentation* of information inferred by the tool 'to the designer'. For $D_sF\Sigma X$, the interface therefore needs to allow designers to :

- define/manipulate compositional solution models at the required level of synthesis decision commitment;
- view the co-evolving 'artefact' and 'life-phase system' models;
- specify search queries to the 'reuse' synthesis elements library;
- view the results of queries made to the re-use library;
- continuously view changes in the artefact's 'virtual' behaviour during solution synthesis;
- view relevant information providing explanations about LCCs detected;
- view the design session history;
- update/manage the LCC knowledge base.

9.2 Prototype Implementation Issues

The identified system requirements were used to realize a KICAD prototype system named FORESEE (alias 4C – 'C'onsciousness of life-'C'ycle 'C'onsequences 'C'ommitments). In realizing this prototype, a number of implementation issues had to be decided upon, namely *level of implementation, component domain, knowledge representation* together with *development software and hardware.* These are discussed next.

Table 9.1 - KICAD system requirements and level of implementation

Requirement	FORESEE
Allows designers to adopt a flexible synthesis sequence	✓✓✓
Supports concurrent artefact and life-phase system synthesis	✓✓
Supports concurrent & integrated solution modelling	✓✓✓
Supports the reuse of artefact life solutions e.g. PDEs, LCPEs with	
• static models	✓✓✓
• dynamic models	✘
Supports both least/specific and concretization/detailing commitments	✓✓✓
Supports the evolution of the design solution model	✓✓
The system should provide pro-active support through	
• guidance in the retrieval of relevant option sets {O}	✓
• timely awareness of LCCs related to synthesis commitments	✓✓
• rapid exploration	✓✓
• explanation of co-evolving LCCs	✓✓✓
• LCC avoidance support	✓✓
• concurrent synthesis	✓✓
Supports company-specific knowledge re-use	✓✓
Provides on-line knowledge maintenance support for LCC_{ni} knowledge	✓✓
Provides on-line knowledge maintenance support for LCC_i knowledge	✘
Supports a customizable Knowledge Perspective	✘
Provides a suitable user interface	✓✓
Supports truth maintenance of LCCs, solution models, performance measures	✓✓
Supports the monitoring of fluctuations in multiple life-phase performance	✓✓✓
Automatically maintains a synthesis decision commitment & LCC history	✓✓✓

✓✓✓=Supported ✓✓=Partially Supported ✓=Limited Support ✘= Not Supported

9.2.1 Level of Implementation

Prototype level of implementation

Within the scope and time frame of this research, there was a need to prioritize, which of the identified KICAD requirements (Table 9.1) were essential, which were less required and which could be safely omitted. Features that have been omitted as they are considered essential from a practical point of view but not necessary for this research purposes are

dynamic synthesis element models, on-line LCC_i knowledge maintenance and a customizable knowledge perspective. Other KICAD requirements have been implemented to different levels as outlined in Table 9.1

9.2.2 Component Domain

For the following reasons, the domain selected for prototype implementation concerns thermoplastic components:

Domain selection reasons

- plastic component design provides a suitable case for exploration due to alternative PDEs e.g. assembly features and materials;
- plastic component design solutions can influence a number of total life activities e.g. mould design & construction, fabrication, assembly systems and disposal systems;
- public, domain knowledge is easily available e.g. in text books [Pye 1989] and material supplier guidelines [DuPont 1992];
- various thermoplastic component designers and artefact life-actors were easily accessible for consultation, this making it possible to acquire private LCC knowledge e.g. mould tool construction difficulties, artefact assembly difficulties etc.;
- it covers a large range of practical applications (e.g. computer enclosures, hi-fi enclosures, domestic appliances, make-up cases).

9.2.3 Knowledge Representation Issues

Reasoning with codified knowledge

For a KICAD tool to take a pro-active role in $D_sF\Sigma X$, it must reason about the current solution state with the LCC knowledge stored in the knowledge base to infer LCCs co-evolving with the current solution state. Reasoning is concerned with [Smith 1990; Popovic et al. 1994] 'The drawing of inferences or conclusions from known or assumed facts'. Different knowledge *representation schemes* (see Appendix D) commonly supported by AI development tools support reasoning in different ways. For a *KICAD* system, the way the knowledge acquired can be represented is therefore a key issue. To disclose FORESEE's knowledge representation scheme, this section therefore presents first a non-exhaustive background to *how* frames and *production rules* support reasoning.

Reasoning with Frames

Inheritance Mechanism

As explaned in Appendix D, frames allow a chunk of declarative knowledge about an object/concept to be conveniently structured and represented (e.g. the frame Fastener in Figure 9.6). Further, they can be organized into *kind_of* taxonomies (see Appendix D). Frames at lower levels of a *kind_of* taxonomy can obtain knowledge from frames at higher levels in the taxonomy. This mechanism of passing such knowledge from one frame to lower frames in a taxonomy, from *general* to *specific*, is known as *inheritance* [Walters et al. 1988]. Besides allowing new knowledge about an object to be inferred, inheritance allows a lot of repetition in a knowledge base to be avoided thus enhancing the storage efficiency [Rychener 1988]. Both slot *attribute* (e.g. Name) and *values* (e.g. Fastener) can be inherited by lower frames.

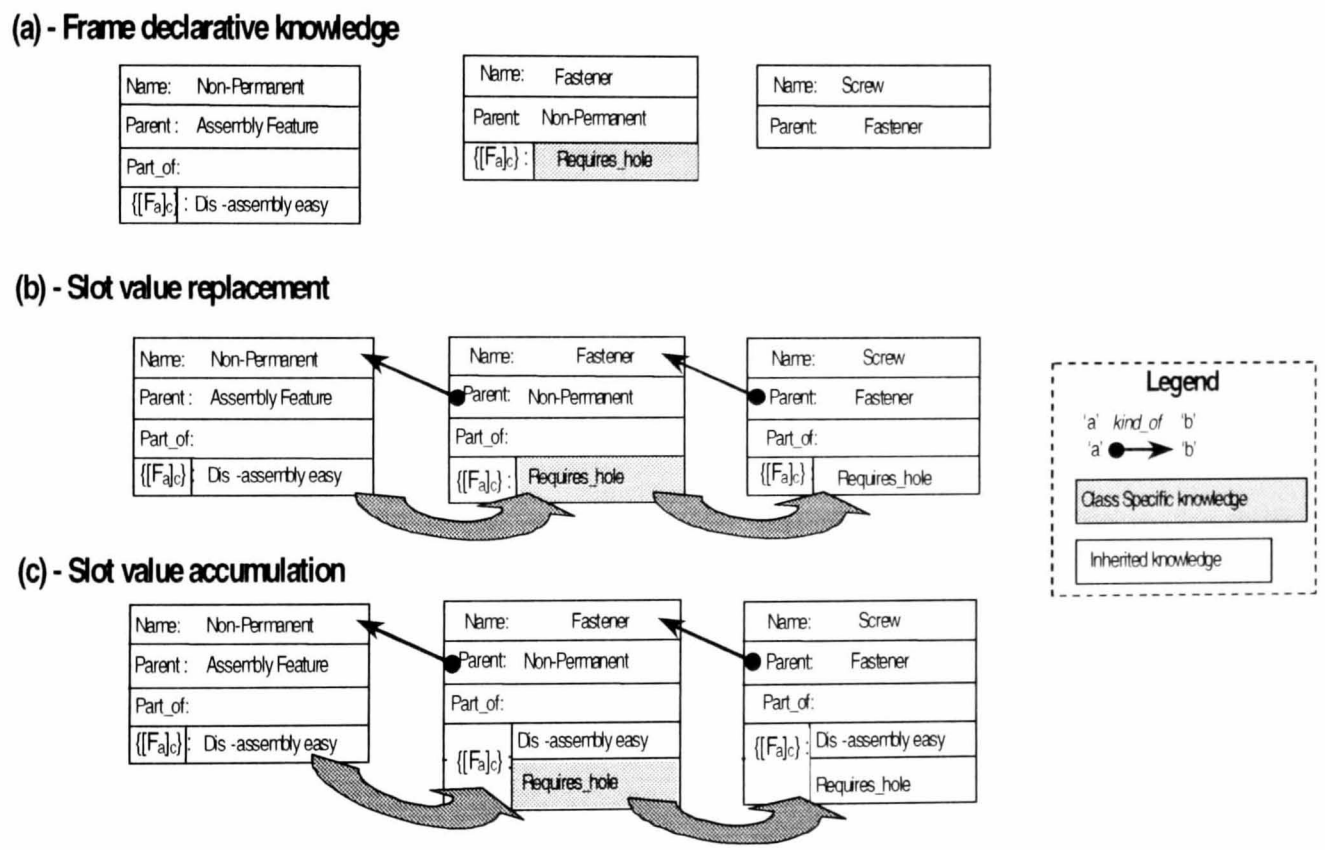


Figure 9.6 - Replacement versus cumlative slot value inheritance mechanism

Slot attribute inheritance

In *slot attribute inheritance,* the slot attribute itself is inherited. For example, consider three assembly features whose initial declarative knowledge is represented by the frames in Figure 9.6a. As reflected by Figure 9.6b, as *Fastener* is defined as a kind_of *Non-permanent*, it inherits the slot *'Part_of"* from its parent.

Slot value replacement

In *slot value replacement,* a slot can inherit a *default value* from a parent frame. For example, the slot concerning assembly characteristic $\{[F_a]_c\}$ values for a *Screw* fastener in Figure 9.6b, inherits the default value 'Requires_hole'. With this mechanism, if a class has a specific value declared for a slot, then

that specific value *replaces* the inherited value. e.g. the frame *Fastener* in Figure 9.6b has a specific $\{[F_a]_c\}$ value declared i.e. 'Requires_hole' which replaces the value 'Dis-assembly easy' inherited from the parent *Non-Permanent* frame.

Cumulative value inheritance

Cumulative value inheritance allows values inherited from a parent frame to be *appended* to values specifically declared to that frame. For example, the frame *Fastener* in Figure 9.6c has a specific $\{[F_a]_c\}$ value declared i.e. 'Requires_hole' which is appended to the slot value 'Dis-assembly easy' inherited from the parent *Non-Permanent* frame. Thus, through *cumulative value inheritance*, the sub-class *Screw* will inherit both values (Figure 9.6c).

Reasoning with Production rules

As explained in Appendix D, a production rule represents procedural knowledge with a *conditional part* and related *actional part*. Reasoning with production rules requires the *matching* of the *conditional part* of a rule stored in a knowledge base with a similar pattern in the contents of the system's *working memory*. If a rule is matched, its *actional part* is executed, thus adding or removing new items contained within the working memory. Such inference can be approached in two different ways [Walters et al. 1988; Popovic et al. 1994]:

- *forward chaining*: in this strategy, the initial state is defined by the data available (facts) at the beginning and the goal is determined;
- *backward chaining*: in this strategy, the goal is guessed and the search from the goal to the initial state, defined by the given data, carried out in order to justify the goal postulation.

Frame-based Production Rules

Frame-based reasoning

Production rules can also reason about the characteristic of a *frame* by referring to its *attribute* or slot *value,* this termed *frame-based reasoning* [Walters et al. 1988]. Consider for instance, that a frame about engineering materials contains a slot for the *material name attribute [M]* and a slot for storing a set of the *material property values {m}.* Then, a production rule can have its *conditional part* referring to the material's property value as in (9.1) :

(9.1)

IF (non-magnetic $\in$ {m}) **THEN** using a magnetic separation process in the disposal phase is difficult for components realized from material [M].

Thus with such a *frame-based production rule* a piece of procedural knowledge represented by one rule can be applicable to a number of objects (e.g. PDEs) represented by frames. Hence, rule 9.1 avoids the need of defining different rules e.g. (9.2) for *specific* non-magnetic materials such as ABS, wood and rubber.

(9.2) **IF** [M]=[ABS] **THEN** using a magnetic separation process in the disposal phase is difficult for material [ABS].

Hierarchical reasoning

The *conditional part* of a production rule can also refer to a *class* of frames (or a pattern of classes), this resulting in what is termed *hierarchical reasoning* [Walters et al. 1988]. An example is rule 9.3:

(9.3) **IF** object is a *kind_of* ⌊Fastener⌋ **THEN** add a [hole] to <model>$_{component}$

Least & specific commitments

As the *conditional part* of rule 9.3 refers to the *class* ⌊Fastener⌋, it is equally applicable to sub-classes of a taxonomy such as ⌊Screws⌋ and ⌊Pop-rivets⌋. Another advantage of representing knowledge as in rule 9.3 is that the same piece of knowledge is applicable to classes belonging to different abstraction levels in a *kind_of* taxonomy. For example rule 9.3 is equally applicable to a *bolt* and a more specific commitment *e.g. an M6 bolt instance.*

Representation in FORESEE - A hybrid scheme

A hybrid representation scheme

Many applications cannot have their knowledge easily represented by a single representation scheme, so a *hybrid representation* [Walters et al. 1988; McMahon et al. 1993] scheme is adopted. This allows associated knowledge to be divided into sections that can be represented and reasoned upon by different representation schemes. Given the LCC knowledge modelling and structuring formalism presented in Chapter 8 and the representation schemes commonly available with commercial AI development tools, it was decided to employ a hybrid representation scheme (see Figure 9.9) for codifiying a *synthesis elements library, LCC inference knowledge* and *LCC action knowledge* as described next.

9.2.3.1 Synthesis Elements Library Representation

Representation of synthesis element models

Models of different synthesis elements captured in a *Synthesis Elements Library (S.E.L)* are stored in FORESEE with a *frame* [Walters et al. 1988] representation, as these allow a chunk of declarative knowledge about an element to be conveniently structured and represented. Each frame representing a synthesis element will have a number of *slots* with *attributes*

and *values* particular to the element's formalism presented in Chapter 8. For example, from relationship (8.17), an assembly feature model (Figure 9.7) will, in addition to an attribute storing the element's name, have a slot attribute for storing a set of different assembly characteristics $\{[F_a]_c\}$.

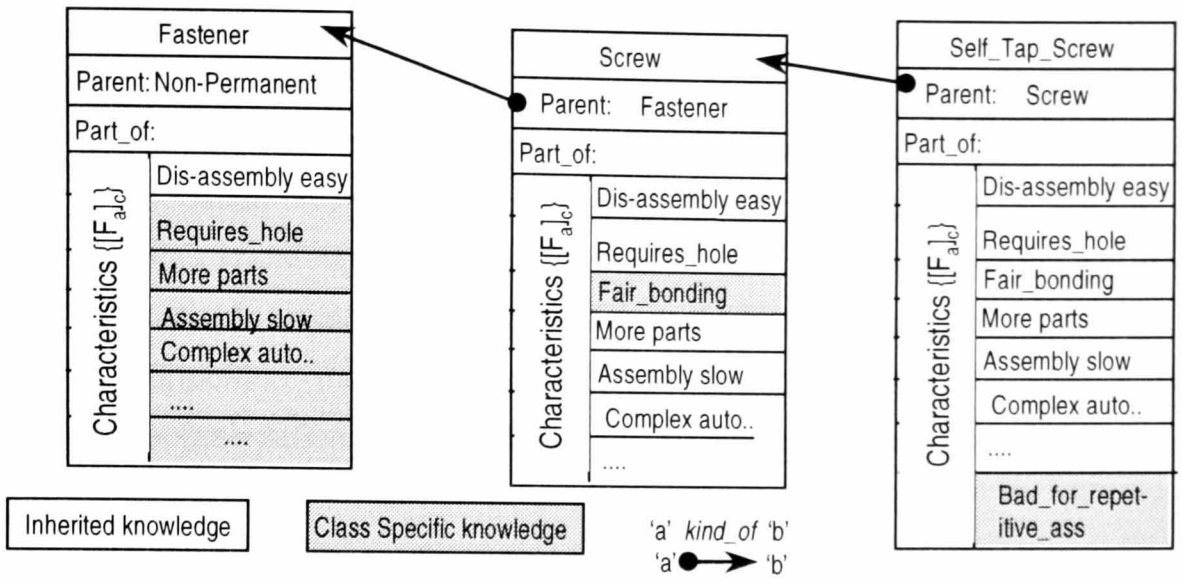


Figure 9.7 - Typical synthesis element, frame-based taxonomic representation

Representation of Kind_of relationships

By including a slot named *Parent* (Figure 9.7), frames can be related to other frames (as explained in Appendix D) to allow the elements to be organized in the S.E.L. into *kind_of* taxonomies, with an inferential structure as discussed in section 8.4. To facilitate this structure, FORESEE employs *slot attribute* and *slot value cumulative* inheritance mechanisms. A partial representation of the taxonomy of Figure 8.13 is illustrated in Figure 9.7 with the slot *'Parent'* storing the value of the frame's parent class.

Representation of Part_of relationships

Similarly *synthesis element* frames also have a *'part_of'* slot (Figure 9.7) whose value is specified during synthesis, when frame instances are created and related to other frames. In this way, related frames form a semantic network (see Appendix D) representing an evolving compositional model, based on the arefact life modelling frame (section 7.3.3) concept.

9.2.3.2 Representing LCC Inference Knowledge

LCC_{ni} knowledge representation scheme

This LCC knowledge derived from both public and private (e.g. company specific) sources, concerns synthesis elements captured in the S.E.L. As reflected by the formalism in (8.1) and the structuring of LCC_{ni} knowledge presented in Chapter 8, a LCC_{ni} can be associated with a class of elements. Thus, a fact describing a LCC_{ni} is stored in slots of frames representing synthesis elements. For example, the ⌊Fastener⌋ frame in Figure 9.7 has 'Requires_hole' stored as part of its declarative knowledge model. This is then inherited by the ⌊Screw⌋ and ⌊Self_Tap_Screw⌋ sub-class frames.

LCC_i knowledge representation scheme

Based on the formalism in (8.27), a LCC_i can be inferred once an interacting relationship (IR) is present in the evolving *artefact life model*, the latter described as a semantic network between frames. Thus, LCC_i knowledge is represented by a *production rule* scheme of the form:

(9.4)

IF (IR)$\subseteq$ <life model> **THEN** LCC_i

To reduce knowledge duplication, the interacting relationship (IR) should be modelled on the most general class as explained by (8.60). For this reason, LCC_i knowledge is represented in FORESEE with a *frame-based production rule* to exploit *frame-based reasoning* and *hierarchical reasoning* for supporting the structure disclosed in section 8.4. Rule (9.5) is a representation of example of (8.61), this expressed in system code as in Appendix J:

(9.5)

IF (([M] •→ ⌊Thermoplastic⌋) → <model>$_{componet}$)

AND (([F] •→ ⌊Protruding⌋) → <model>$_{componet}$)

AND (([P]•→⌊Solidification_based⌋) → <phase>$_{realization}$)

THEN (LCC_i = 'sink mark defect' → <model>$_{componet}$)

9.2.3.3 Representing LCC Action Knowledge

Based on the LCC knowledge modelling concepts presented in section 8.2.1.2, the representation of *LCC action knowledge* concerns the representation of *performance mapping knowledge*, *concurrent synthesis patterns* and *guidance/explanation knowledge* as discussed next.

Performance mapping knowledge representation

Performance mapping knowledge (8.5) is represented in FORESEE as a *performance measure function* (see Appendix G) which is executed when a LCC-specific production rule is triggered. The function stored in the *actional part* of a production rule will cause a specific relative change (e.g. +5) in a specific performance measure (e.g. time) of a specific process (e.g. assembly) or phase (e.g. realization). An example of such a representation is in (9.6) (expressed in system code in Appendix J):

(9.6)

IF (LCC_i = 'sink mark defect' → <model>$_{componet}$)

THEN ((Quality)<phase>$_{use}$ = -9)

AND ((Time)<phase>$_{realization}$ = +5)

AND ((Cost)<phase>$_{realization}$ = +4).

Concurrent synthesis pattern representation

Concurrent synthesis patterns are represented in FORESEE as *functions* in the *actional part* or a production rule. When a LCC-specific production rule is triggered, the functions cause the necessary manipulation (add/delete/modify) of a specific model forming part of the artefact life model. An example of this representation is given in (9.7)with typical system code as in Appendix J:

(9.7)

IF (LCC_{ni} ='hole required')

THEN ($[d]_E\{[F]\}$= [hole])→ $<model>_{componet}$.

Guidance/ Explanation representation

Guidance/explanation knowledge does not manipulate the facts stored in the working memory. The guidance/explanations are thus represented in hypermedia [Woodhead 1990] format as this provides a means of providing textual, graphical and animation based descriptions of a consequence and guidance to its avoidance. The explanations are displayed in a hypermedia based *consequence browser*. Through purposely defined *functions* the *actional part* of a production rule (see Appendix J for an example) triggers the relevant hypermedia based explanation/guidance once a LCC is detected.

9.2.4 Development Software & Hardware

Development software employed

FORESEE has been implemented in wxCLIPS which is a Windows environment of CLIPS [Giarratano et al. 1994]. The implementation employs COOL (CLIPS Object-Oriented Langauge) [CLIPS 1997]. wxCLIPS is readily and freely available on the internet, making it attractive from a cost and availability point of view. A major reason for selecting CLIPS is that it provides *multiple knowledge representation schemes* thus supporting the *hybrid representation scheme* introduced in section 9.2.3. In this respect, wxCLIPS provides a complete environment for the construction of rule and/or object based expert systems through three different programming paradigms:

- *object-oriented,* this essentially being a frame representation scheme;
- *rule-based* which allows user defined classes/instances to be pattern matched to support *frame-based* and *hierarchical reasoning;*
- *procedural* similar to languages such as C, Pascal and LISP, useful for specifying a number of system *functions*.

Hardware platform employed

wxCLIPS runs on different hardware platforms including PCs and Macintosh. FORESEE has been implemented on the PC platform, using a Pentium 100MHz computer with 32 Mbytes RAM and 1Gbyte hard disk running the Windows95 operating system.

9.3 FORESEE Architecture

FORESEE architecture

Based on the identified system requirements (outlined in Table 9.1) and the implementation issues discussed in 9.2, the architecture of FORESEE is as shown in Figure 9.8. It consists of a *Knowledge based system* integrated with a number of components: *Library access module, Knowledge manager, Solution model manipulator, Solution model viewer, Consequence browser, Multi-X behaviour module, Session history module* and a *User interface* that will be described in this section. Appendix E lists implementation filenames.

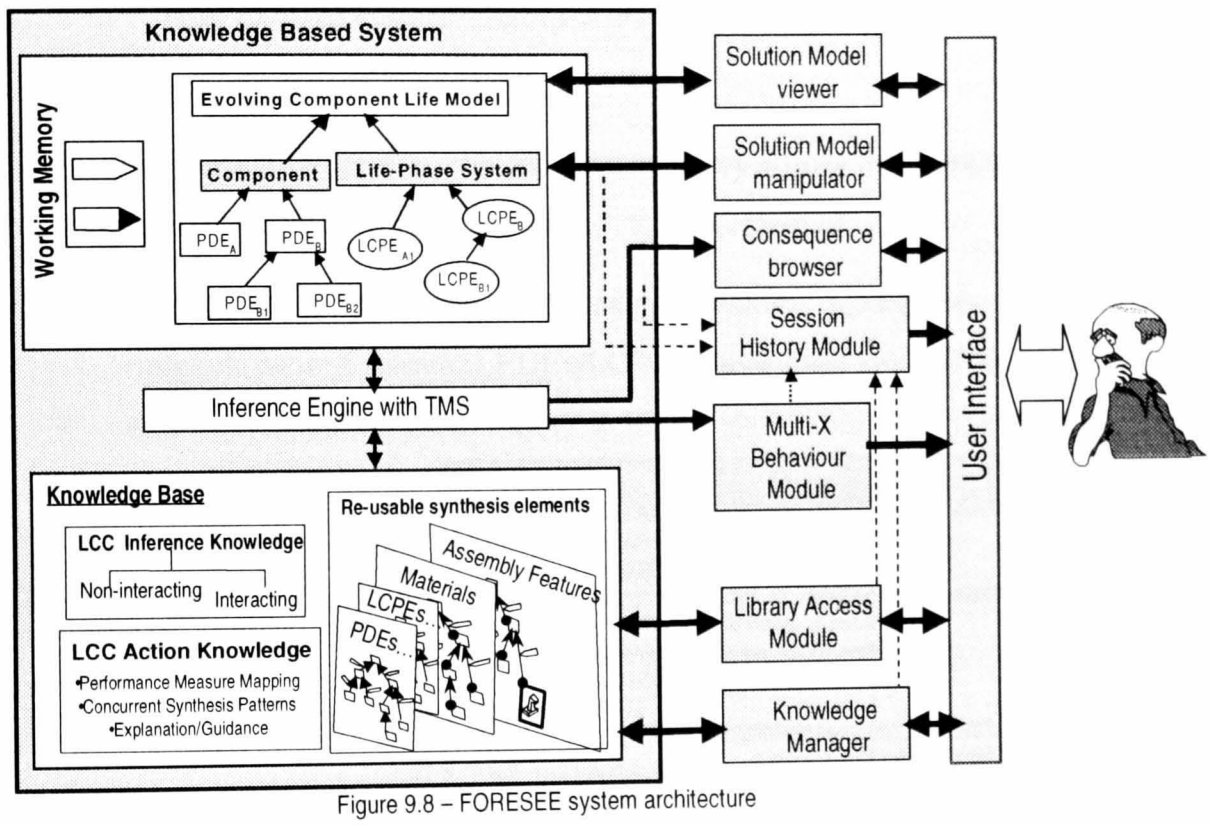


Figure 9.8 – FORESEE system architecture

9.3.1 Knowledge Based System

The *knowledge-based system* (Figure 9.8) employed in FORESEE comprises a relevant *Knowledge base*, a *Working memory* handling 'current working knowledge' and an *Inference engine*, these described next.

The knowledge base

Contents of the knowledge base

The knowledge base represents a *LCC knowledge model* and thus consists of a *synthesis elements library* and related captured *LCC knowledge*, which following the arguments made in section 9.2.3, are codified in computational form as in Figure 9.9.

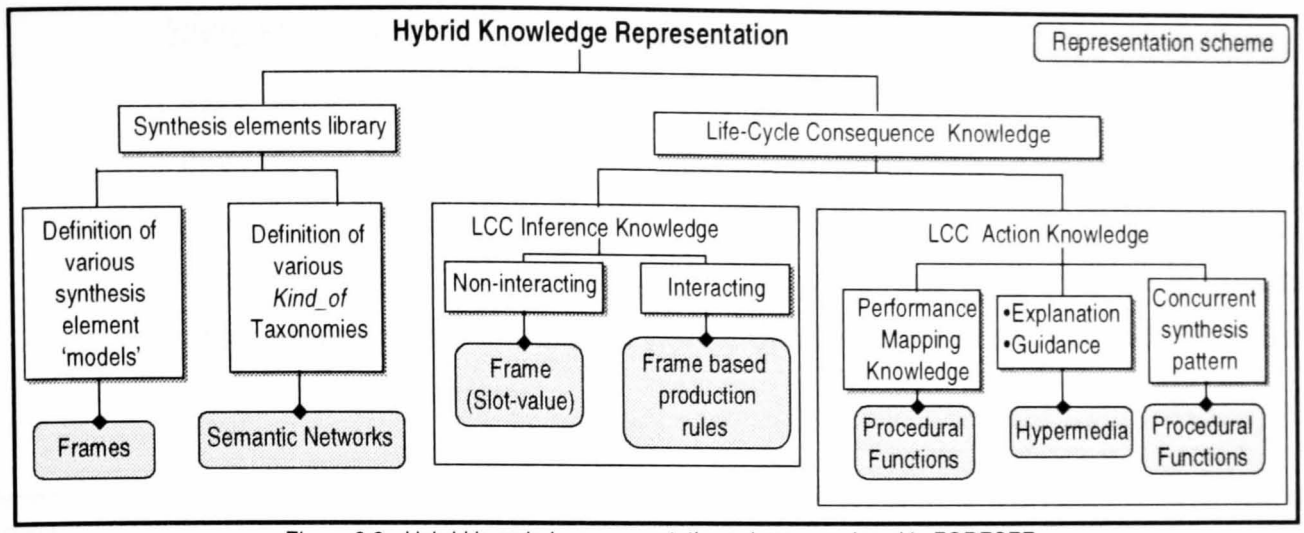


Figure 9.9 - Hybrid knowledge representation scheme employed in FORESEE

Working Memory

During system execution, the *working memory* stores the *concurrent synthesis current working knowledge* (section 7.1) consisting of:

Artefact life composition model representation

- *an evolving artefact life compositional model,* represented as a semantic network, where relevant PDEs/LCPEs have their *part_of* slot defined. An example is illustrated in Figure 9.10.

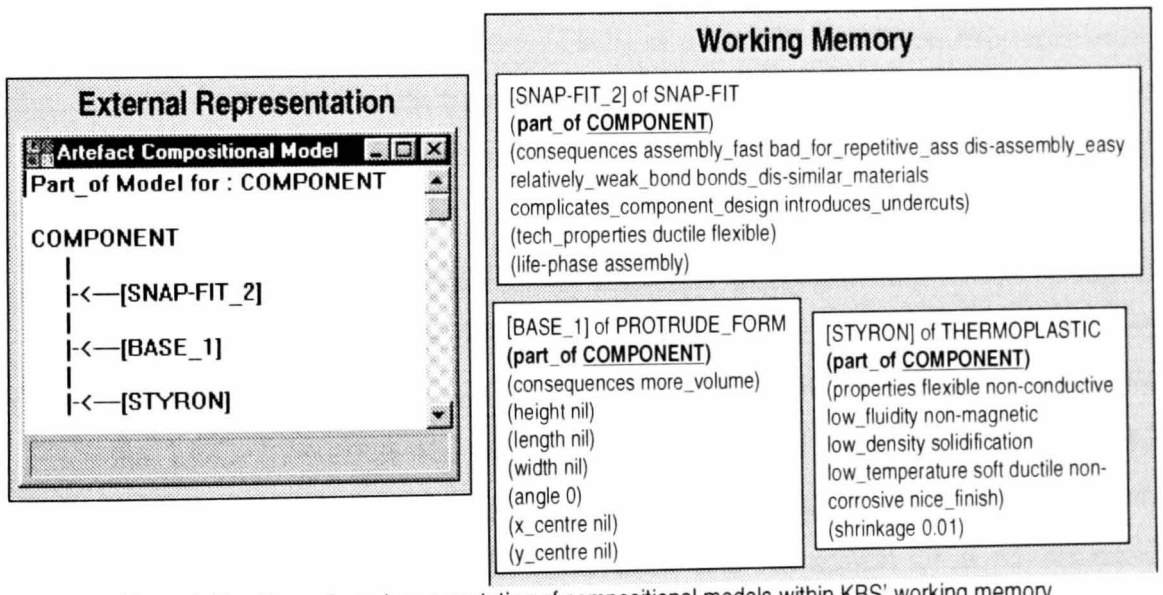


Figure 9.10 – Frame-based representation of compositional models within KBS' working memory

Representation of current LCCs

- *the current LCCs,* as a set of facts. Figure 9.11 shows an example.

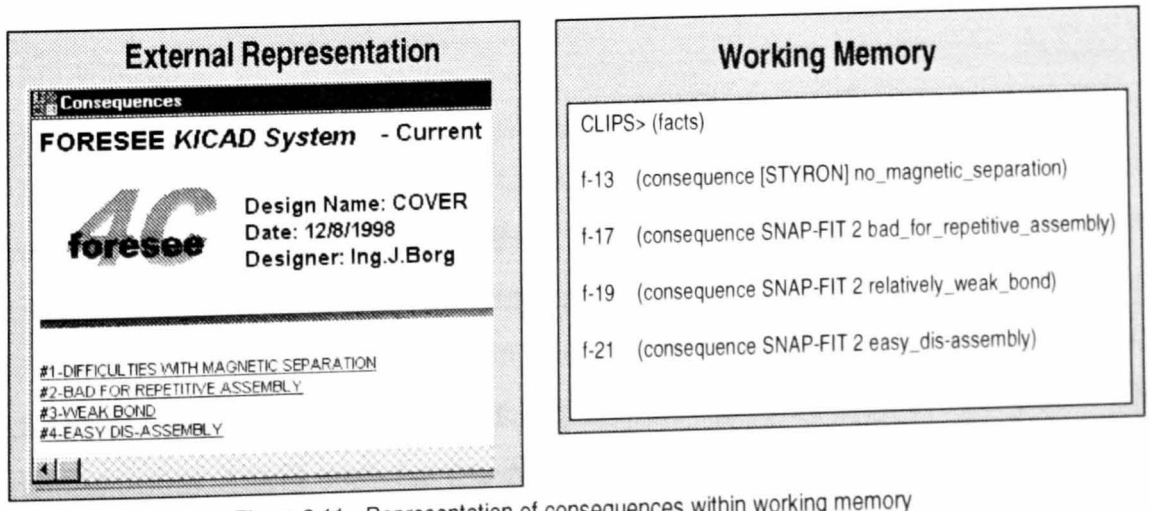


Figure 9.11 - Representation of consequences within working memory

Inference Engine

This is a domain knowledge-independent software program containing instructions to reason with the information contained in the knowledge base [Popovic et al. 1994]. In the case of FORESEE it can select and interpret the LCC inference knowledge stored in the knowledge base to draw conclusions or to generate new knowledge (facts) out of *concurrent synthesis CWK* stored in the *working memory*.

Inference strategy

Due to the need to *foresee* LCCs associated with the evolving solution, the inference method employed in FORESEE is *forward chaining*. With this inference, the state of the evolving solution is used to infer LCCs and take necessary actions through LCC action knowledge.

Truth maintenance system

To cope with both an evolving and a *changing* artefact life model stored within the *working memory,* FORESEE employs *truth maintenance* [Popovic et al. 1994] to maintain LCC inferred, performance measures, the LCC list in the browser and causal dependencies between component and life-phase models. A *truth maintenance system (TMS)* is a knowledge base management system that is activated each time the reasoning system generates a new truth-value. It keeps track of interrelations between assertions and conclusions drawn. It is not within the scope of this research to discuss how different TMSs operate. Such information is detailed in [Walters et al. 1988]. As LCC_{ni} inference knowledge is represented as a slot within a frame describing a synthesis element, then retracting that element automatically retracts the related LCC_{ni}. As LCC_i inference knowledge is represented with frame based production rules, the maintenance of LCC_i is more complex as a non-valid *interacting relationship* has to be detected *after* the retraction of a synthesis decision commitment. To cope with this situation, *logical dependency* provided by wxCLIPS has been employed as explained in Appendix H. However, logical dependency is not suitable for undoing actions performed by LCC action knowledge. For this purpose, a mechanism was specifically developed, based on the principle of segmenting LCC knowledge into three parts – a rule for detecting the existence of a LCC, a rule performing necessary actions and a rule undoing actions due to the retraction of a LCC. This mechanism is explained in Appendix H with examples given in Appendix J. Similarly, to maintain a valid list of LCC in the hypermedia based browser, a special function (see Appendix I) was developed to dynamically generate HTML [Bacon et al. 1997] code for the current LCCs stored in the working memory.

9.3.2 User Interface

To address the KICAD tool requirements, FORESEE's user interface provides a number of *pull-down menus* by which information is *acquired* from the designer. To *present* information to the designer, the user-interface employs a number of *windows* for different purposes as indicated in Figure 9.12.

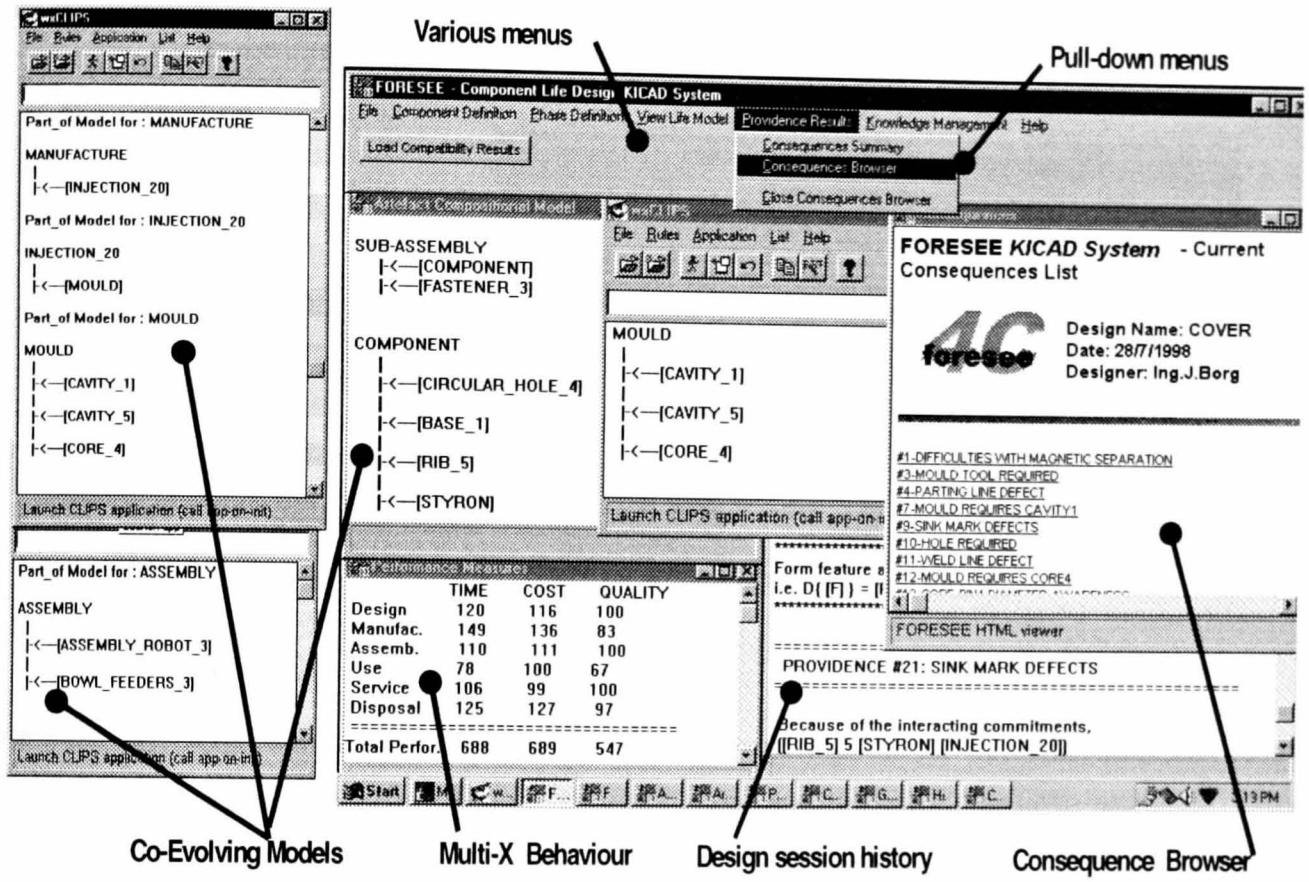


Figure 9.12 - FORESEE's user interface

9.3.3 Library Access Module

This module provides functions that allow a designer to qualitatively query the *synthesis elements library* for elements that result in intended consequences e.g. search for assembly features that make dis-assembly easy (Figure 9.13).

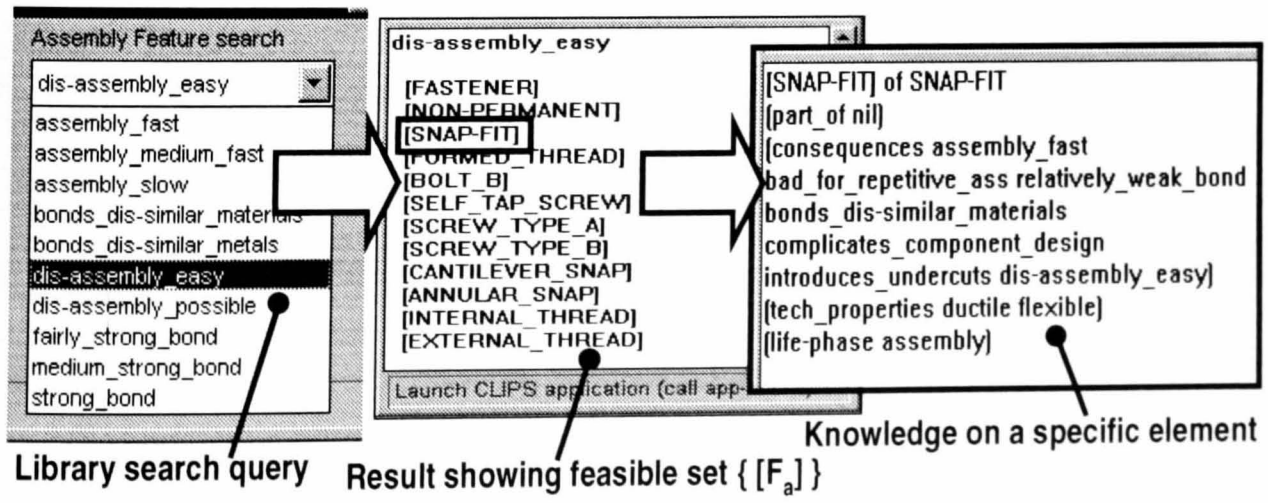


Figure 9.13 - Library access module

9.3.4 Solution Model Manipulator

This provides a set of functions, which allows a designer to *add* or *remove* synthesis elements from the models being concurrently synthesized. The manipulator allows elements to be added to the evolving model at a least or specific commitment level. Through pull-down menus (Figure 9.14), the manipulator also allows *detailing* and *modifications* to elements already committed, such as changing a feature parameter value.

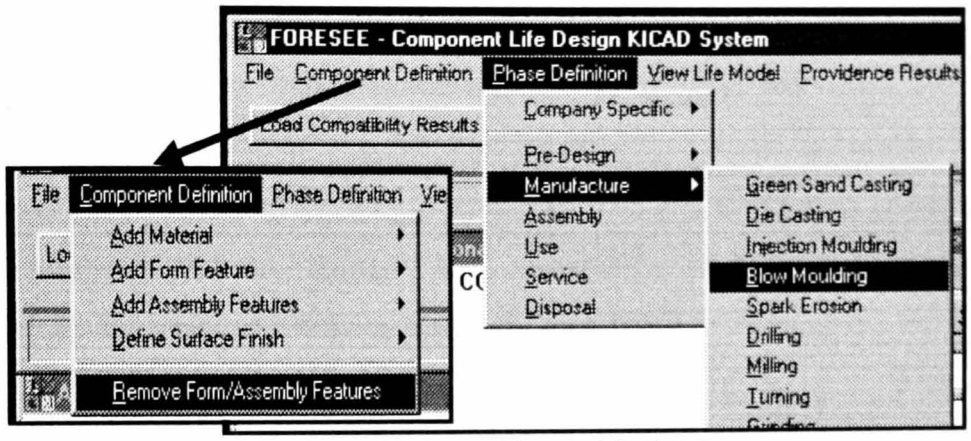


Figure 9.14 – Solution model manipulation functions

9.3.5 Solution Model Viewer

Externalizes evolving solution

This system architecture module is responsible for externalizing the various, evolving compositional models stored within the working memory (Figure 9.10) as a textual *part_of* hierarchy displayed in different windows of the user interface (Figure 9.12).

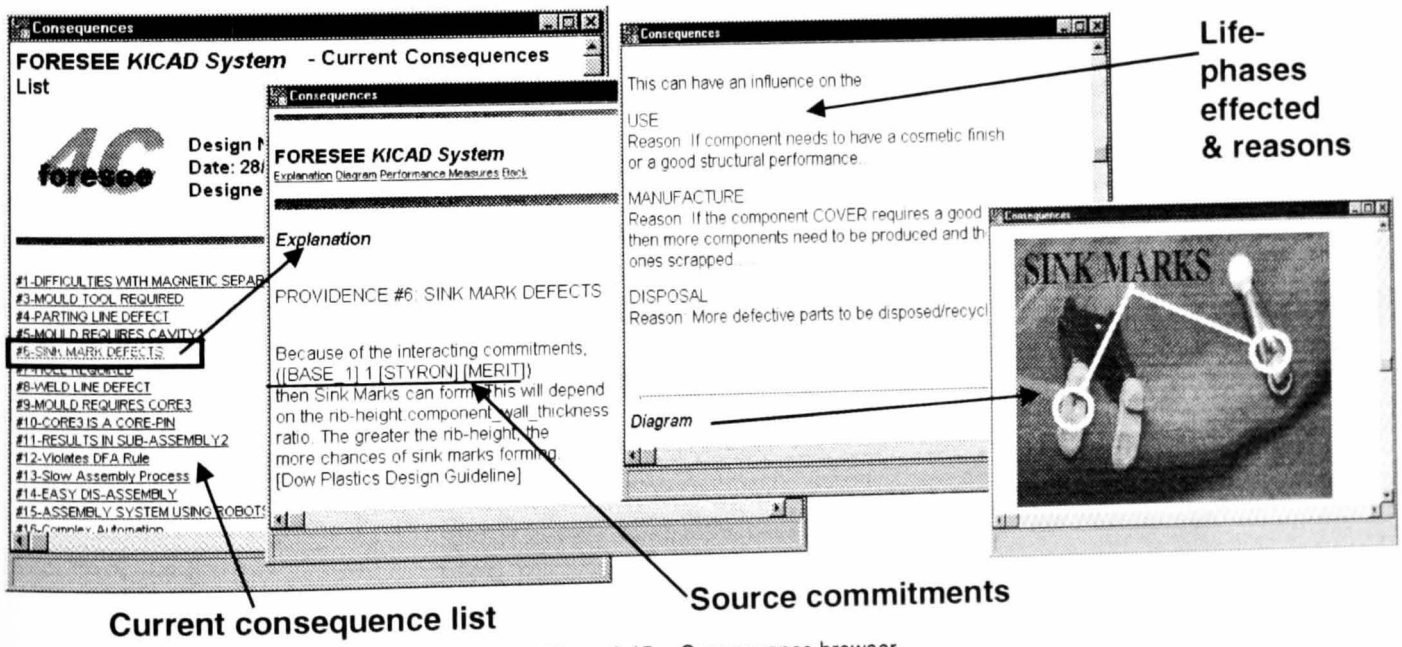


Figure 9.15 - Consequence browser

9.3.6 Consequence Browser

Hypermedia based

This is a hypermedia-based browser (Figure 9.15) used for externalizing the LCC facts stored within the *working memory* (Figure 9.11). The browser has

been implemented in HTML, the latter also supported by wxCLIPS. It allows a specific LCC to be browsed, this providing the designer with explanations of a consequence, its source commitment(s) and guidance to its avoidance. As reflected by Figure 9.8, the TMS ensures that LCCs abandoned due to decision retractions are not displayed in the browser.

9.3.7 Multi-X Behaviour Module

Displays performance measures

This module utilizes the values of the performance measures estimated through *performance mapping knowledge* to display the state of relative performance measures for the different artefact life-phases in matrix form. To display a finer resolution of fluctuations in the *Multi-X behaviour* window, it has been implemented to display the realization phase decomposed into manufacturing and assembly system (Figure 9.12).

9.3.8 Knowledge Manager

This module permits knowledge in the knowledge base to be managed by providing utilities (Figure 9.16) that allow:

Implemented knowledge management utilities

- knowledge concerning synthesis elements to be *viewed*;
- new classes to be *added* to a taxonomy during execution e.g. add a sub-class ⌊CNC_Milling⌋ of the class ⌊Milling⌋ processes;
- new synthesis elements to be *added* during execution – e.g. a company specific 'CNC_milling_01' can be defined;
- knowledge of synthesis elements to be *modified* during execution - e.g. change the *economic quantity* value for 'CNC_milling_01';
- (some) synthesis elements to be *accessed* from a particular perspective (e.g. particular bought-out item supplier).

As implemented, only LCC_{ni} can be added/modified *during* execution.

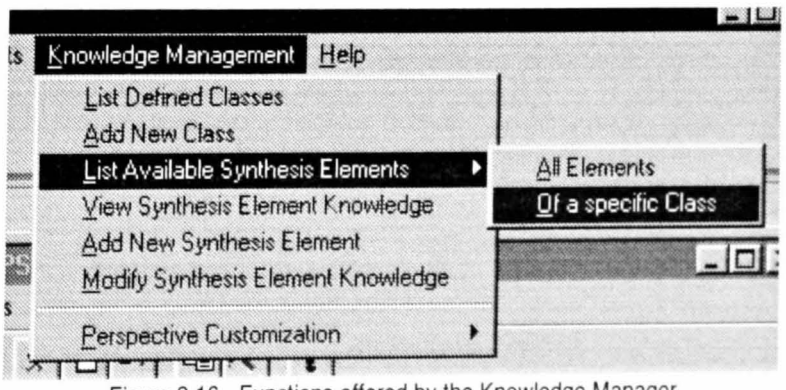


Figure 9.16 - Functions offered by the Knowledge Manager

9.3.9 Session History Module

Automates design session recording

This module provides functions that automatically and sequentially record the different commitments/retractions made, the resulting *multi-x behaviour* at each state, new classes/instances created etc. The designer can browse and/or print the recorded design session history.

9.4 Chapter Conclusions

Implementation issues

This chapter, the last one in Part 'B' of this dissertation has presented how the 'KC' approach framework to $D_sF\Sigma X$, can be realized into computational form, as a KICAD prototype system named FORESEE. For this purpose, section 9.1 presented a set of requirements (summarized in Table 9.1) for such a KICAD tool realizing the 'KC' approach framework. As discussed in section 9.2, to codify the LCC knowledge model into computable form, a *hybrid knowledge* representation scheme (summarized in Figure 9.9) has been employed. Due to this hybrid scheme, the development software employed was wxCLIPS whilst the hardware used for realizing the FORESEE prototype was a Pentium 100MHz PC. The domain selected for prototype implementation is that of thermoplastic components.

FORESEE system architecture

Section 9.3 presented the architecture of the FORESEE prototype, this derived from the identified system requirements and the implementation issues decided. FORESEE is composed of a *Knowledge based system* integrated with a number of components: *Library access module, Knowledge manager, Solution model manipulator, Solution model viewer, Consequence browser, Multi-X behaviour module, Session history module* and a *User interface.*

Utility of FORESEE to $D_sF\Sigma X$

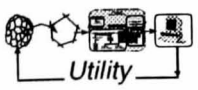
The next chapter, which commences Part 'C' of this dissertation, presents a scenario of employing the prototype FORESEE for thermoplastic component design, this providing a basis for evaluating the 'KC' approach to $D_sF\Sigma X$.

Evaluation, Discussion & Conclusions

chapter

10.0 Evaluation of FORESEE To Supporting $D_sF\Sigma X$

Scope



This chapter presents an evaluation of utilizing FORESEE to support a component conceptual design scenario as basis for discussing the effectiveness of the 'KC' approach to $D_sF\Sigma X$. Section 10.1 presents issues related to the evaluation approach adopted. Section 10.2 than presents a component $D_sF\Sigma X$ scenario via FORESEE. Evaluation results and a critical discussion on their interpretation are disclosed in section 10.3. Chapter conclusions are made in section 10.4.

10.1 Evaluation Approach

Logical arguments have been made throughout the dissertation to demonstrate that there is a connection between the initial research state (the real problems, existing theories and models) the hypothesis and the established research solution.

However, the main approach considered suitable for critically evaluating the result with respect to a number of established criteria is that of experimentation with the FORESEE prototype (this a realization of the 'KC' approach framework), in its effectiveness to supporting component $D_sF\Sigma X$. The evaluation has been conducted within the thermoplastic domain.

Evaluation Criteria

Evaluation perspective

An evaluation related to this research can be performed from mainly two different viewpoints: the effectiveness of the support to $D_sF\Sigma X$ as considered by designers, *or,* the KICAD system's performance (e.g. speed of retrieving relevant knowledge). Although both are relevant to research of this type, the hypothesis of this Ph.D. (Chapter 1) together with the outcome of the critical review of existing providence means (Chapter 4) indicate that the evaluation exercise via FORESEE should mainly *focus* on the designer's viewpoint.

Criteria

Thus, a number of key evaluation criteria were identified, these being:

i. Are designers supported in becoming aware of unintended LCCs co-evolving *during* component solution synthesis?

ii. Are designers *assisted* in becoming aware of *multiple* and *interacting* LCCs?

iii. Are designers made aware of component life-specific or generic LCCs?

iv. Are designers motivated and supported in *exploring* alternative decision commitments?

v. Are designers guided to make decision commitments consciously with respect to artefact life issues?

FORESEE based component $D_sF\Sigma X$ scenario

To assess the KICAD based 'KC' approach to component $D_sF\Sigma X$, a number of formal demonstrations via FORESEE were therefore carried out in the U.K. with a total of 23 participants, purposely having a different background:

Evaluators' background

- *7 consulting and practicing designers* from Scottish Design, Forth Product Design, Integrated Technologies, Digital Equipment Corporation (Ayr) and Glasgow Development Agency. These had practical experience of DFX, how early design is executed in industry and of the various problems encountered during the development activities of various types of commercial mechanical artefacts, including thermoplastic components;

- *9 academic researchers* from the CAD Centre, University of Strathclyde, Glasgow. These had a thorough understanding of design activities and were familiar with various *early design* and *CAD* research issues;

- *7 Computer Aided Engineering Design M.Sc. students* from the University of Strathclyde, Glasgow, who had formal training in design process activities and current research issues concerning *design* and *CAD*.

Evaluation Procedure

Explanation to evaluators

Individuals who participated in the evaluation were handed introductory information (Appendix K) concerning the objective of the demonstration, reasons why participants would not be using FORESEE itself and issues they would be asked to comment upon after the demonstration. Following

sufficient time for the participants to read this background information, they were given explanations and definitions to certain terms (e.g. LCCs) that were to be encountered during the design scenario involving FORESEE.

Scenario demonstration

The evaluators were then given a demonstration of the utility of FORESEE concerning the early design of a thermoplastic component (see section 10.2). During the demonstration, evaluators were encouraged to ask questions to help them clarify any issue.

Structured interview

To avoid evaluators from influencing each other, at the end of the demonstration, evaluators were *individually* taken through a structured interview. This involved the demonstrator asking and, where necessary, explaining a set of pre-defined questions based on the identified evaluation criteria (see Appendix L) aimed at developing a coherent interpretation of the evaluators' assessment concerning FORESEE's *strengths* and *limitations* of the support it provides to $D_sF\Sigma X$. Comments made by the evaluators were recorded on paper.

Evaluation Difficulties

Degree of proof

The objective of the $D_sF\Sigma X$ scenario demonstration exercise is not to prove the solution, as this is difficult to achieve since:

i. comparing the respective influence on multiple life-phases of alternative component design solutions (one generated *via* FORESEE in a real design office environment and another *without*) during their 'real life' involves a substantial period of time. Whilst acknowledging that this would provide a more realistic basis for evaluating the effectiveness of FORESEE to supporting $D_sF\Sigma X$, embarking on this empirical exercise is difficult if not impossible to achieve within the timeframe for Ph.D. research;

ii. only a limited amount of knowledge, was codified in FORESEE. Although this enables evaluators to indicate the strengths and limitations of the 'KC' approach to $D_sF\Sigma X$, it is not extensive enough to support an exhaustive exploration activity;

iii. as stated in Chapter 9, FORESEE lacks some functionality, that can have an influence on the effectiveness of the 'KC' approach to $D_sF\Sigma X$;

iv. the range of evaluators was not extensive, both in terms of the number of people involved and also their technical background.

These difficulties correlate with arguments that in the *'AI in design'* research field, there are problems in carrying out effective evaluation exercises [Duffy et al. 1998]. Hence, with awareness of these difficulties, the purpose of demonstrating the application of FORESEE to a component $D_sF\Sigma X$ scenario, was to reveal indications of its *strengths* and *limitations* in supporting DsFΣX.

10.2 A Component $D_sF\Sigma X$ Scenario

Scenario background

Although a specific scenario is being presented, the concept of the 'KC' approach to $D_sF\Sigma X$ presented in this thesis is relevant to other domains, where concurrent synthesis can be performed with the reuse of *life synthesis elements* and where related *LCC knowledge* could be captured and reused. The scenario is based on the integration of a number of real artefact LCCs collected from industry and literature. It concerns the conceptual life design of a component intended to act as a cover for electronic circuitry housed in a separate enclosure (Figure 10.1).

Early design requirements

The requirements, mostly qualitative, known at this early design stage are that the cover needs to be non-corrosive, light-weight, easy to dis-assemble to allow servicing of the electronic circuitry, to be used in normal ambient temperatures and that a minimum of 9000 covers are required.

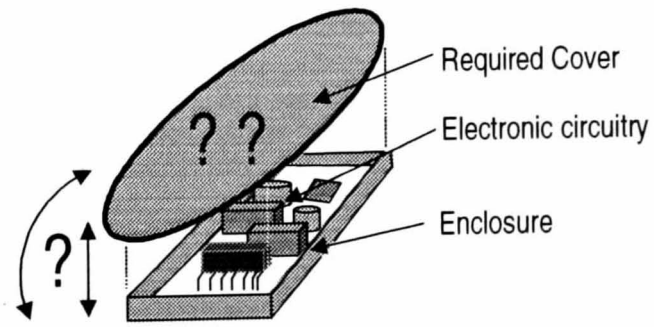


Figure 10.1 - Scenario component - a cover

Scenario description

To help the reader, *part_of* (→) relationships being established through decision commitments, are shown in scenario diagrams with an associated alphabetical symbol e.g. *(a)* in Figure 10.3, that will be referred to in the scenario text. Also, the scenario diagrams employ a mixture *of FORESEE screen dumps* (portraying the external view) and *PDE/LCPE frame models / taxonomies* portraying an internal representation of the *evolving solution / synthesis elements library* respectively (see Figure 10.2).

Scenario Initialization

FORESEE default knowledge status

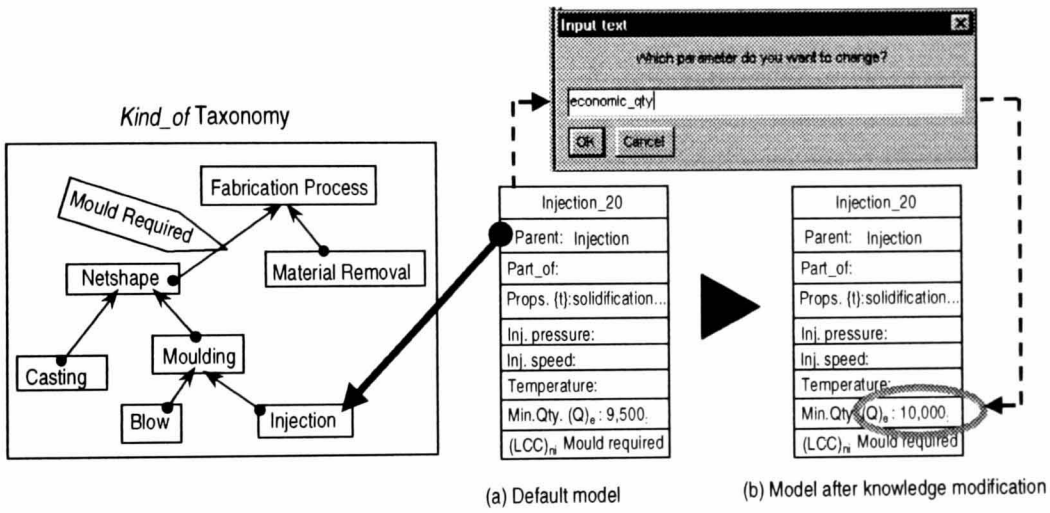
When FORESEE is initialized, it contains a number of default synthesis elements organized into default taxonomies (see Appendix F). Also, the default value of all performance measures is 100 e.g. (Time)<phase>$_{use}$=100.

Example of adding a new synthesis element

Company and domain specific *synthesis elements* can be added to the S.E.L in FORESEE. As an example, assume that the firm involved in this scenario has acquired a new injection moulding machine *'Injection_20'*. Through FORESEE's *knowledge manager*, the user adds this new LCPE as an instance of ⌊Injection⌋ in the fabrication taxonomy. Through inheritance, a model of this new LCPE is generated (Figure 10.2a).

Example of modifying an element's knowledge

As this new LCPE has a company specific minimum economic quantity $(Q)_e$, the default slot value inherited i.e. 9,500 is modified by the designer via *the knowledge manager* to 10,000 (Figure 10.2b).



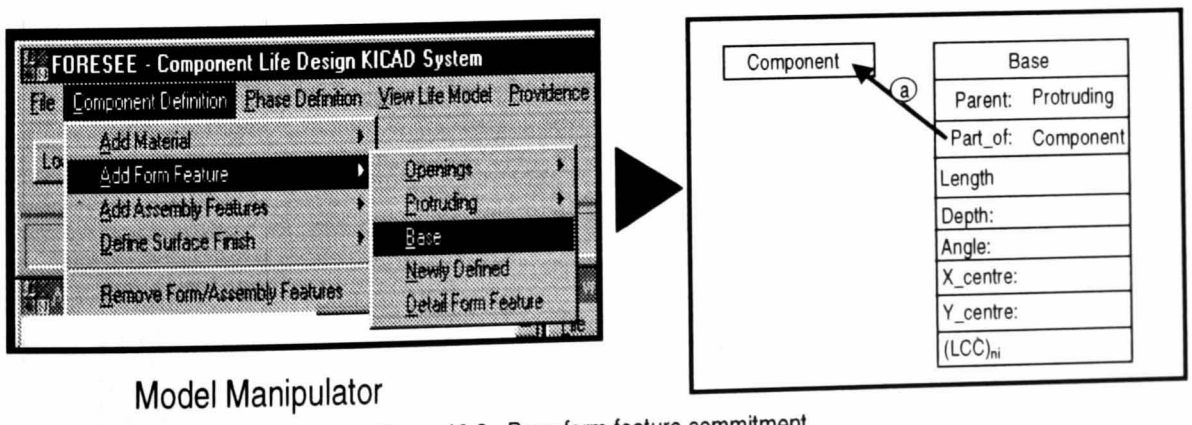
(a) Default model (b) Model after knowledge modification

Figure 10.2 - Adding a new synthesis element and modifying its knowledge content

Component Design

Specification of required [Q]

At the beginning of the design session, FORESEE *recommends* that the user inputs the quantity [Q] required in the *use phase*, in this case [Q]=9,000, in order for FORESEE to be able to reveal solution and problem specific LCCs.



Model Manipulator

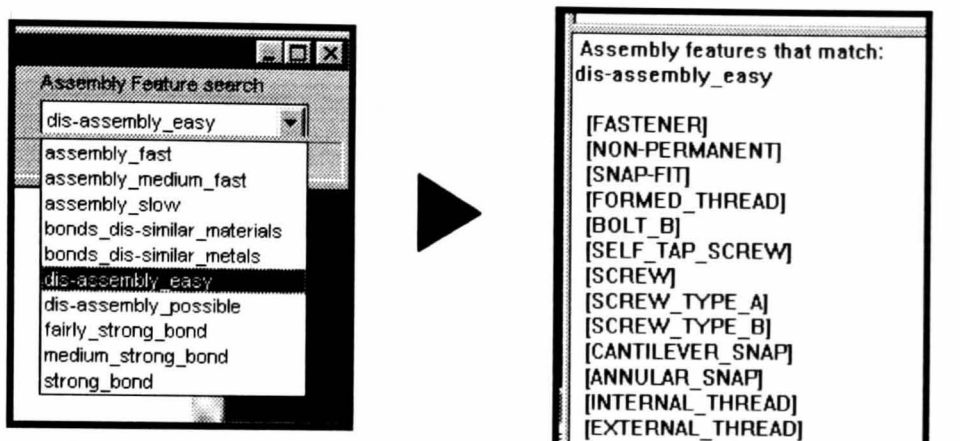
Figure 10.3 - Base form feature commitment

Component form synthesis

By using FORESEE's *solution model manipulator*, the designer commits a *[base]* form feature (Figure 10.3), to become *part_of (a)* the evolving component model. At this early stage, the designer makes a *minimum commitment* and thus the [base] parameter values are not specified.

Intended consequence – easy dis-assembly

To address the servicing requirements, the designer uses FORESEE's *library access module* to qualitatively search for PDEs that *intentionally* result in a non-permanent bond *i.e. dis-assembly_easy* (Figure 10.4a). This gives rise to a decision proposal $D\{[F_a]\}$ with a set of feasible options (Figure 10.4b).



(a) Search query

(b) Result showing feasible options

Figure 10.4 – Qualitative search for feasible assembly features

Artefact concurrent synthesis

From the set revealed, the designer selects and commits a [screw] to become *part_of (b)* the evolving solution (Figure 10.5). The [screw] which is a *kind_of* ⌊Fastener⌋ (see the taxonomy in Figure 8.13), co-evolves a set of LCC_{ni} given from (8.54), these displayed to the user as in Figure 10.6a. The LCC_{ni} 'more parts', which 'violates a DFA guideline', results in the component becoming *part_of (c)* a *sub-assembly* (Figure 10.5).

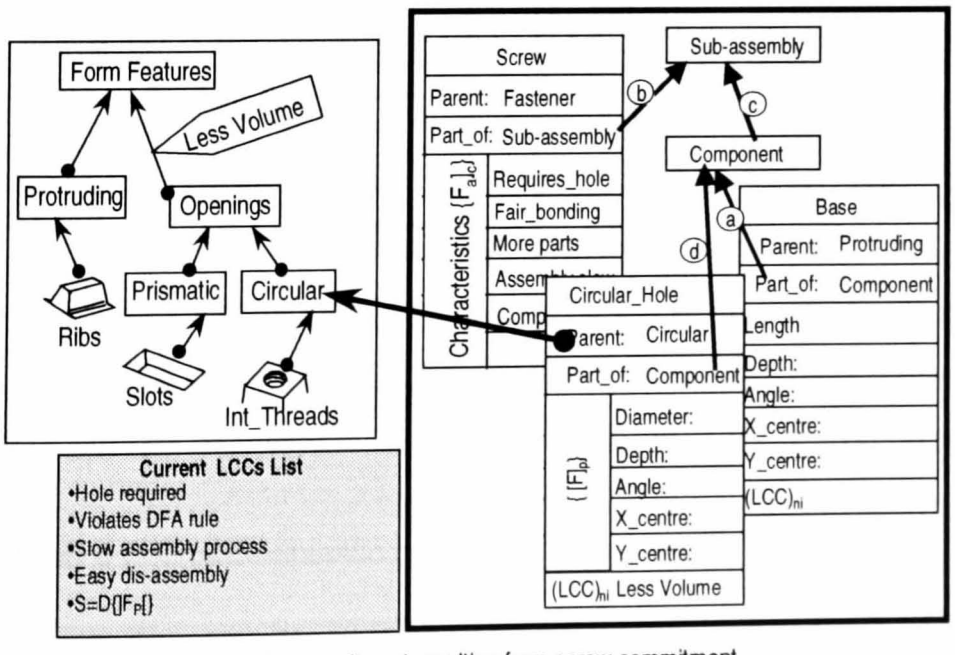


Figure 10.5 - Hole commitment resulting from screw commitment

Performance monitoring

Through *performance mapping knowledge*, these LCC_{ni} influence the behaviour of different life-phases e.g. from 8.7, 8.8 and 8.9, 'assembly slow' influences the *assembly process* in the 'realization phase' and the *service process* in the 'use phase' (these shown decomposed in Figure 10.6b).

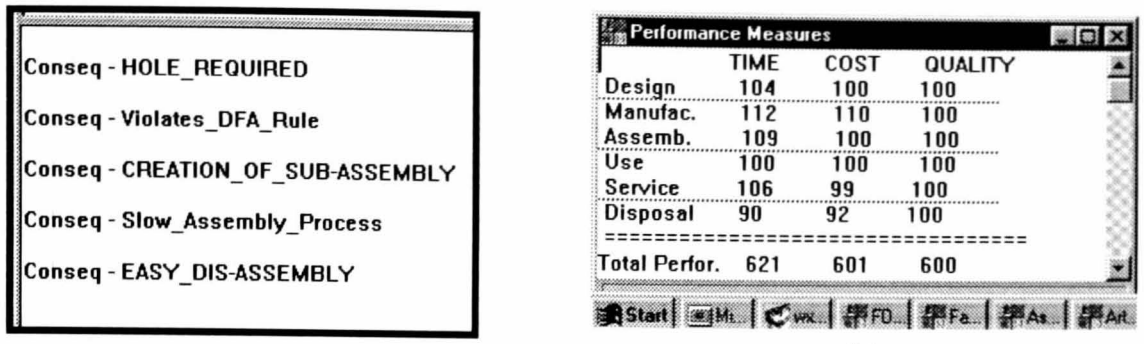


Figure 10.6 – Screw commitment LCCs and resulting performance measures

Component concurrent synthesis

Due to the LCC_{ni} 'hole_required' introduced by the [screw], based on the *concurrent synthesis pattern* in (8.12), FORESEE adds a [circular_hole] to become *part_of (d)* the component model (Figure 10.5). The [circular_hole] gives rise to a decision space 'S' concerning hole parameters as formally given by (8.28). At this stage, the designer prefers to make a minimum commitment and thus proceeds without specifying the hole parameter values.

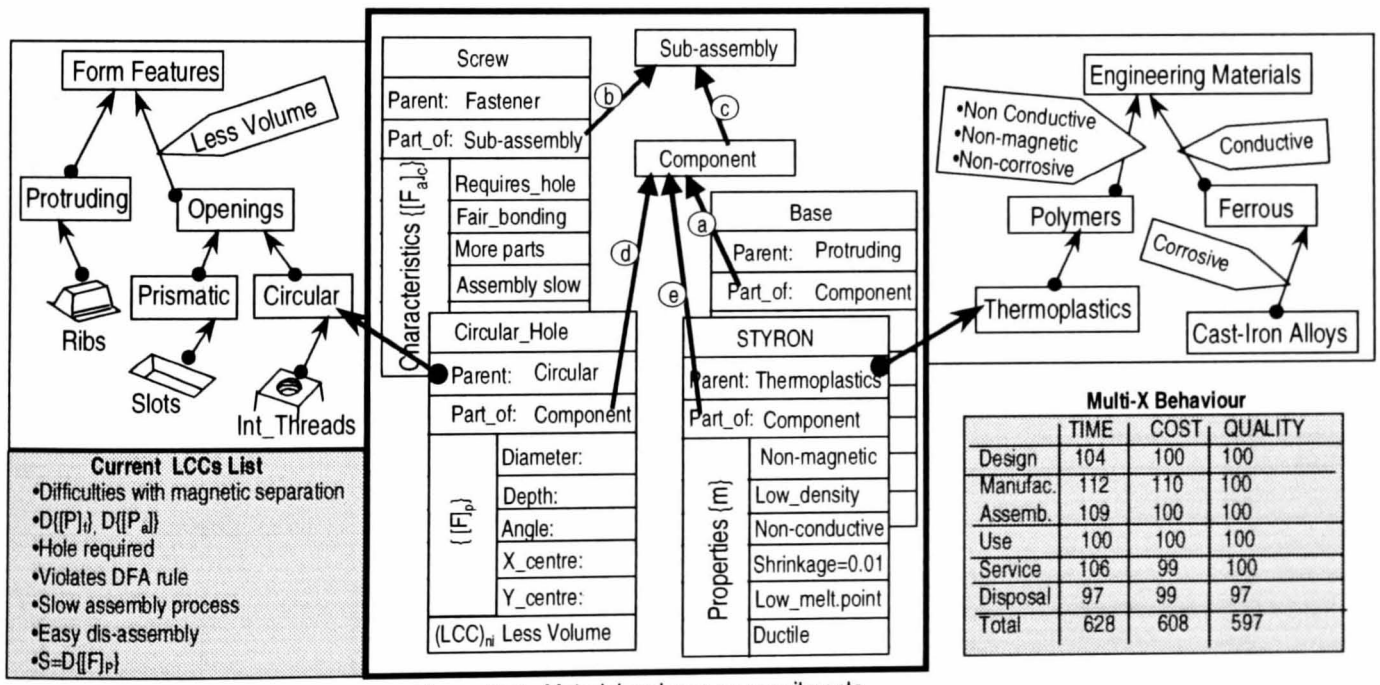


Figure 10.7 - Material and screw commitments

Search for suitable material

To address the given requirements, the designer uses FORESEE's *library access module* to qualitatively search for materials that have a *low_density* and are *non-corrosive.* This search results in a decision proposal:

(10.1)

$$D\{[M] = \{\lfloor \text{Thermosplastic} \rfloor \vee [\text{STYRON}] \vee [\text{ABS}] \vee [\text{Bakalite}] \vee [\text{Aluminium}]\}.$$

Material's disposal phase consequence

In this case, the designer explores committing STYRON (*part_of (e)* the component model - Figure 10.7). This material PDE introduces a set of properties {m} inherited from ⌊Thermoplastic⌋ including the *unintended* property 'Non-magnetic'. With the component still incompletely and imprecisely defined, FORESEE works independently from the designer and through *LCC inference knowledge*, reveals difficulties in the component's disposal phase – specifically separating the component with magnetic separation is not possible. This is explained via the LCC consequence browser (Figure 10.8). Relevant *performance mapping knowledge* updates *the multi-X behaviour* due to this LCC.

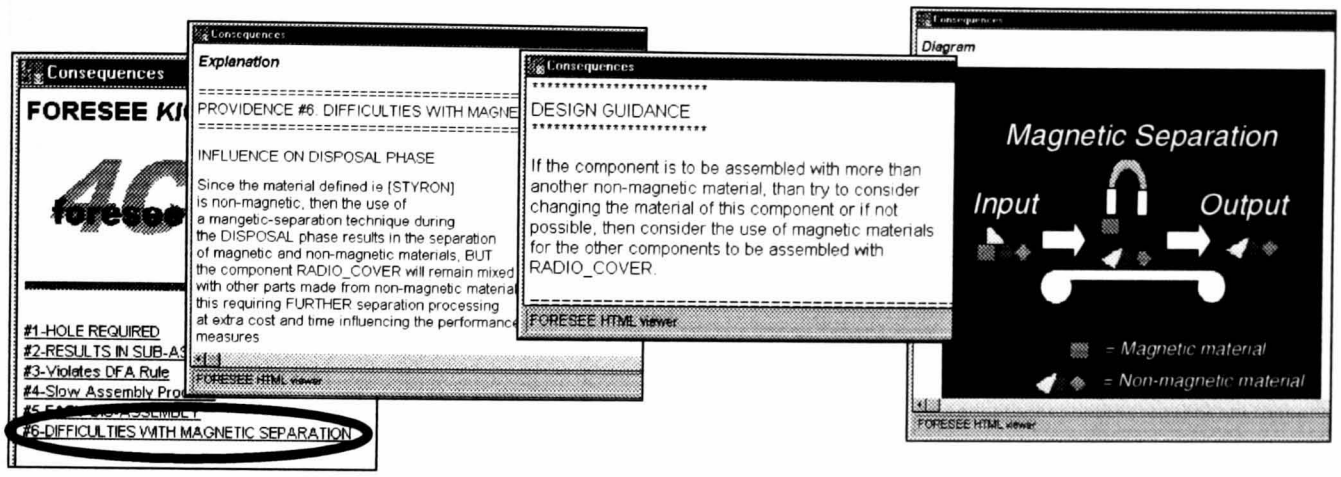


Figure 10.8 - Disposal phase consequence due to material commitment

Foreseeing compatible processes

Based on (8.31), a LCC_{ni} of this material commitment is the decision proposal D{[P]}. The set {[P]} compatible with the STYRON material is inferred by FORESEE via relationships (8.33), with the results shown as in Figure 10.9. Due to the mechanism of (8.33), even *Injection_20,* newly added by the user to the S.E.L., is matched and included in {[P]}.

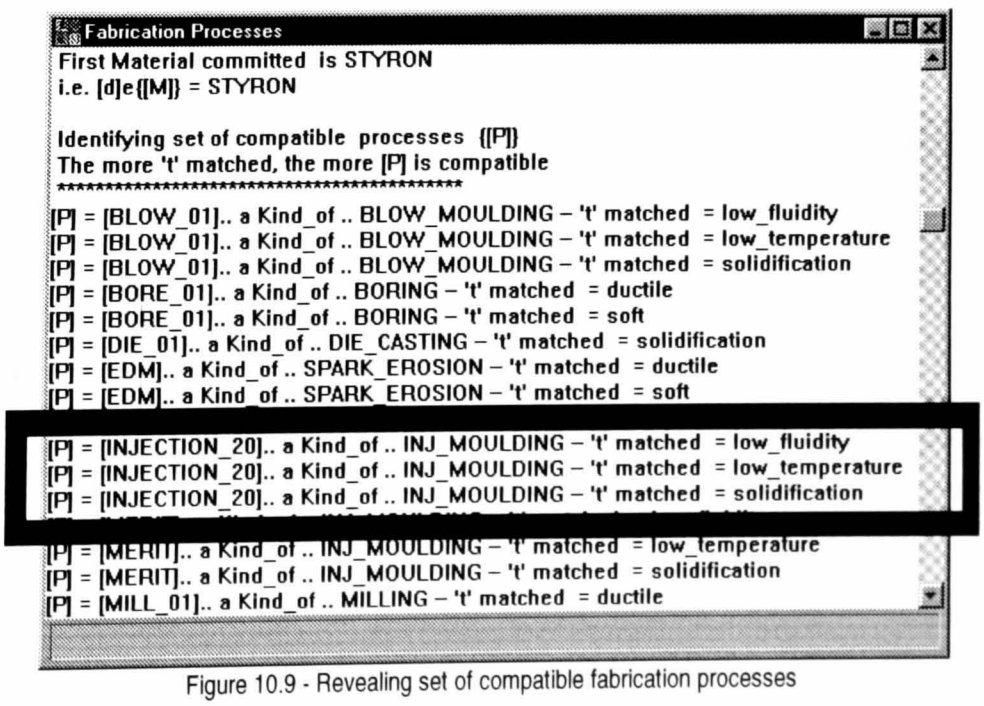


Figure 10.9 - Revealing set of compatible fabrication processes

Realization phase concurrent synthesis

From the set revealed (Figure 10.9), the designer explores committing *Injection_20* to become *part_of (f)* the *manufacturing system* (Figure 10.10). From relationship (8.41), this introduces the LCC_{ni} = 'mould is required'. FORESEE thus adds a mould as *part_of (h)* Injection_20 (Figure 10.10). From (6.8), the mould tool co-evolves its own LCC_{ni}s influencing *multiple* life-phases.

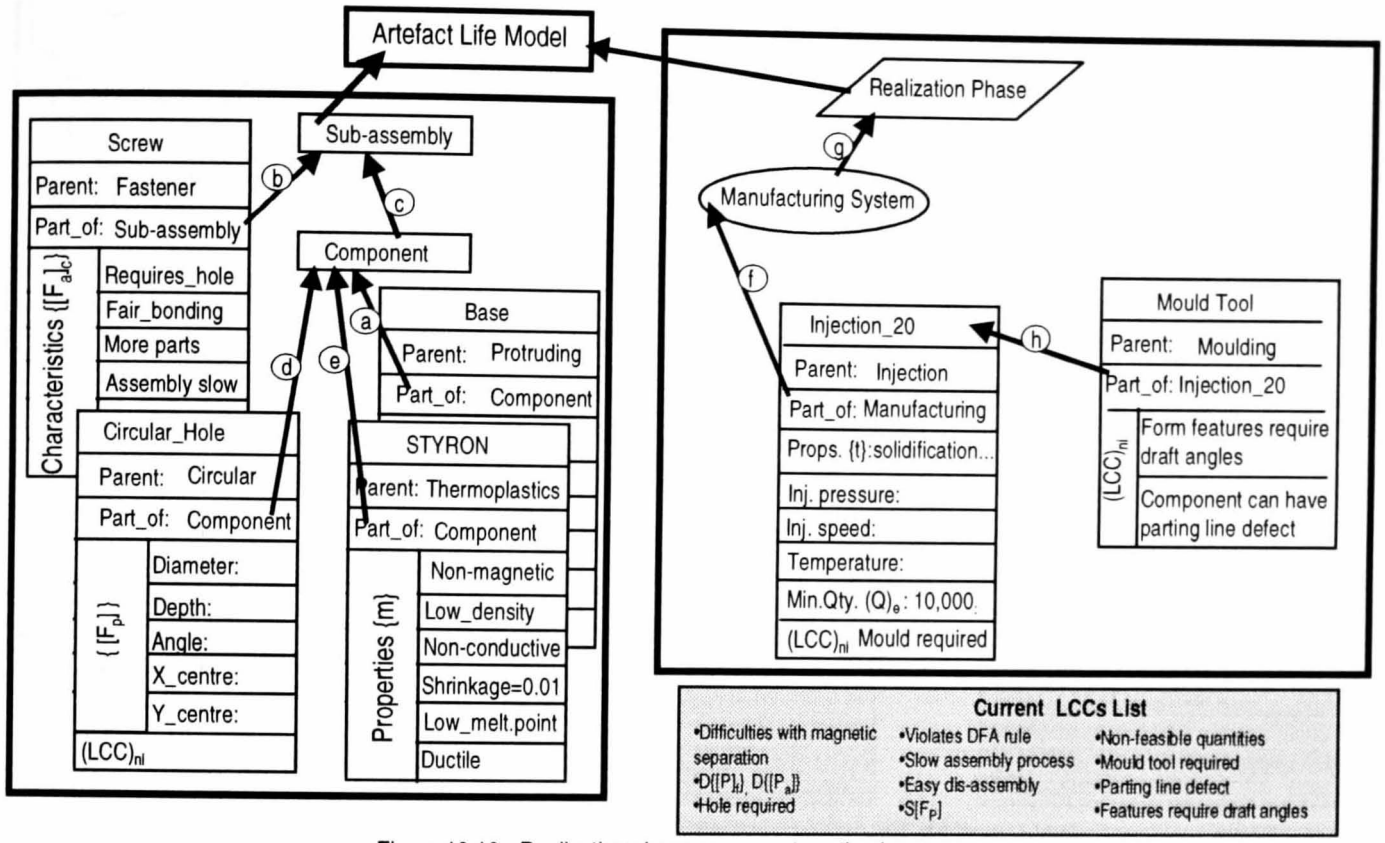


Figure 10.10 - Realization phase concurrent synthesis

Pro-active guidance on bought out items

The realization of mould tools involves the use of standard mould parts [Pye 1989] that can be purchased from different suppliers. Using embedded *LCC guidance knowledge*, FORESEE therefore *recommends* the selection of an appropriate supplier if it is to reveal, during synthesis, artefact 'life specific' LCCs. The designer accepts this recommendation and from the known standard mould part supplier options (Figure 10.11), selects supplier DME.

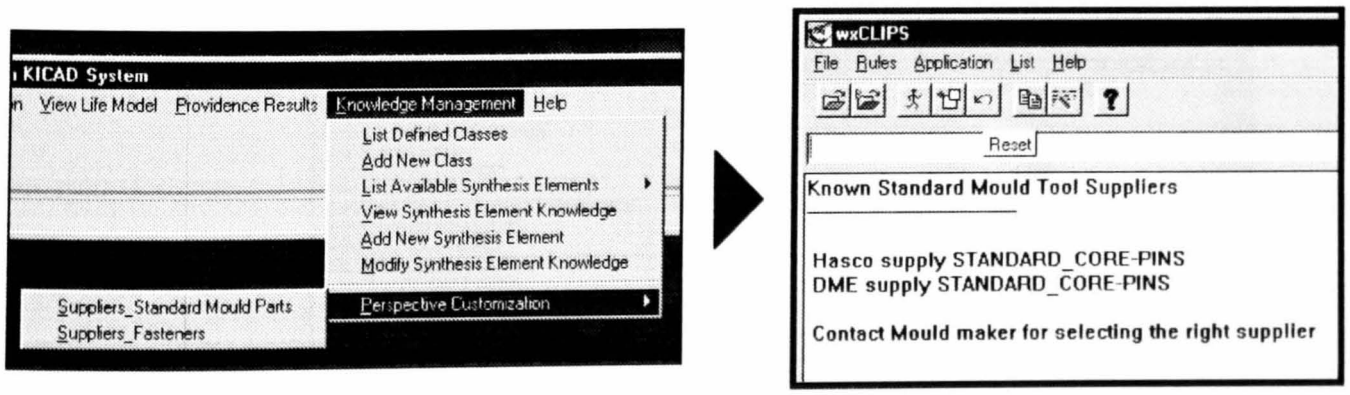
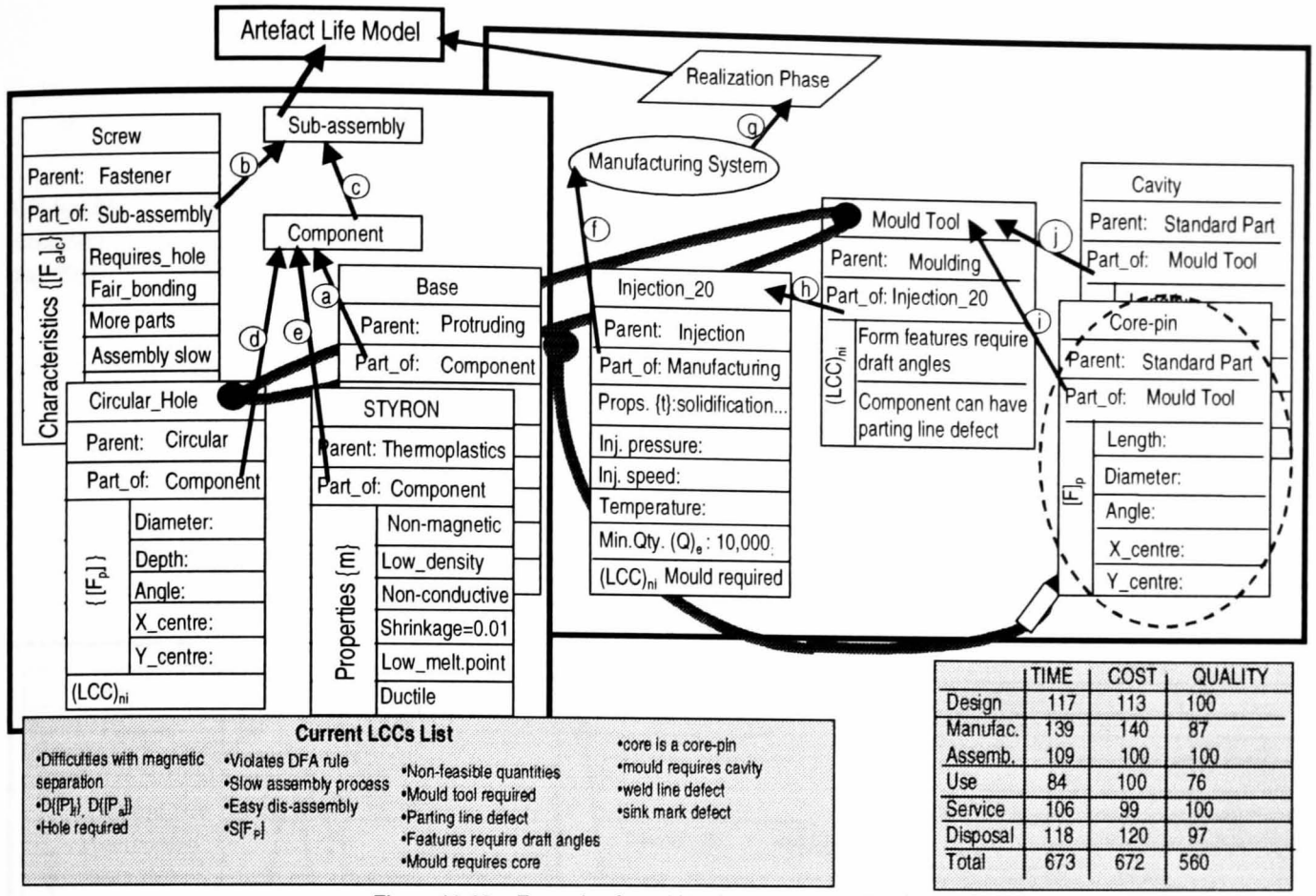


Figure 10.11 – Selection of a specific mould tool supplier

Mould tool concurrent synthesis

Due to interacting commitments in the current artefact life solution model, the LCC_i = 'core-pin required' given by (6.9) co-evolves. Thus, based on the concurrent synthesis pattern in (8.13), FORESEE adds a [core-pin] to become

part_of (i) the mould tool LCPE (Figure 10.12). Similarly, due to an interacting relationship between the [base] form feature and the [mould tool], FORESEE adds a [cavity] to become *part_of (j)* the mould tool (Figure 10.12).



	TIME	COST	QUALITY
Design	117	113	100
Manufac.	139	140	87
Assemb.	109	100	100
Use	84	100	76
Service	106	99	100
Disposal	118	120	97
Total	673	672	560

Figure 10.12 – Example of mould tool concurrent synthesis

Mould design guidance

Based on embedded *LCC guidance knowledge*, the introduction of the core-pin allows FORESEE to work independently from the designer and reveal a range of standard diameter core-pins available from the selected supplier (DME), that can be employed for mould tool design:

(10.2) $$D\{[\text{core-pin}]_{\text{diameter}}\} = \{1.5 \vee 2.5 \vee 3.0 \vee 3.5 \vee 4.0 \vee 4.5 \vee 5.0 \vee 5.05 \vee 5.1 \vee 5.5\}$$

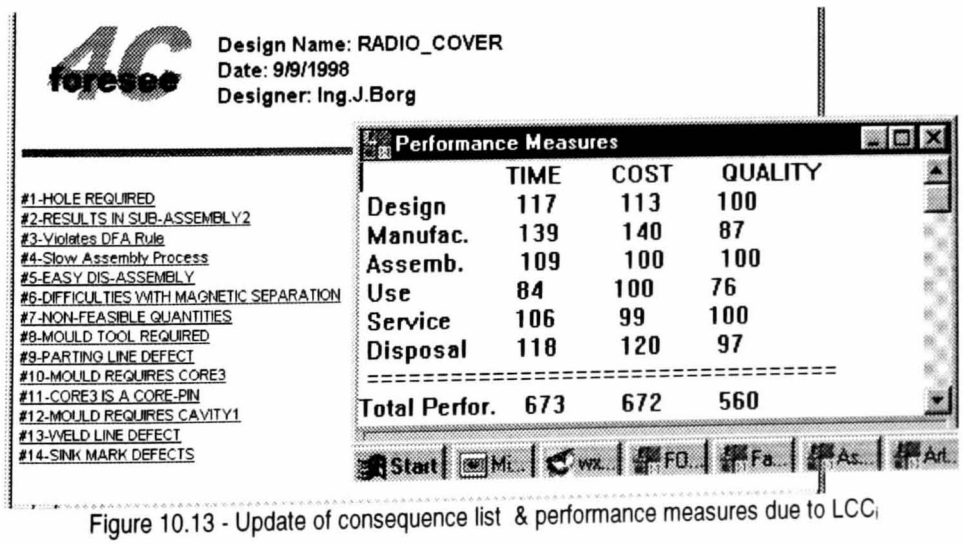


Figure 10.13 - Update of consequence list & performance measures due to LCC_i

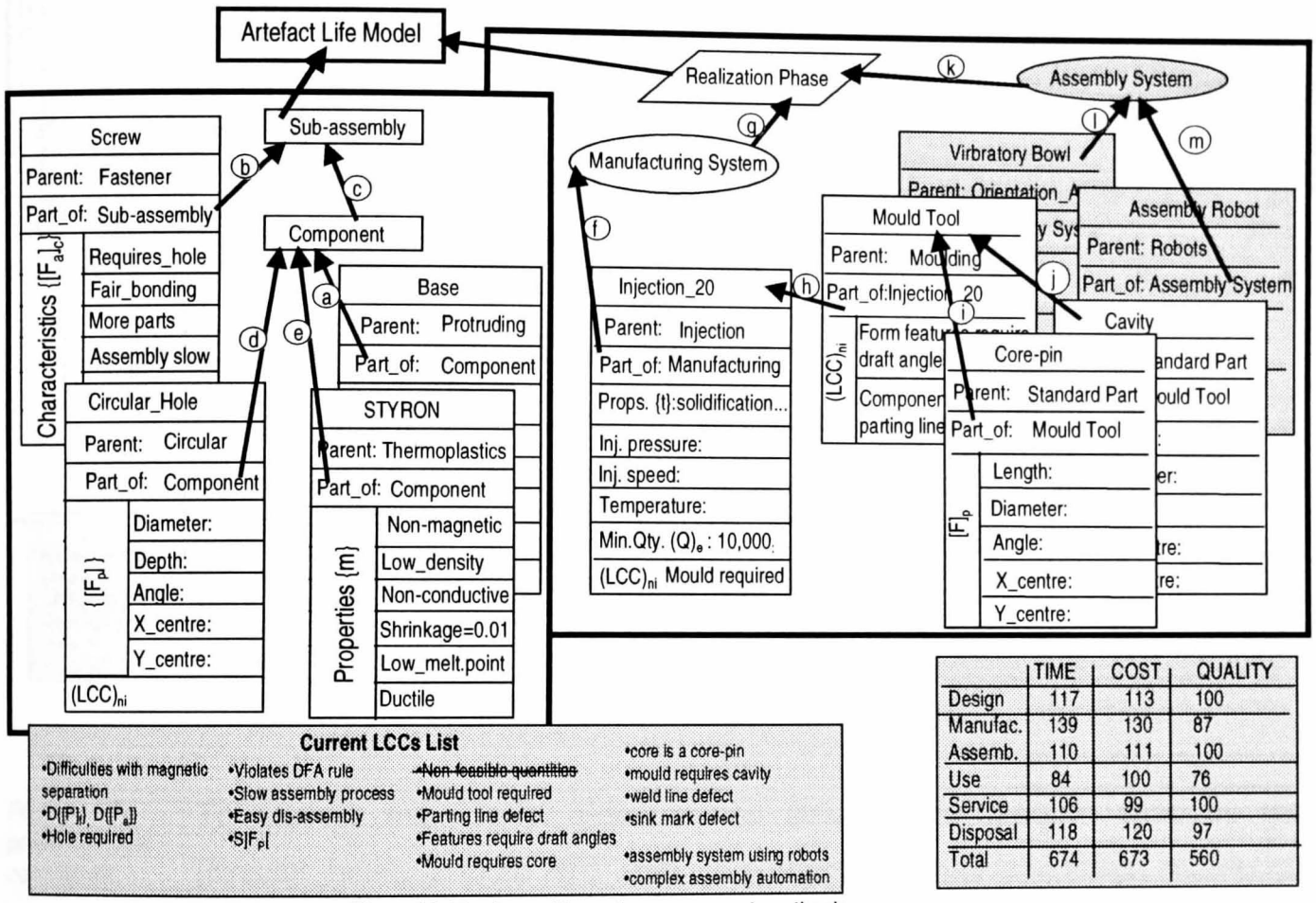
Multiple life-phase consequences

Due to a number of *interacting relationships* present in the incomplete artefact life model shown in Figure 10.12, a number of other LCC_is (Figure 10.13) are inferred. For example, from (8.61) the component can have a *sink mark*

defect, this propagating a number of LCCs across *multiple* life-phases e.g. more manufacturing time in the realization phase and a weak component structure resulting in a lower use phase quality. Similarly, due to the interaction between STYRON, the [circular_hole] and the *Injection_20* process, the current solution state will result in a *weld line* defect [Borg et al. 1998]. These new LCCs influence different performance measures (Figure 10.13).

Non-feasible production quantities

The LCPE model of *Injection_20* includes knowledge of the *minimum* feasible economic quantity $(Q)_e$, this being 10,000. This $(Q)_e$ value interacts with the specified use-phase quantity [Q] = 9,000, which through relationship (8.39) results in a LCC that *Injection_20* is not suitable due to 'non-feasible quantities'. To relax this LCC, the designer increases [Q] to 10,100 justified by catering for spare covers.



	TIME	COST	QUALITY
Design	117	113	100
Manufac.	139	130	87
Assemb.	110	111	100
Use	84	100	76
Service	106	99	100
Disposal	118	120	97
Total	674	673	560

Figure 10.14 - Assembly system concurrent synthesis

Assembly system concurrent synthesis

The new quantity ([Q]=10,100) triggers a piece of company *specific LCC guidance knowledge* that recommends assembly automation when [Q]>9,500. The designer accepts this recommendation, which through the concurrent synthesis pattern in (10.3) introduces an *assembly system* as *part_of (k)* the realization phase model (Figure 10.14).

(10.3)

IF [Q]>9,500

THEN $([d]_E\{[LCPE]\} = [automated\_assembly\_system]) \rightarrow \langle phase\rangle_{realization}$.

Further, based on the *LCC_i inference knowlege* in (10.4), the [screw], a *kind_of* ⌊Fasteners⌋, interacts with the *automated assembly system* to commit the sub-systems [bowl_feeders] and [assembly_robot] as *part_of (l)* and *(m)* respectively, the *assembly system* model (Figure 10.14). As a consequence, performance measures (e.g. assembly cost) are influenced.

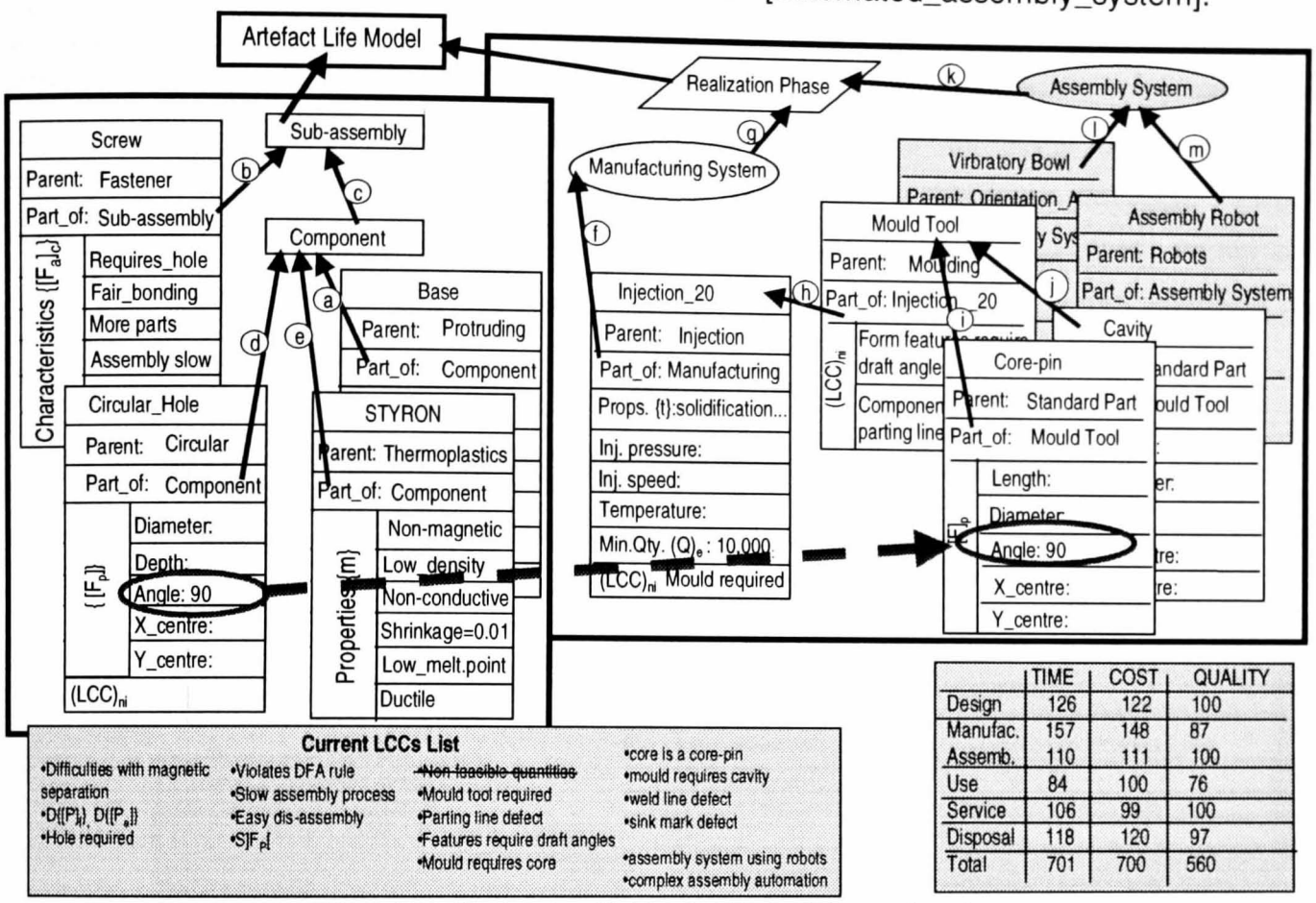
(10.4)

IF ([F_a] •→ ⌊Fasteners⌋) → <model>$_{component}$

AND ([automated_assembly_system] → <phase>$_{realization}$)

THEN ([bowl_feeders] ∧ [assembly_robot])

→ [automated_assembly_system].



	TIME	COST	QUALITY
Design	126	122	100
Manufac.	157	148	87
Assemb.	110	111	100
Use	84	100	76
Service	106	99	100
Disposal	118	120	97
Total	701	700	560

Figure 10.15 - Propagation of hole's orientation angle to core-pin angle

Feature parameter value commitment

To explore the use of side-mounted screws, the designer specifies the [circular_hole]$_{angle}$ to be 90°. Through embedded mould tool design knowledge (similar to 8.52), this value is propagated to the [core-pin]$_{angle}$ (Figure 10.15).

Consequence on ejection mechanism

As explained via the LCC browser (Figure 10.16), due to this [core-pin]$_{angle}$, a complex ejection mechanism is required. Further, from the example in (8.52), this LCC influences performance measures (Figure 10.15). Using *LCC guidance knowledge*, FORESEE therefore makes a suggestion (Figure 10.16) to avoid this LCC, which however the designer ignores at this stage.

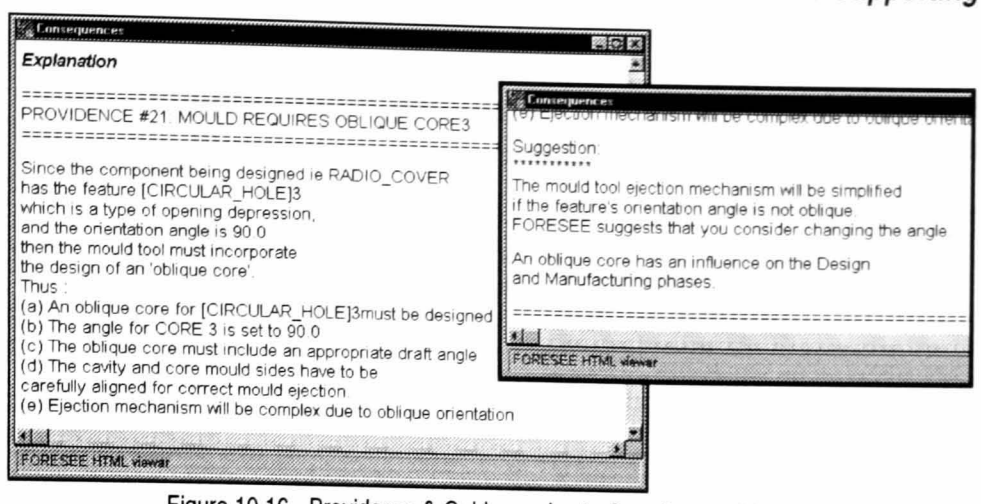


Figure 10.16 - Providence & Guidance due to Complex mould design

Consequence on mould tool construction

At this stage, the designer also details the $[circular\_hole]_{diameter}$ by committing a value of 4.9 mm. FORESEE detects this new [circular_hole] state and through embedded knowledge uses STYRON's shrinkage factor (0.01) to estimate the $[core\text{-}pin]_{diameter}$ value as 4.949. As this $[core\text{-}pin]_{diameter}$ value is not available from the standard range in (10.2) of supplier DME, FORESEE reports that as a consequence, more *time* and *cost* is required to construct the mould tool (Figure 10.17).

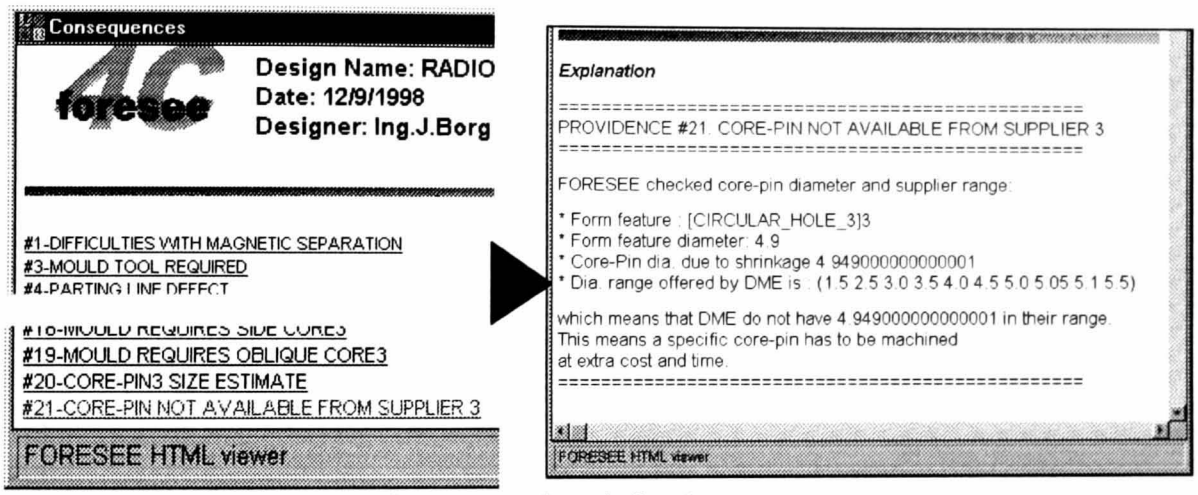


Figure 10.17 – Core-pin diameter awareness

Screw based partial solution

At this early component design stage, the different compositional models that have been *partially* and concurrently generated are illustrated in FORESEE's user interface as in Figure 10.18. Also shown are the associated *multi-X performance measures* and the list of *current LCCs*.

Investigation of LCC source commitments

By investigating the source of the currently generated LCCs, the designer decides to retract the [screw] forming *part_of (b)* this incomplete component life solution (Figure 10.15) and to explore the use of an alternative assembly feature. Due to this retraction, FORESEE makes the necessary maintenance of the performance measures, the LCC list in the browser and the evolving models (e.g. [circular_hole] is retracted, [core-pin] is removed).

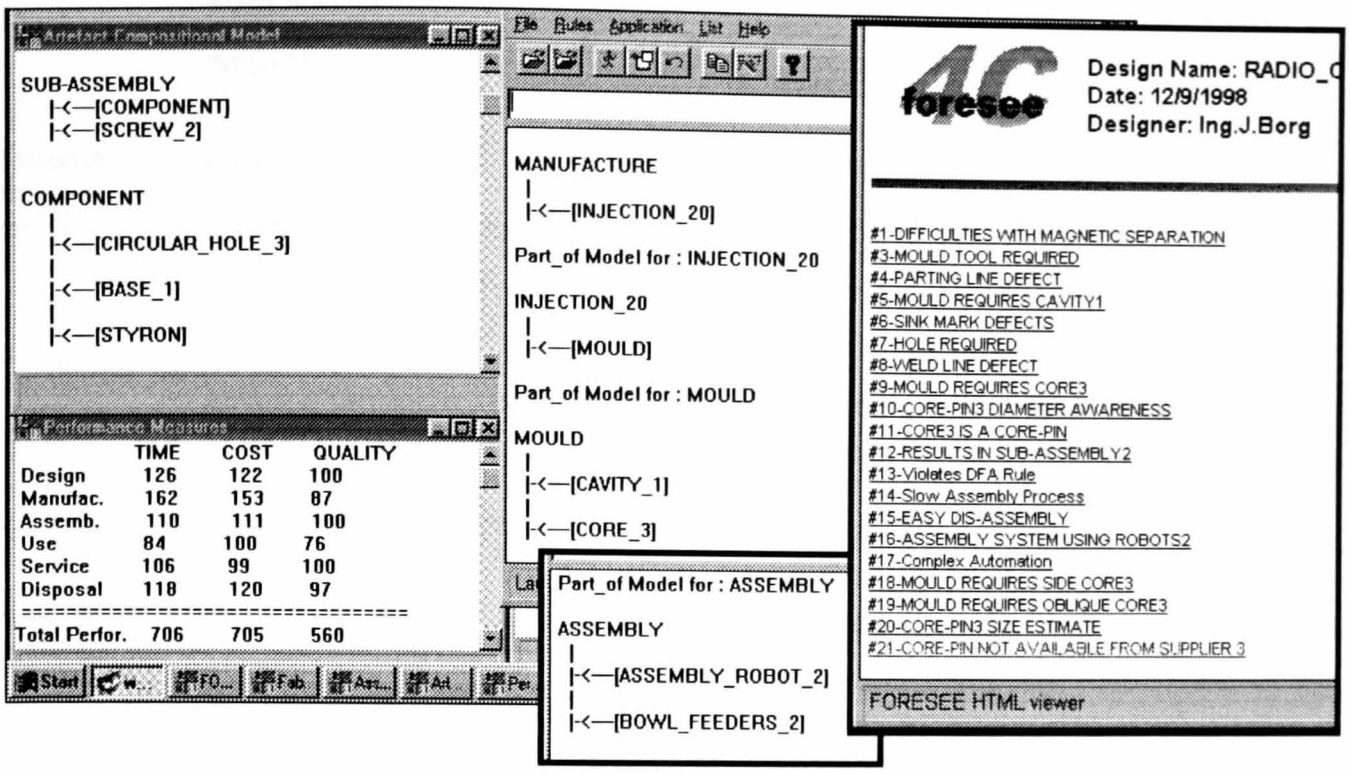
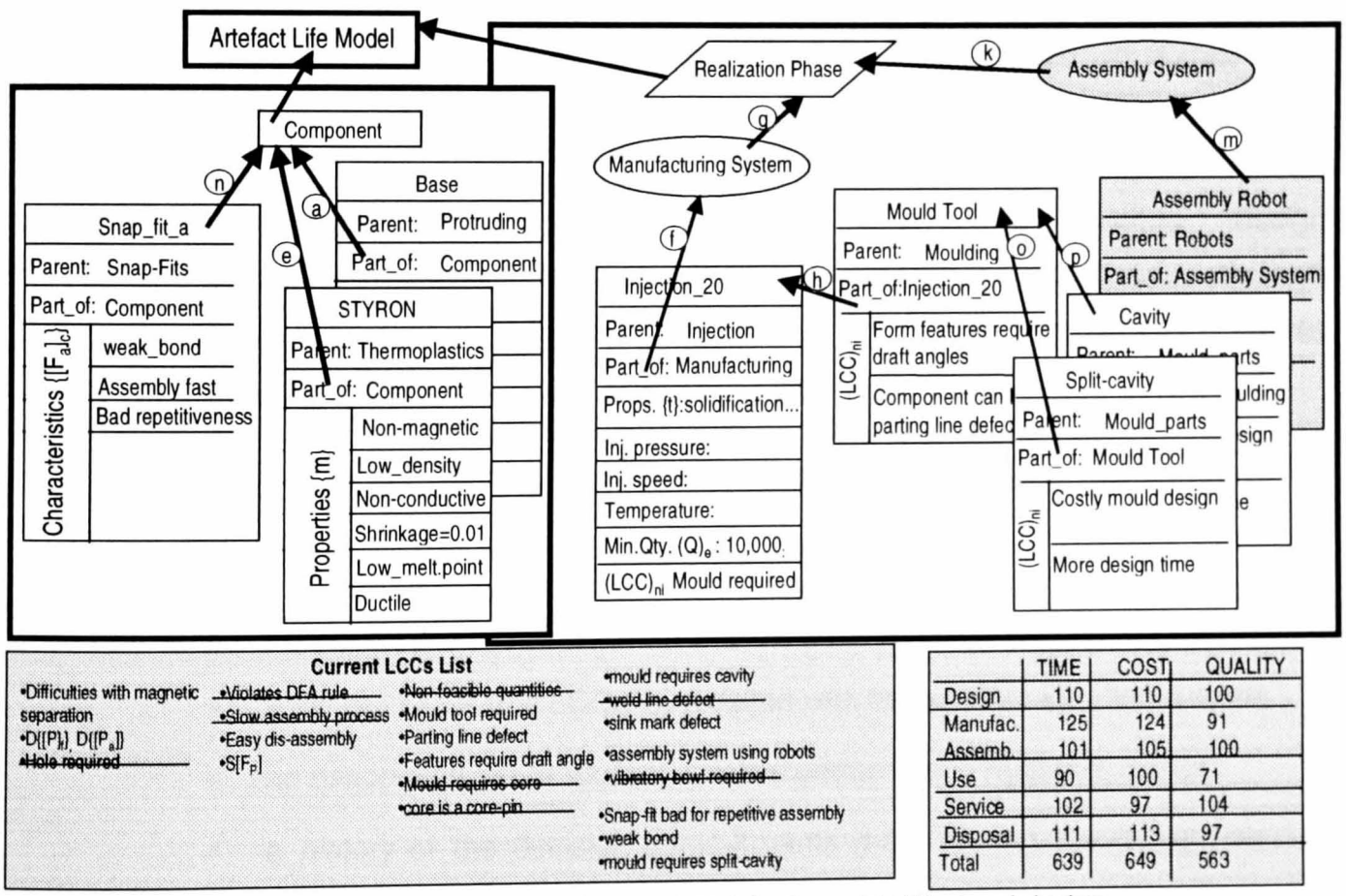


Figure 10.18 - Screw based partial artefact life solution & associated life-phase behaviour



	TIME	COST	QUALITY
Design	110	110	100
Manufac.	125	124	91
Assemb.	101	105	100
Use	90	100	71
Service	102	97	104
Disposal	111	113	97
Total	639	649	563

Figure 10.19 - Alternative, snap-fit based partial solution & associated life-phase behaviour

Exploration of an alternative partial solution

From the set of alternative assembly features (Figure 10.4), the designer now decides to commit a [snap-fit] to form *part_of (n)* the alternative solution (Figure 10.19). As from 8.57, this is also a ⌊Protruding⌋ form feature, it introduces its own LCCs, such as from 8.61, the formation of a 'sink mark defect'. This alternative, *partial* component solution, *avoids* certain LCCs (e.g. violates DFA rule, hole required, mould requires core, vibratory bowl required,

weld line defect) but as illustrated in Figure 10.19 co-evolves other LCCs (e.g. snap-fit bad for repetitive assembly, weak-bond, mould requires split-cavity).

Solution life-behaviour comparison

Using the *multi-X behaviour* values, the designer can compare the screw-based partial solution (Figure 10.18) with the alternative snap-fit based partial solution (Figure 10.20). In this case, the total performance of the snap-fit *partial* solution is considered *relatively* more life-oriented. The decision as to which alternative *partial solution* to select and proceed with *during* synthesis is still determined by the designer. It is also the designer who decides when to terminate the component's life conceptual design process

Design process continuation

The component's conceptual $D_sF\Sigma X$ process proceeds as illustrated in the above typical scenario, with the designer interacting with FORESEE to build the artefact model, life-phase models and explore alternative commitments (e.g. materials). Although the scenario has been presented in a specific sequence, in reality, the designer can make synthesis decision commitments in the order desired. As demonstrated, by using the embedded domain LCC knowledge model, FORESEE can work in the background independently from the designer to reveal LCCs co-evolving with the solution, update performance measures, maintain models and provide design guidance/explanation knowledge. Further, where appropriate, through captured concurrent synthesis patterns, FORESEE automatically makes commitments to co-evolve solution models.

$D_sF\Sigma X$ session output

At the end of such a design session, FORESEE provides :

- an early *artefact* compositional model;
- a number of early *life-phase system* compositional models;
- a list of LCC_{ni}s and LCC_is associated with the artefact life model state;
- an associated set of *multi-X, relative* performance measure values;
- a *history* of the decision commitments made during the design session and the associated fluctuations in performance measures.

10.3 Critical Evaluation Results

Structured interview results

The collective results of the structured interviews made at the end of the design scenario demonstrations are expressed in graphical form in Appendix M and are discussed next. For reference to a specific graph, the interview question number *e.g. Q5,* will be referred to in the text. Due to the evaluation

criteria and the evaluation's aim, the review results are presented in terms of FORESEE's support to:

- *providence*
- *life-oriented synthesis decision making*

Further, other results revealed are then presented to disclose:

- *the practical acceptance of the developed means to $D_sF\Sigma X$*
- *strengths & limitations of FORESEE's functionality.*

10.3.1 Providence Support

Are designers supported in becoming aware of unintended LCCs co-evolving during solution synthesis?

As Q2 indicates, 96% of the evaluators reported that they were made aware of LCCs *during* component definition. Further, as indicated by the result of Q3, 74% were made aware of LCCs when the solution was still *imprecise* whilst Q4 reflects that 96% were made aware of LCCs when the solution was still *incomplete.* Collectively, these three results indicated that designing via FORESEE effectively allows designers to foresee LCCs *during* synthesis. This has been explicitly stated by one participant:

> 'the approach *integrates* the activity of designing components with the activity of taking into consideration product life issues, which is *fundamentally* different from *first designing*, and *then* at a penalty of extra time, analysing the solution for conflicts with product life issues'.

Are designers assisted in becoming aware of multiple and interacting LCCs?

The results of Q5 indicate that designers become simultaneously aware of *multiple* life-phase consequences - 64% of the evaluators indicated that awareness concerned 'multiple and interacting' LCCs. One industrial participant in fact stated that:

> 'this knowledge intensive approach allows me to consider 6 times as many life aspects as I normally do.'

Are designers made aware of life-specific or generic LCCs?

The results of Q6 indicate that 24% of the evaluators considered the LCCs revealed to be generic. The rest considered LCCs revealed to be *solution* (31%) and/or *company* (45%) specific.

10.3.2 Life-oriented Synthesis Decision Making

Are designers motivated and supported in exploring alternative commitments?

87% reported (see result of Q8a) that the awareness of LCCs *motivated* them to *explore* alternative decision commitments. A reason provided is that due to LCC awareness, evaluators were motivated to try and avoid LCCs revealed so as make the currently evolving solution more life-oriented. Further, the result of Q8b indicates that only 26% of the evaluators felt that this awareness would

hinder their design freedom. As the results of Q9 indicate, 65% considered the use of running *cost, time* and *quality* performance measure values as suitable for *monitoring,* during synthesis, the impact of decision consequences on the different life-phases. As some evaluators commented, this supports them in rapidly exploring alternative commitments. In this sense, as stated by one practicing designer:

> 'the tool makes it possible for designers to engage in "what-if?" artefact life scenario exercises'.

Concerns on type & magnitude of performance measures

However, evaluators reported that there may be other measures of interest (e.g. environmental) or that some metrics (e.g. cost) might be more important than others. Further, some evaluators stated that the magnitude of the fluctuations might be debatable. For example, is a unit of time in the design phase, equivalent to a unit of time in the use phase? Also, one evaluator argued whether certain measures e.g. quality, could be assigned a value.

Are designers guided to make commitments consciously with respect to artefact life issues?

Nevertheless, 83% of the evaluators (Q8c) reported that awareness of LCCs co-evolving with their solution, assists them in committing synthesis decisions more consciously. A reason given is that a decision-maker is guided to take into consideration a wider range of issues than originally considered.

10.3.3 Practical Acceptance of Developed Means to $D_sF\Sigma X$

Is the approach considered useful in practice?

83% of the evaluators (see Q13 results) appreciated how captured LCC knowledge could be timely reused in new component design situations. In this sense as stated by some evaluators, FORESEE addresses a practical problem they encounter – the loss of valuable expertise associated with the change of designers and other artefact life actors within an organization.

Some evaluators also reported that the detection and reporting of LCCs *during* synthesis via FORESEE, allows designers not familiar with the LCCs detected, to *acquire* LCC knowledge. As stated by one of the industrial evaluators:

> 'it is an excellent tool for relatively inexperienced designers and also for designers such as electrical engineers who are now expected to do mechanical design'.

Thus, as the result of Q15 explicitly indicates, provided that more knowledge is embedded within FORESEE, 91% of the evaluators considered the 'KC' approach useful in practice. Actually, 100% of the evaluators who were practicing or consulting designers (see Appendix M.2) agreed to this.

Designers' resistance

However, two separate product development consultants were concerned with *whether* designers would want to design in that ('KC' approach) way. They argued that designers tend to be unwilling to adopt new approaches/tools as they wrongly believe that they already take artefact life issues into consideration and feel that using a tool such as FORESEE will add more time to their design process.

Support limited to component design

Although evaluators were briefed beforehand that FORESEE focuses on component design, some evaluators considered this a weakness, due to the varying size and complexity of artefacts designed in industry. They argued that the 'KC approach' concept would be enhanced if FORESEE could be equally applied to other *artefact system levels* (see section 2.1).

LCC knowledge concerns

As the approach is LCC knowledge intensive, some evaluators were also concerned with the following knowledge related issues:

- Where does all this LCC knowledge come from?
- Who will maintain the ever-expanding knowledge base?
- How true (certain) is a detected consequence?

Designer ethical concerns

Some ethical concerns were also raised by the industrial evaluators :

- Will a tool such as FORESEE 'downgrade' designers? i.e. will designers feel they have little to offer from their own skills?
- Will designers rely less on themselves and more on the support provided by a tool such as FORESEE ?
- Who is responsible for the design solution, the *designer* or the *KICAD system* supporting the designer? Does this mean that the *knowledge engineer* is now responsible for the design solution?

10.3.4 FORESEE System's Functionality

Handling of LCCs

Are LCCs reported adequately in FORESEE ?

Results of Q10a indicate that only 57% of the evaluators consider LCCs as being reported adequately by FORESEE. The simultaneous reporting of LCCs propagating across multiple life-phases was seen by some as a disturbance to their ability to focus on individual issues. Also, comments were made that with FORESEE, users are not supported in distinguishing between LCCs that were generated from previously made commitments and the last

commitment. Also, no indication is provided as to which LCC is most severe. Further, Q10b indicates that 50% of the evaluators considered that *locating* and *browsing* information via FORESEE on specific LCCs was easy.

Inadequate presentation of multi-X behaviour

Although evaluators considered the concept of monitoring fluctuations in performance measures good, a few commented that as implemented in FORESEE, displaying fluctuations in numbers, that quickly and frequently change, is probably worse than using any monitoring of life-phase behaviour at all. A better means to display the life behaviour was thus demanded.

Lack of clear presentation of exploration advice

Q11 indicates that 45% of the evaluators found the *advice* and *guidance* to be explicitly clear. Further, an industrial evaluator commented that FORESEE does not indicate what new value a parameter should have.

Useful decision commitment history format

66% of the evaluators (see Q12) found that FORESEE's capability to automatically retain a history of the decision commitments made and their resulting LCC influence on mulltiple performance measures, to be useful. However, some commented that the way history information is presented to the user needs to be improved.

Solution Modelling Capability

Solution reuse and concurrent modelling

87% of the evaluators (see Q1 results) considered FORESEE to support the generation of solution compositional models by the reuse of *PDEs*. Further, Q7 results indicate that 66% considered FORESEE to support designers in concurrent synthesis - generating early *artefact* and *life-phase system* compositional models.

Only one 'artefact life' solution

However, since FORESEE only handles *one* evolving *artefact life model* at any one time, some evaluators argued that this makes it difficult to rapidly compare LCCs associated with *alternative* evolving artefact life solutions.

Non-traditional solution display

Some evaluators commented that an evolving compositional model expressed as a *text based* hierarchy is not the traditional manner in which designers visualize early design solution concepts.

LCC Knowledge Management Capability

LCC_i knowledge management not supported

91% of the evaluators (see Q14 results) appreciated that FORESEE allowed *new* synthesis elements to be added and their knowledge content, including LCC_{ni} knowledge, changed. However, some commented that the practical success of the 'KC' approach depends on the continuous addition of new

knowledge, as and when it becomes available. FORESEE does not allow new LCC_i knowledge to be *added* or *modified* on-line by the user.

User Interface

Demands unnecessary designer effort

In order to avoid syntax errors, FORESEE requires its user to learn its 'language' (e.g. valid names for synthesis elements) thus demanding unnecessary user effort and time. Further, some commented that the user-interface employs too many clustered window panels. Also, user-defined synthesis elements are not automatically included in pull-down menu options.

10.4 Chapter Conclusions

Evaluation approach

As concluded in Chapter 4, existing means do not adequately support $D_sF\Sigma X$ from the perspective of providence. To address this situation, Part 'B' of this dissertation presented the development of a computational 'KC' approach framework. The realization of this framework, the prototype FORESEE, has been utilized to evaluate its effectivenss to supporting component $D_sF\Sigma X$, with respect to a number of evaluation criteria presented in section 10.2. For this purpose, 23 evaluators, having a mixture of industrial and academic backgrounds, were given a demonstration of utilizing FORESEE during a $D_sF\Sigma X$ scenario. This demonstration was followed by a structured interview.

Overall evaluation result

The evaluation results, presented and critically discussed in section 10.3, are summarized in Table 10.1. Globally, these results provide evidence that the realized 'KC' approach framework – FORESEE, allows designers to:

- foresee *multiple* and *interacting* LCCs co-evolving with an incomplete and imprecise component solution *during* the synthesis activity;
- be provided with a *solution life-specific insight* rather than a generic one;
- be motivated and guided in 'artefact' and 'life-phase system' exploration, to avoid/relax detected LCCs, this leading to a more 'life-oriented' conscious commitment of decisions.

Moreover, 91% of the evaluators consider the concept of the realized means to supporting $D_sF\Sigma X$ as being useful in practice. However, the evaluation has also revealed that before FORESEE can be practically applied to support $D_sF\Sigma X$, a number of improvements are required to FORESEE's functionality and to the underlying 'KC' approach framework. These are discussed in the next chapter.

Table 10.1 – Summary of Evaluation Results
(+ Strengths - Limitations)

Providence Support

+ Component synthesis integrated with the activity of foreseeing component LCCs.

+ Designers become simultaneously aware of multiple and interacting LCCs during component synthesis.

+ Designers made aware of company and/or life-oriented LCCs during synthesis.

Life-Oriented Synthesis Decision Making Support

+ Designers motivated and guided to consider exploring a wider range of issues than originally conceived.

+ Monitoring the virtual artefact life-behaviour supports early artefact life exploration.

+ Awareness of LCC does not hinder the designer's freedom.

- To indicate the influence of a decision commitment only a 'specific number' of performance measures are employed.

- The magnitude of performance measures is only useful for relative comparisons.

Practical Acceptance of Developed Means to Supporting $D_sF\Sigma X$

+ The 'KC' approach concept as demonstrated by FORESEE has been rated as useful for practical applications by 91% of the evaluators.

+ A practical benefit is that it integrates synthesis with 'on-the-job' LCC training.

- Focuses on design at the component level.

- It may be difficult to employ the 'KC' approach concept due to the lack of willingness by designers to adopt new approaches/tools.

- The effectiveness of the approach depends on the availability and maintenance of a vast amount of LCC knowledge.

- Raises a number of ethical issues concerning the role of human designers.

FORESEE System's Functionality

+ It supports solution 'reuse'

+ It supports LCC knowledge retention and reuse.

+ Supports early 'artefact life' solution modelling.

+ Maintains the session's decision commitment history in a useful format.

- LCCs are reported in an inadequate way: limited ease to locate & browse specific LCC information.

- Artefact life-behaviour monitoring is inadequately displayed.

- Tool lacks to present clear exploration and design guidance advice.

- Early solution is displayed in a non-traditional way.

- Handles only one 'artefact life' solution at a time;

- Does not provide on-line LCC_i knowledge management support

- Current user-interface demands unnecessary effort & time from its users.

chapter 11

11.0 Discussion

Scope

The underlying theme of the research reported in this dissertation has been the development of a $D_sF\Sigma X$ approach to support mechanical artefact designers in generating 'life-oriented' conceptual design solutions. The scope of this chapter is to review the work carried out and discuss some of the general implications of the established computational *'Knowledge of life cycle Consequences' approach framework* to $D_sF\Sigma X$. For this purpose, section 11.1 presents the research carried out to disclose the original contributions made. Section 11.2 assesses the implications of these research results. The validity of the results is discussed in Section 11.3. Section 11.4 discusses avenues for future research work. Chapter conclusions are presented in section 11.5.

11.1 Research Results

Main contribution

As a result of the research carried out, the following *original* contributions have been made:

i. the establishment of a *framework* for a *'**K**nowledge of life-cycle **C**onsequence' (KC) approach* to support $D_sF\Sigma X$ (Chapters 7-9);

ii. *a phenomena model explaining LCC generation* (Chapter 6);

iii. *a $D_sF\Sigma X$ computational model* (Chapter 9) and

iv. *an incremental understanding of the $D_sF\Sigma X$ problem* (Chapters 2-4).

The *'KC' approach framework* is the main contribution arising from this Ph.D. This contribution and the other results arising from this research work are related as illustrated in Figure 11.1. They are discussed here in the order of the *computational means development framework* (Figure 1.3).

An Incremental Understanding Of The $D_sF\Sigma X$ Problem

An incremental understanding to the problem reality

Based on the work in Chapters 2 and 3, it can be concluded that:

- decisions made during the design phase can influence the *interaction* that takes place between an artefact and the different life-phase systems encountered. This results in decision consequences *propagating* across multiple life-phases - termed *life-cycle consequences* (LCCs);

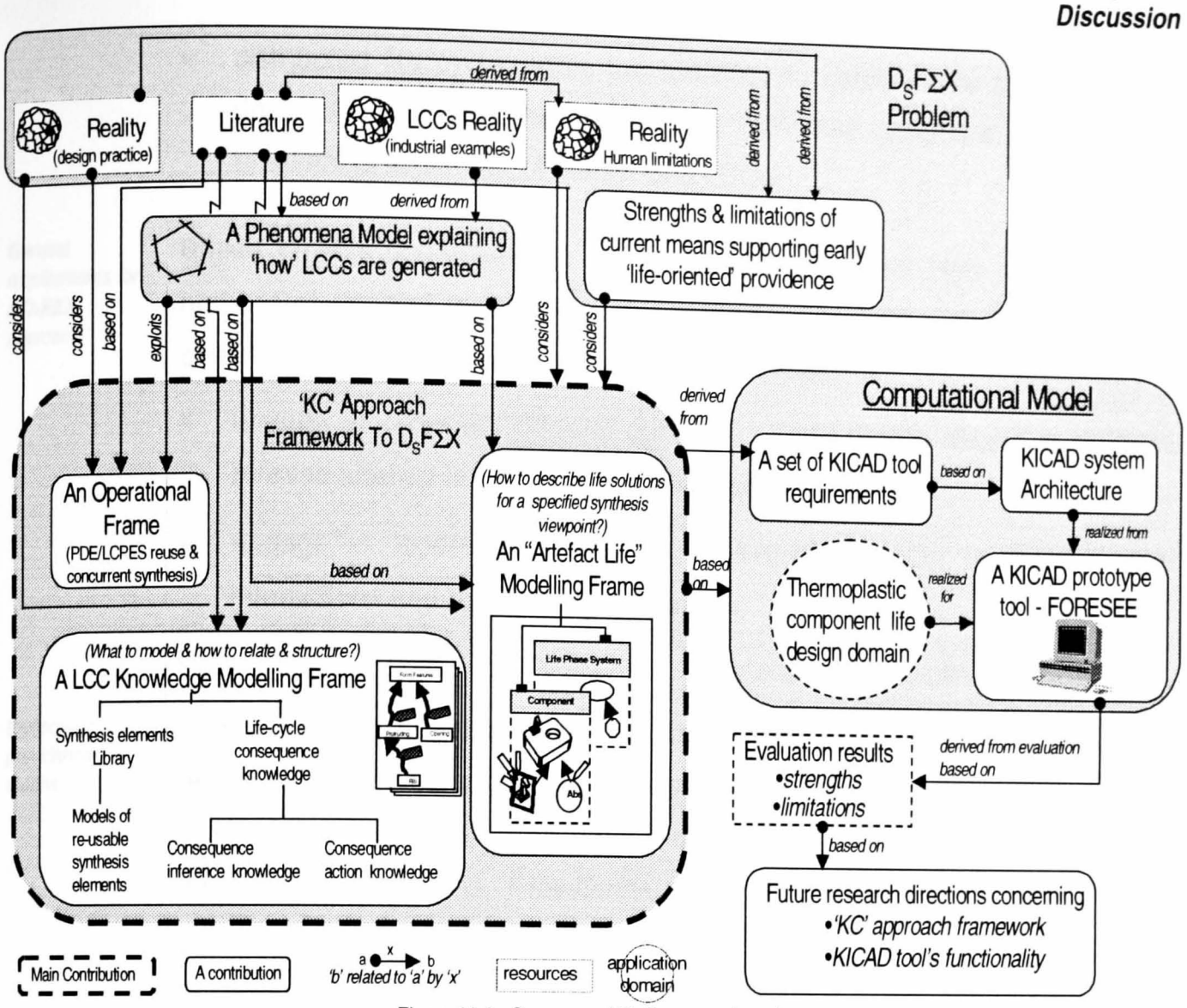


Figure 11.1 - Summary of Ph.D. research work

- in general, these LCCs are (i) an influence on the artefact's behaviour (ii) an influence on the life-phase systems' behaviour; (iii) creation of a new decision space in the same or different life-phase and (iv) the introduction of new artefact life-phase constraints;

- due to this *LCC propagation effect* phenomena, the designers' responsibility covers all artefact life-phases. For 'life-oriented' design solutions, designers should therefore adopt DFΣX, which is fundamentally different from an approach in which designers employ *multiple* but *segmented DFX* approaches i.e. not ΣDFX. Further, designers need to adopt a 'Design Synthesis for Multi-X' ($D_S F\Sigma X$) approach if they are to *generate* 'life-oriented' design solutions (see Figure 3.18);

- during an artefact's life, LCC knowledge about 'artefact and life-phase system' interactions is being generated. As this takes place *after* the design phase, designers do not generally acquire such LCC knowledge. Thus they require access to a large amount of distributed LCC knowledge;

- distributed life cycle actors are too busy to continuously form part of a design team. Thus, during synthesis, individual designers are *frequently unaware* of a number of unintended LCCs.

General requirements for a $D_sF\Sigma X$ approach

Based on this understanding, this research contributed (see section 3.3) a number of *general requirements* for a $D_sF\Sigma X$ approach. Key requirements established are that designers need to collectively :

- engage in *provident thinking* during the early design stages in order to *foresee* artefact life interactions and their propagations;
- engage in *'artefact life' solution exploration* in order to foresee such interactions and their co-evolving LCCs;
- engage in the *concurrent synthesis* of artefact and life-phase systems.

Inadequate providence means

However, a review of current providence means (Chapter 4) revealed that designers are not adequately supported in $D_sF\Sigma X$ from the perspective of providence. Individually, the support means reviewed exhibit one or more of the following limitations. Providence:

- takes place *late* during candidate solution analysis, not during synthesis;
- covers a *narrow* and *segmented* rather than multiple, life-span view and
- is *generic* and not 'life-specific', as life-phase modelling is not supported.

This review helped establish the scope of providing designers with a 'LCC knowledge' intensive means to pro-actively support them in foreseeing and exploring LCCs co-evolving with an early design solution description. Given that designers have knowledge processing limitations, the research problem therefore essentially concerned *how* LCC knowledge can be *modelled* and be explicitly *utilized* through a computational framework, during synthesis.

A Phenomena Model Explaining LCC Generation

Explains how LCCs are generated

As a foundation for developing a computational 'LCC knowledge' based approach, a *LCC phenomena model* (see Chapter 6, Figure 6.15) was established. This model contributes an explanation of how LCCs are generated in two fundamentally different ways :

i. by *individual* (i.e. non-interacting) synthesis decision commitments. Such a LCC is generated by *one* specific synthesis decision commitment made to the artefact life model, *independent* of other commitments made. Such a non-interacting type consequence, is denoted by LCC_{ni}.

ii. by *interacting,* synthesis decision commitments. A LCC in this case is generated from the *interaction* between a set of **'n' (n >1)** specific synthesis decision commitments made to the artefact life model. Such an interacting type consequence is denoted by LCC_i.

Highlights 'need' of concurrent synthesis

This explanation highlights that the concurrent synthesis of *'artefact'* and *'life-phase system'* solutions is necessary *if* LCC_is resulting from the interaction of decision commitments are to be revealed and catered for *during* synthesis.

Discloses 'what' to model and 'how' to relate

Further, the phenomena model discloses that the *source* of LCCs are synthesis decision commitments. In the case of component concurrent synthesis, these commitments concern PDEs and LCPEs. The phenomena model thus provides a general foundation for modelling *causal relationships* between synthesis decision commitments concerning PDEs or LCPEs and LCCs to develop a domain specific 'LCC knowledge model'.

Explains how LCC knowledge can be utilized

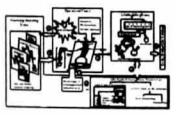
The explanations provided by the phenomena model therefore reflect how a 'LCC knowledge model' can be utilized as a means for :

a) *LCC awareness*: depending on commitments made, LCCs (including those unintended) can be causally revealed during synthesis;

b) *solution synthesis guidance*: designers can identify which elements can be committed in order to generate an intended LCC;

c) *avoiding/relaxing the formation of unintended LCCs:* designers can be guided to *which* specific commitments (interacting or non-interacting) can be retracted in order to explore avoiding/relaxing a LCC.

'KC' Approach Framework For Supporting $D_sF\Sigma X$



The framework (Chapter 7) of the 'KC' approach to $D_sF\Sigma X$ is based on the explanations provided by the phenomena model. The approach concept is to *guide* designers in generating life-oriented conceptual solutions, by explicitly making them aware of the *current LCCs* co-evolving with their synthesis decision commitments and propagating across multiple life-phases. For this purpose, the 'KC' approach framework developed consists of three frames (Figure 7.3) providing a general description of how to *model* LCC knowledge and disclose the general principles of how to *reuse* the modelled LCC knowledge in a *timely* manner during the synthesis and exploration of a solution description.

LCC Knowledge modelling frame

The *LCC Knowledge Modelling Frame* provides a formal explanation of *what* to *acquire*, *model* and *relate* for an application domain, together with *how* to transform and structure the established relationships into *meaningful LCC inference* and *action knowledge* (Figure 7.4). This frame, detailed in Chapter 8 for the mechanical component domain, provides a *formalism* of:

i. *a synthesis element library*, consisting of various models of PDEs and LCPEs reused during the design and life of a mechanical artefact domain. In the case of components, these include models of form features, assembly features, materials and LCPEs such as fabrication systems and assembly systems;

ii. *relationships* between PDEs, LCPEs and LCCs describing how to model:
   (a) *LCC inference knowledge* for revealing both LCC_{ni}s and LCC_is;
   (b) *LCC action knowledge* for
      - LCC to performance measure mapping;
      - concurrent synthesis patterns;
      - guidance and explanation of LCC avoidance/relaxation.

Operational frame

The *Operational Frame* presents:

i. a *model of the $D_sF\Sigma X$ working environment* composed of the human designer; a set of domain specific PDEs and LCPEs; an LCC knowledge model for the domain specific elements; a means by which the designer can search for synthesis elements resulting in intended consequences; a means by which a designer can interact with and evolve an artefact life solution model; a communications medium providing designers with awareness of co-evolving LCCs and guidance to their avoidance/relaxation;

ii. a description of the *$D_sF\Sigma X$ mode of operation*, this being: designers interact with a synthesis elements library (PDEs and LCPEs); synthesis decision making takes place as in the model of Figure 6.8; commitments can be made in any order and at a least or specific level; the designer is engaged in concurrent artefact and life-phase systems' synthesis.

As explained in section 7.4, through this *$D_sF\Sigma X$ working environment* and *$D_sF\Sigma X$ mode of operation*, the designer can search for synthesis elements (i.e. PDEs or LCPEs), commit/retract elements to evolve or modify the artefact life model, specialize elements, take into consideration co-evolving LCCs revealed, and explore alternative commitments based on design guidance provided through the available LCC knowledge model.

Artefact life modelling frame

The *Artefact Life Modelling Frame* (Figure 7.5), is concerned with providing a formalism for describing an evolving *'artefact life' compositional model,* for a domain application, artefact system level (e.g. sub-assembly) and for a synthesis viewpoint (e.g. functional). This formalism is required to support 'artefact life model' based LCC inference. In this research, the formalism established (see section 8.2.3) concerns *component life solutions* described from a constructional viewpoint, consisting of an evolving conceptual component model and evolving life-phase system models. These models are composed of a set of component domain specific PDEs and LCPEs linked with *part_of* relationships.

A $D_sF\Sigma X$ Computational Model

How to codify LCC knowledge into computable form

Chapter 9 disclosed *how* the established 'KC' approach framework can be realized in computational form as a KICAD tool. As highlighted in Figure 11.1, this contributes:

- a set of requirements (section 9.1) for a $D_sF\Sigma X$ KICAD tool;
- a relevant KICAD tool architecture (section 9.3);
- how through the utilization of a hybrid of existing knowledge representation schemes, the established LCC knowledge model formalism can be codified into computable form (see Figure 9.9).

A prototype, FORESEE, was realized for the thermoplastic component domain (Chapter 9). As implemented, FORESEE provides designers with: an early artefact compositional model, a number of early life-phase system compositional models, a list of LCC_{ni} and LCC_i associated with the current solution state, an associated set of *multi-X*, relative performance measure values, explanations (via a hypermedia browser) of LCCs and guidance to their avoidance, a history of decision commitments made during the design session and their associated fluctuations in performance measures.

11.2 Research Result Assessment

FORESEE was utilized as explained in Chapter 10, to provide a basis for assessing the strengths and limitations of the 'KC' approach framework. This assessment is discussed in this section.

11.2.1 Strengths

Support To $D_sF\Sigma X$

Based on an empirical evaluation of FORESEE (Chapter 10), it has been established that the 'KC' framework allows designers to:

Exploits LCC phenomena to proactively support $D_sF\Sigma X$

i. become explicitly aware *during* synthesis of co-evolving LCC_{ni} and LCC_i, propagating across *multiple* life-phases;

ii. be made aware of LCCs co-evolving with both *least* and *specific* synthesis decision commitments;

iii. foresee and handle multiple *life-specific interactions* together with the resulting interaction consequences - this is possible as through concurrent synthesis, designers generate and handle both conceptual *artefact* and *life-phase system* models.

iv. be pro-actively *guided* to *which* artefact life decision commitments can be *explored* to achieve intended LCCs or to avoid/relax unintended LCCs;

v. *assess* the impact of both single and interacting synthesis decision commitments on the artefact's virtual life, by *monitoring* the relative fluctuations in performance measures of multiple life-phases;

vi. have *design process freedom* – they are still in control of the process and with its termination;

vii. *learn* about artefact LCCs *during* design, this considered beneficial especially to non-experienced designers.

Practical novelity

From the above, a key feature of the 'KC' approach is that the activity of foreseeing multiple LCCs is *integrated* with the activity of solution synthesis, which is *fundamentally* different from *first generating* a solution and *then,* at a penalty of extra time, analysing the candidate solution to reveal conflicts with artefact life issues. This allows designers to make explicit investigations of unintended artefact LCCs during synthesis, thus promoting a *'look ahead'* approach, which as argued in [Andreasen et al. 1997a], is significant for developing methods to handle the DFΣX problem. In this way, designers are supported in handling the phenomena of propagation effects at the right time during synthesis. Thus, by addressing a limitation with current means supporting $D_sF\Sigma X$ from the perspective of providence, the 'KC' approach framework promotes a novel means to $D_sF\Sigma X$. Further, of relevance to the arguments being made here is that 91% of the evaluators considered the approach useful in practice to supporting $D_sF\Sigma X$.

Significance To Domain Knowledge Modelling & Reuse

Knowing 'what' to model is significant for design research

Knowing *what* to model is a major issue for the engineering design research community [Tomiyama 1996]. Therefore, the LCC knowledge modelling frame is beneficial for developing LCC knowledge bases for implementation in tools aimed at supporting $D_sF\Sigma X$.

LCC knowledge systematization

As revealed from experimentation with FORESEE, the established LCC knowledge structuring concept (section 8.4) provides an incremental step towards the systematization of LCC knowledge by supporting a reduction in LCC knowledge duplication. This is achieved by organizing the different synthesis elements (PDE/LCPE) into *kind_of* taxonomies, with known LCC_{ni} associated to the relevant abstraction levels in the taxonomy. This permits more specific elements lower down in the taxonomy, to inherit the LCC_{ni} knowledge associated with more general elements, thus reducing the duplication of common knowledge. When new knowledge of LCCs is captured, it can be associated to the relevant taxonomic abstraction level, thereby becoming available through inheritance to elements lower down. Similarly, LCC_i inference knowledge can be structured by modelling *interacting relationships* between classes of synthesis elements, allowing it to be inherited by PDE/LCPEs lower down in the taxonomies.

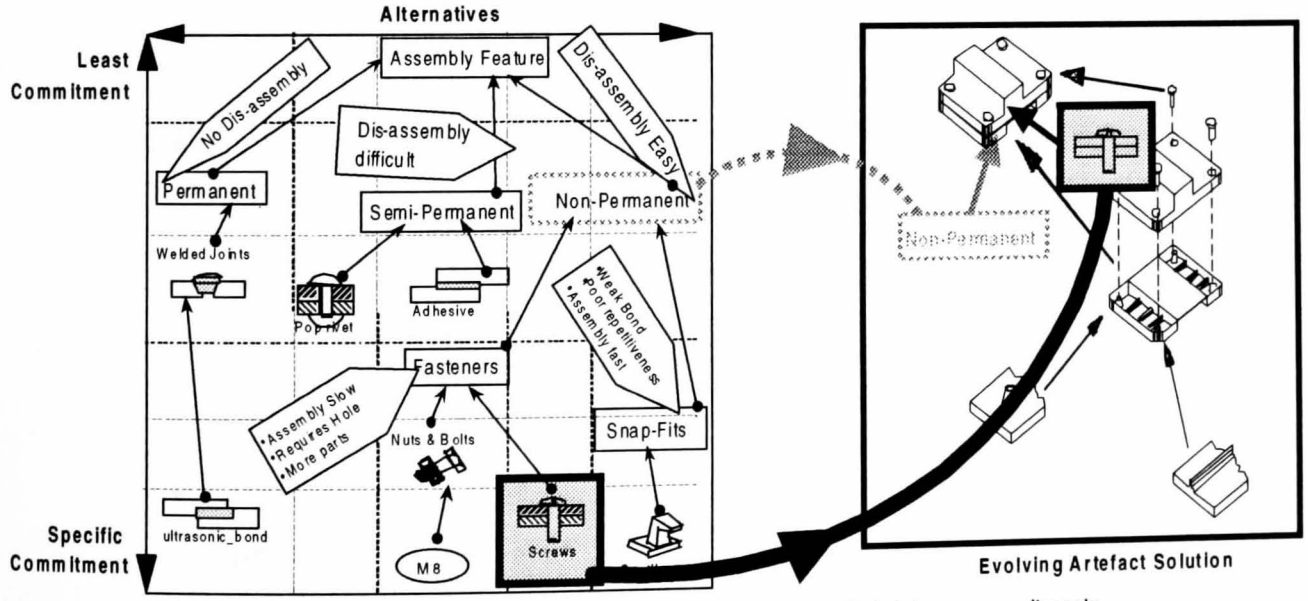


Figure 11.2 - Structure supports alternative exploration and specific/minimum commitments

Integrates 'synthesis elements' and 'LCC knowledge' reuse

Further, the structuring of PDEs/LCPEs into *kind_of* taxonomies supports domain 'past artefact life solution' reuse *combined* with 'LCC knowledge' reuse. As revealed via FORESEE, this supports:

i. LCC detection to take place with both *least* or *specific* synthesis decision commitments. This gives designers the freedom of making decisions at the desired level of commitment (Figure 11.2) and in any order;

ii. the exploration of alternatives *during* synthesis.

Supporting such 'artefact life scenario reuse' is significant to research concerning for instance *concurrent engineering* [Tomiyama 1996], the *designer's workbench* [Olesen 1992] and the *Design Coordination framework* [Andreasen et al. 1996b; O'Donnell 1997].

Significance To CAD$_s$FΣX Tool Development

A computational means to D$_s$FΣX

As reflected from the evaluation via FORESEE, the framework provides a means of how the *theory of dispositions* [Olesen 1992] can be taken up for developing CAD tools that support the *integration* of solution synthesis with provident thinking i.e. CAD$_s$FΣX tools. In this sense, FORESEE :

- explains *how* to model and operate a DFΣX approach for conceptual synthesis i.e. D$_s$FΣX, rather than employing DFΣX *later* for candidate solution analysis;

- demonstrates that KICAD tools based on the developed 'KC' approach framework potentially provide a suitable means to support the retention, processing and explicit *reuse* of LCC knowledge for proactively guiding designers in generating 'life-oriented' solutions;

- reflects that the general KICAD tool requirements (section 9.1) on which its architecture is based, are significant for the development of the designer's workbench concept [Andreasen 1992] and a design coordination system [Duffy et al. 1993] - in this case, for supporting the *generation* of life-oriented mechanical component design solutions;

- reflects that a set of *reusable* synthesis elements (PDEs) to describe artefact models and also a set of reusable elements (LCPEs) for describing life-phase system models are required when developing domain specific CAD$_s$FΣX tools. In this sense, the reuse of PDEs and LCPEs extends the concept of a *feature based 'artefact' design approach* to *early 'artefact life' solution design.*

11.2.2 Weaknesses

Needs a vast amount of distributed LCC knowledge

As LCC knowledge is dynamic in nature, the 'KC' approach requires the regular acquisition and validation of LCC knowledge. However, as LCC is distributed, its acquisition is difficult - practitioners in industry need to *capture* and *manage* LCC knowledge possessed by distributed artefact life actors. Only in this way, can valuable LCC knowledge be codified, shared and explicitly reused during solution synthesis in new design situations.

Conflicting inheritable knowledge

Inheritance through *kind_of* taxonomies as employed in the established LCC knowledge structure contributes to a reduction in knowledge duplication. However, as revealed through experimentation with FORESEE, it requires the manual *validation of* the codified knowledge because inherited LCC can be conflicting. Currently the LCC knowledge modelling frame provides no mechanism for validating knowledge captured and inherited. This can sometimes result in conflicting knowledge.

Limited mechanical artefact synthesis

Synthesis is currently limited to the *component level* and from a *constructional* viewpoint. To employ the 'KC' approach to $D_sF\Sigma X$ in industrial environments, the framework needs to be extended beyond the research boundary to support synthesis at different *artefact system levels* and from different v*iewpoints*.

Who is responsible for the solution?

Evaluators raised concerns as to whether it is the *designer* or the *KICAD system* that is responsible for a design solution. As indicated by the operational frame principles, it is the designer who should assume responsibility. As with other design tools (e.g. FEA) responsibility would be problematic only *if* a designer accepts the LCC awareness being provided by without any questioning of its relevance or accuracy.

11.3 Research Result Validation

Range of applicability

Due to the research boundary, this research focused on addressing the objectives from the perspective of *mechanical component* $D_sF\Sigma X$ and from a *constructional synthesis* viewpoint. Thus, the research result is valid in general for the life-oriented synthesis of mechanical components that involve a high degree of *synthesis element reuse* and for which distributed LCC knowledge can be captured. Also, since LCC is very domain specific, the LCC knowledge model formalism disclosed in Chapter 8 lacks soundness and completeness.

Further evaluation

For reasons given in chapter 10, proving the solution is difficult within a project of this type. To have a sounder basis on which to claim the effectiveness of the developed 'KC' approach framework to supporting $D_sF\Sigma X$, the sample of evaluators needs to be larger, the case example demonstrated needs to cover the synthesis of a more complex artefact. Also, the evaluation questions need to be re-structured to reveal potentially contradicting answers. Thus, applying the result in practice and beyond the research boundary requires further development work (see section 11.4) and further evaluation.

Validation

Nevertheless, the implementation of FORESEE (Chapter 9) and the evaluation of its effectiveness to supporting mechanical component $D_sF\Sigma X$ (Chapter 10) provided:

- a degree of evidence of the hypothesis made. Explicitly providing designers *with* knowledge of LCCs co-evolving with their solution, can *assist* them in becoming aware of a host of unintended LCCs influencing *multiple* artefact life-phases. Also, it can *motivate* them to *explore* during synthesis, the co-evolving artefact life solution and problem space and it can *guide* them to make decision commitments consciously with respect to artefact life issues.

- first hand experience of the possibility of implementing the established *'KC' approach framework* in a computational environment;

- a degree of evidence that the concept of the 'KC' approach makes a positive contribution to the proactive support of $D_sF\Sigma X$, as identified by 91% of FORESEE's evaluators.

11.4 Future Research Directions

The research result assessment presented indicates that further work is required before the 'KC' approach can be practically employed. Given therefore the broadly positive evaluation, there is considerable scope for future research. Section 11.4.1 therefore proposes further research work required to improve and extend the 'KC approach' framework beyond the current research boundary. Section 11.4.2 proposes future work to improve the practical functionality of FORESEE as a $D_sF\Sigma X$ tool. Finally, a future direction of relevance to 'design support' researchers, is a refinement to the research approach adopted, as proposed in section 11.4.3.

11.4.1 'KC' Approach Framework Improvements

Knowledge Modelling Frame

Synthesis elements beyond the research boundary

Research on LCC knowledge modelling and formalization should be extended to support the reuse of synthesis elements from other synthesis viewpoints (e.g. organ) and at different artefact systems levels (e.g. sub-assembly level). For instance, LCC_{ni} knowledge associated with a set of alternative *organs* that provide power, can be structured into a *kind_of* taxonomy (Figure 11.3).

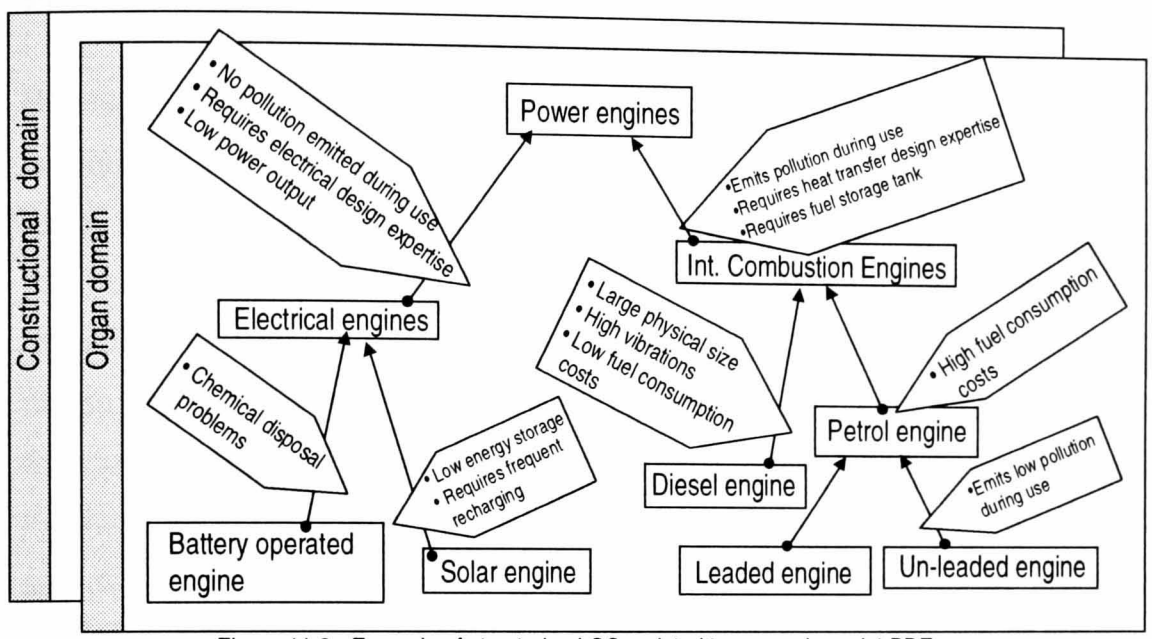


Figure 11.3 - Example of structuring LCC_{ni} related to organ viewpoint PDEs

The resultant extended set of synthesis elements would perhaps require research into the use of alternative rationalization mechanisms and not solely *kind_of* relationships.

Modelling of dynamic elements

To address an identified KICAD tool requirement (section 9.1), research into how LCC knowledge associated with synthesis elements of a dynamic nature can be modelled needs to be carried out.

Knowledge scaling

Research needs to be carried out to establish how LCC knowledge will scale up with the population of a large number of synthesis elements in order to cater for synthesis beyond the research boundary. Employing the current LCC knowledge structure for practical applications therefore requires further research concerning the maintenance of the integrity of a very large knowledge base. This would also involve capturing, codifying and validating for the extended set of synthesis elements, relevant *LCC inference knowledge* and *LCC action knowledge*.

A customizable knowledge structure

Further research is required to generate a structure with a *customizable* knowledge perspective for supporting say the exploration of LCCs associated with bought-in items (e.g. fasteners) furnished by different suppliers.

Integration with established representation standards

The LCC knowledge modelling concepts could be exploited to structure LCC knowledge as an integral part of supplier catalogues [Culley 1998; Dagger 1998] of synthesis elements (e.g. materials, standard mould parts) commonly available in electronic form. Such 'reuse' catalogues could then be used as an add-on to a KICAD tool such as FORESEE. However, to embed LCC knowledge in such electronic catalogues requires research into integrating the LCC knowledge structure, with those of existing computer representation standards (e.g. ISO 13584) of such *parts*[1] library data.

Performance measure research

Research to *benchmark* the relative fluctuations in the performance measures is required. For example on a scale of 10, should the formation of a weld line result in a fabrication cost increase of say 3 units or 8 units? Similarly, what influence in user perceived quality would such a weld line have? Also, as no distinction is currently made between say, one unit of time in the realization phase and one unit of time in the use phase, research is required concerning the *normalization* of performance metrics (e.g. time) for different life-phases. Further, the *virtual artefact life behaviour* is currently modelled (Chapter 8) by summing the different values of a metric (e.g. time) for each life-phase. Although suitable for comparing the *relative* scores of alternative conceptual solutions, it assumes that processes [P] forming part of a phase are executed sequentially, which in reality is not always the case. For calculating *absolute* estimates, further research to improve the modelling of artefact life behaviour is thus necessary.

Modelling LCC certainty

As LCCs are currently modelled, the commitment of the relevant synthesis decision(s) results in the occurrence of the LCC with 100% certainty. In reality, such an occurrence has an associated probability. Research is therefore required to *establish* and *model* LCC certainty factors.

Operational Frame

Group concurrent synthesis decision making

Due to the specified research boundary, the operational frame was developed for *individual designers* working on a *component*. If the concept of the 'KC' approach framework is to be extended to team based, 'product level' synthesis, further research is required into the operating principles supporting group concurrent synthesis decision making (Figure 11.4).

[1] In ISO13584, a 'part' is a collection of material or functional elements that are equivalent, at a defined level of abstraction and which is capable of being characterised by an identifier

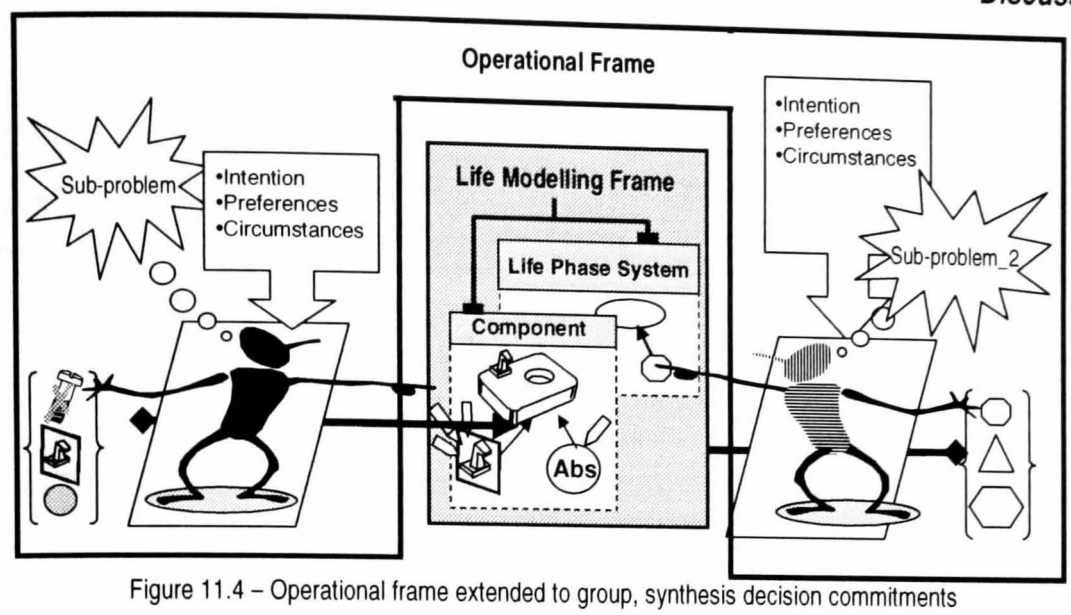


Figure 11.4 – Operational frame extended to group, synthesis decision commitments

Artefact Life Modelling Frame

Different system levels, viewpoints and domains

Applying the 'KC approach' to artefact domains beyond the research boundary (e.g. electronic) requires further research to establish how to describe such early *artefact life* models. Also, further work is required to extend *artefact life* solution modelling to other artefact system levels (e.g. sub-assembly) and from other viewpoints (e.g. functional) as compositional solution modelling may not be suitable. Similarly, life-phase system solution modelling needs to be extended to for example handle the transformation of artefact structures involving components made from different materials.

Process sequence modelling

Components can encounter *'n' (n > 1)* technical processes during a life-phase. To enable designers to explore the influence on performance measures of alternative technical process *sequence plans*, research is required to extend *artefact life* modelling for describing such process sequence plans.

11.4.2 FORESEE Improvements

FORESEE needs to cater for improvements to the 'KC' approach framework proposed in section 11.4.1. Further, for utilization in an industrial context, its functionality needs to be enhanced through the implementation of all the system requirements identified in Chapter 9 and to cater for feedback provided by the evaluators. The following research is thus proposed.

Improved solution display and manipulation

To support designers in rapidly comparing performance measures of alternative evolving solutions, FORESEE needs to be extended to support the *simultaneous manipulation* of two or more evolving models (Figure 11.5). Research is also required to improve the way in which the evolving *artefact life solution* is displayed. A possibility is a hierarchy of graphical symbols, with symbols that also reflect the level of commitment (e.g. an opening).

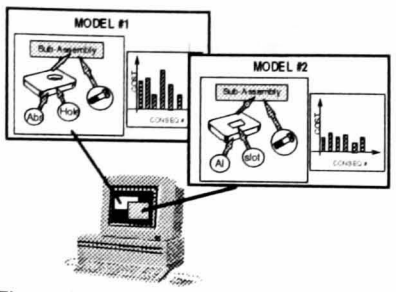


Figure 11.5 – Simultaneous manipulation of alternative models

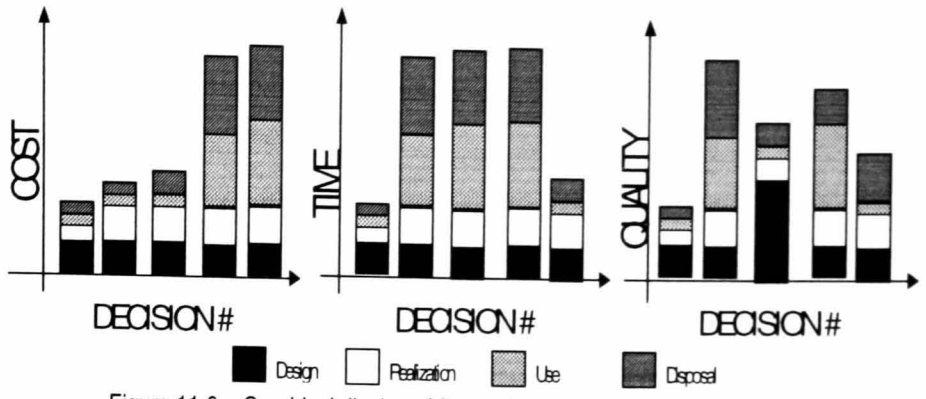


Figure 11.6 – Graphical display of fluctuations in performance measures

Improvements to LCCs handling

Besides guidance as to *which* commitments can be explored in order to avoid/relax a detected LCC, evaluators stated that ideally designers should be *guided* to what alternative *concretization* or *detailing* commitments can be made. The way in which LCC are reported also needs to be improved. Suggestions made by evaluators include: the automatic opening of the consequence browser when LCCs are detected; using a different colour for LCC entries in the browser resulting from the latest decision commitment; sorting LCCs in a reverse order, showing the latest LCC first in the list. In addition, the display of *multi-x behaviour* monitoring needs to be improved e.g. using a graphical display (Figure 11.6). Another improvement suggested by some evaluators, is the inclusion of a mechanism allowing designers to specify the minimum amount of a percentage change (%) in performance measures for which they would want to be informed. Although possible, this suggestion needs to be taken up carefully since a LCC causing a minor fluctuation could propagate LCCs that result in major influences.

Improved truth maintenance capability

To utilize FORESEE in practice, further work is required to extend the truth maintenance mechanism developed to cater for situations where more than one *similar* LCC exists. As implemented for evaluation purposes, FORESEE's maintenance mechanism functions for multiple, *different* LCCs. However, it currently only maintains one LCC (e.g. a weld line) if *n (n >1) similar* LCC are present at any one time.

Improved LCC knowledge management capability

For practical purposes, FORESEE's knowledge manager needs to be extended to support distributed life-actors *inputting* any new LCC knowledge they encounter in the *right modelling format* and in *validating* the input knowledge (Figure 11.7). The LCC knowledge manager also needs to be improved to allow users to simply click on a graphical symbol representing say a class or instance (Figure 11.8), to *acquire* details of the current content and to make *modifications* on-line without the need of system developers.

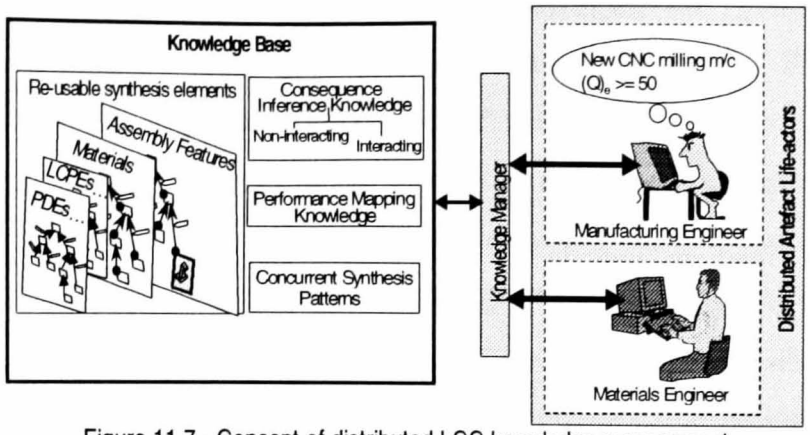


Figure 11.7 - Concept of distributed LCC knowledge management

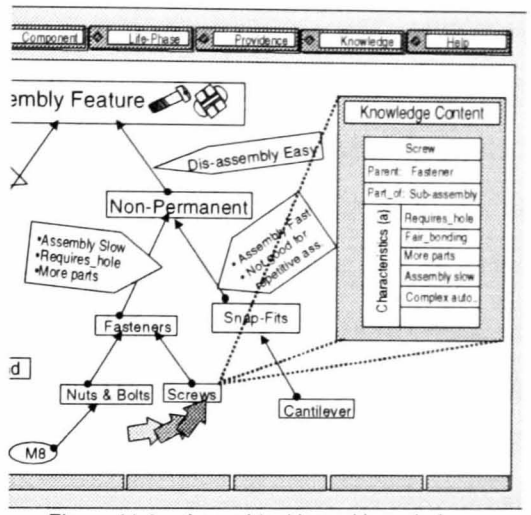


Figure 11.8 – A graphical based knowledge management interface

On-line LCC$_i$ knowledge management

Research into supporting on-line LCC$_i$ knowledge management will contribute a significant step towards the practical application of FORESEE. Users should be able to dynamically relate synthesis elements with LCC and to define interacting relationships without the need of resorting to system developers. As wxCLIPS is not intended for dynamic constructs, FORESEE employs a temporary measure by which rules are re-loaded *after* the knowledge manager adds new classes or instances. With this measure, rule pattern matching works but it is computationally expensive. An alternative programming environment supporting the dynamic creation of constructs should therefore be considered for future implementation work.

Improved user functionality

FORESEE's user interface should be improved by utilizing graphical icons to provide faster command input and the avoidance of typing errors. Another improvement is to provide a means by which user-defined synthesis elements are automatically added to the pull-down menu. FORESEE does not support all design activities and stages. Thus, a significant improvement would be the inclusion of a facility to export the solution into a neutral format (e.g. STEP) to allow other computer based tools to read the 'artefact life' solution model.

11.4.3 'Design Support' Research Approach Refinement

Refinement to the computational means development framework

Based on the outcome of this Ph.D. research, it can be stated that the *research methodology* and the *computational means development framework* adopted (see Chapter 1), collectively provide a suitable planned research approach to executing 'design support' research. As the work in this dissertation reflects, this results in an increased understanding, through *phenomena modelling*, of a design problem encountered in *reality*. By exploiting the explanations provided, a *knowledge model* can be generated

and utilized for developing a computer based tool to address the relevant design problem. However, for future research, the *computational means development framework* (Figure 1.3) can be refined by replacing the *knowledge model* with a *framework* (see Figure 11.9) made up of: (i) a *knowledge modelling frame*; (ii) an *operational frame* consisting of a model of the 'working environment' and a description of the 'mode of operation' a designer should adopt for utilizing the established knowledge model and (iii) a *solution modelling frame* explaining what solution descriptions it can be used for. It is such a framework that collectively contributes a formal explanation of how a design phenomena-specific *knowledge model* can be generated and operated to support the problem reality encountered *during* design, this allowing relevant computer-based design support tools to be developed.

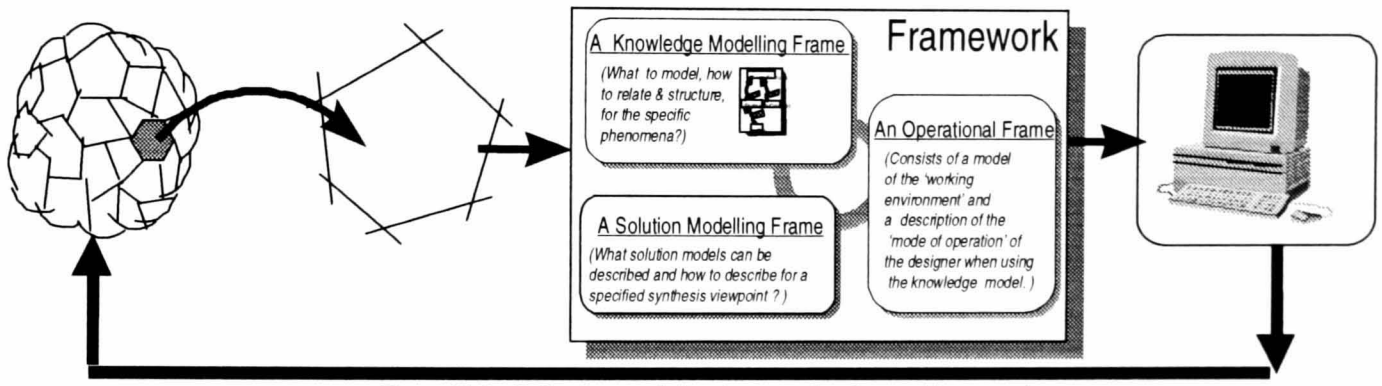


Figure 11.9 - Refined computational means development framework

11.5 Chapter Conclusions

This chapter disclosed in section 11.1, the results obtained during this research, these being:

- *a 'KC' approach framework to support $D_sF\Sigma X$;*
- *a phenomena model explaining 'how' LCCs are generated;*
- *a $D_sF\Sigma X$ computational model and*
- *an incremental understanding to the $D_sF\Sigma X$ problem.*

Section 11.2 discussed the strengths and weaknesses of what has been achieved, this outlining the significance of the overall 'KC' approach framework to*: supporting $D_sF\Sigma X$; domain knowledge modelling and reuse* and *$CAD_sF\Sigma X$ tool development.* The validation of the research result has been discussed in section 11.3. Section 11.4 discussed avenues for future research work concerning both the 'KC' approach framework and the functionality of FORESEE, in order to improve and extend the 'KC' approach to $D_sF\Sigma X$. The next and final chapter presents a number of conclusions arising from this research.

chapter 12

12.0 Conclusions

Scope

Based on the research work carried out, the following conclusions can be made about the problem of $D_sF\Sigma X$, the developed *'Knowledge of life-cycle Consequences ('KC') approach framework to* $D_sF\Sigma X$ and *future research work* required to develop the results further.

A phenomena – consequences propagate across multiple life-phases

$\underline{D_sF\Sigma X}$

Mechanical artefact synthesis decision commitments can influence the *interaction* that takes place between an *artefact* and *life-phase systems* encountered. This interaction results in *unintended* consequences that *propagate* across *multiple* life-phases i.e. *life-cycle consequences* (LCCs). A problem is that due to the sequence of life-phases, knowledge of such LCCs is (i) generated late, *after* decisions have been committed and (ii) *distributed* amongst different artefact life actors. Nevertheless, due to this propagation effect phenomena, the designers' responsibility covers *all* life-phases. Thus designers cannot take a *narrow* and *segmented view* to their evolving design – rather they need to adopt a $D_sF\Sigma X$ approach.

General requirements for a $D_sF\Sigma X$ *approach*

Key requirements for a $D_sF\Sigma X$ approach are that designers need to *collectively* engage in: (i) *provident thinking* during early design in order to acquire knowledge of artefact life interactions and their propagations; (ii) *'artefact life' solution exploration* in order to foresee such interactions and their co-evolving LCCs; (iii) the *concurrent synthesis* of artefact and life-phase system models.

Current state of 'providence' means

Current means supporting life-oriented design lack to pro-actively aid designers in handling the phenomena of LCCs *during synthesis* as they exhibit one or more of the following limitations. Providence:

(i) takes place *late* during candidate solution analysis;

(ii) covers a *narrow* and *segmented* rather than multiple, life-span view and

(iii) is *generic* and not 'life-specific', as 'life-phase' modelling is not supported.

Due to this state of current means and human mental processing limitations, designers have a difficulty in considering LCCs co-evolving with their solution. This highlights that designers can benefit from computational, LCC knowledge intensive means supporting them in foreseeing LCCs *during* synthesis.

The developed computational 'KC' approach framework to $D_sF\Sigma X$

The 'KC' approach framework founded on a LCC phenomena model

An LCC phenomena model has established that LCCs are generated in two fundamentally different ways i.e. by *individual* or *interacting* synthesis decision commitments. This model highlights that concurrent 'artefact' and 'life-phase system' synthesis is a necessity if interacting LCCs are to be taken into consideration during design. It also provides a foundation of what to model and how to utilize LCC knowledge for supporting $D_sF\Sigma X$. Based on these understandings, a novel framework for a *'Knowledge of life-cycle Consequences ('KC') approach* to $D_sF\Sigma X$ was developed, consisting of: a *'LCC knowledge modelling frame'* which presents a formalism of 'what' elements to acquire and model for an application domain, together with how to structure the established relationships into *LCC inference* and *LCC action* knowledge; an *'artefact life modelling'* frame which provides a formalism for describing 'artefact life' compositional models that support the inference of LCCs; and the *'operational frame'* which discloses a model of the '$D_sF\Sigma X$ working environment' and a description of the designer's 'mode of operation' so that they can systematically utilize modelled LCC to reveal LCCs co-evolving *during* synthesis, in order to promote 'artefact life' exploration and hence the *generation* of a life-oriented solution.

FORESEE – a KICAD prototype based on the 'KC' approach framework

Through the establishment of an LCC knowledge model for the mechanical component domain and the identification of (i) a set of general system requirements (ii) an architecture and (iii) knowledge codification schemes, the framework was realized as a *KICAD* prototype, 'FORESEE', for the thermoplastic component domain. This implementation demonstrates that the 'KC' approach framework can be realized using existing computer technology and that KICAD tools provide a suitable means for retaining and explicitly reusing captured LCC knowledge. FORESEE allows designers to generate (i) an early component compositional mode and (ii) a number of early life-phase system compositional models. It also provides a list of LCCs associated with the current solution state, an associated set of *multi-X* relative performance measure values, explanations of LCCs and guidance to their avoidance, a history of decision commitments made and their associated impact on performance measures.

Strengths & limitations of the established 'KC' approach

The analysis of FORESEE's evaluation provides evidence that a KICAD tool based upon the 'KC approach' framework enhances the designers' capabilities in *generating* 'life-oriented' mechanical component solutions, since designers:

a) become explicitly aware of co-evolving LCCs, propagating across *multiple* life-phases;

b) can foresee and handle multiple *life-specific interactions* together with the resulting interaction consequences;

c) are pro-actively *guided* to *which* artefact life decision commitments to explore to achieve intended LCCs or to avoid/relax unintended LCCs and

d) can *assess* the impact of decision commitments on the artefact's life.

It can be therefore concluded that the 'KC' approach integrates synthesis with the activity of foreseeing *multiple* artefact life issues. This is fundamentally different from first generating a solution and then, with a penalty of extra time, analysing a candidate solution for conflicts with artefact life issues.

Future Research Work

Work required to develop the results further

Due to the positive evaluation results and the fact that 91% of the evaluators considered the approach to be useful in practice, further research is merited to address weaknesses of both the *'KC' approach framework* and *FORESEE*. Key areas of useful research work were identified as:

a) extending the 'KC approach' framework to support $D_sF\Sigma X$:
   - *at different artefact system levels;*
   - *from different synthesis viewpoints* and
   - *for non-mechanical artefact domains;*

b) investigating the scaling up of LCC knowledge arising from a very large knowledge base;

c) improving the functionality of FORESEE by completing the implementation of the identified system requirements (Table 9.1) and addressing the recommendations made in section 11.4.2 so as to amplify the effectiveness of the 'KC approach' to $D_sF\Sigma X$.

References

Albano, D. and N. P. Suh (1994). "Axiomatic design and concurrent engineering", *Computer-Aided Design,* Vol.26, No.7, pp.499-504.

Allen, A. J. and K. G. Swift (1990). "Manufacturing Process Selection and Costing", *Journal of Engineering Manufacture*, Vol.204, No.B2., pp.143-148.

Allen, R. E. (1990). *"The Concise Oxford Dictionary",* Oxford University Press Oxford.

Alting, L. and J. B. Legarth (1995) "Life Cycle Engineering and Design", *Annals of the CIRP*, Vol.44, No.2., pp.569-580.

Andersson, P (1994). "Early Design Phases and Their Role in Designing for Quality", *Journal of Engineering Design*, Vol.5, No.5., pp.283-298.

Andreasen, M. M. and L. Hein (1987) *"Integrated Product Development",* IFS Publications Ltd. Springer-Verlag London.

Andreasen, M. M. and J. Olesen (1990). "The Concept of Dispositions", *Journal of Engineering Design,* Vol.1, No.1., pp.17-36.

Andreasen, M. M. (1991). "Design methodology", *Seminar on Design Theory and its Application,* Trondheim, Norway, The Norwegian Academy of Technical Sciences.

Andreasen, M. M. (1991a). "The Theory of Domains", *Workshop on Understanding Function and Function to Form Evolution*, Cambridge University, UK.

Andreasen, M. M. (1992). "Designing on a 'Designers Workbench (DWB)'", *9th WDK Workshop*, Rigi.

Andreasen, M. M. and J. Olesen (1993). "Consensus statements", *WDK Workshop on Design For X*, Technical University of Denmark, Lygby, Denmark.

Andreasen, M. M. and S. Storen (1993a). "Product Development Based on 'Know Your Product' - Philosophy", *4th Symposium on Fertigugnsgerechtes Konstruieren, Egloffstein.*

Andreasen, M. M., C. T. Hansen, et al. (1996). "The Structuring of Products and Product Programmes", *2nd WDK Workshop on Product Structuring*, Delft University, The Netherlands.

Andreasen, M. M. and N. H. Mortensen (1996a). "The Nature of Design Features", *7. Symposium "Fertigungsgerechtes Konstruieren",* Shnaittach.

Andreasen, M. M., A. H. B. Duffy, et al. (1996b). "The Design Co-ordination Framework: key elements for effective product development", *The Design Productivity Debate.* A. H. B. Duffy. London, Springer-Verlag: pp. 151-172.

Andreasen, M. M., C. T. Hansen, et al. (1997). "On The Identification of Product Structure Laws", *3rd WDK Workshop on Product Structuring*, Delft University, The Netherlands.

Andreasen, M. M. and N. H. Mortensen (1997a). "Basic Thinking Patterns and Working Methods for multiple DFX", *8. Symposium "Fertigungsgerechtes Konstruieren",* Shnaittach.

Araujo, C. S. D. and A. H. B. Duffy (1996). "Product Development Tools", *III International Congress of Project Engineering*, Barcelona.

Askin, R. G. and M. Sodhi (1994). "Organization of Teams in Concurrent Engineering", *Handbook of Design, Manufacturing and Automation.* R. Dorf and A. Kusiak. New York, John Wiley & Sons Inc.: pp. 85-105.

Bacon, J. P. and R. G. Sindt (1997). "HTML Authoring Using Netscape Gold", *Understanding and Using Netscape Navigator Browsing and Web Authoring.* Minneapolis, West Publishing Company: pp.140-166.

Bahler, D., C. Dupont, et al. (1994). "An Axiomatic Approach that supports negotiated resolution of design conflicts in concurrent engineering", *Artificial Intelligence in Design '94*, Lausanne, Switzerland, J.S. Gero et, al. Kluwer Academic Publishers: pp.363-379.

Bahler, D., C. Dupont, et al. (1994). "Mediating Conflict in Concurrent Engineering with a Protocol Based on Utility", *Concurrent Engineering Research and Applications*, Vol.2, pp.197-207.

Bahrami, A. and C. H. Dagli (1993). "Models of design processes", *Concurrent Engineering - Contemporary Issues and Modern Design Tools*. H. R. Parsaei and W. G. Sullivan. London, Chapman & Hall: pp. 113-126.

Balendra, R., H. Ou, et al. (1995). "FE Analysis of Pre-Stressed Press Frames", *3rd International Conference, Computer Integrated Manufacturing*, Singapore, World Scientific:Vol.1. pp.583-589.

Bartholomew, O. N., K. Tzong-Shyan, et al. (1991). "Feature reasoning for sheet metal components", *International Journal of Prod. Research*, Vol.29, No.9., pp.1867-1896.

Beitz, W. (1993). "Designing for ease of recycling - General approach and Industrial Application", *International Conference on Engineering Design*, The Hague, WDK:Vol.2 pp.731-738.

Belson, D. (1994). " Concurrent Engineering", *Handbook of Design, Manufacture and Automation.* R. C. Dorf and A. Kusiak. USA, John Wiley & Sons Inc.: pp. 25-34.

Berliner, C. and J. A. Brimson (1988). *"Cost management for Today's Advanced manufacturing: The CAM-I conceptual design",* Harvard Business School Press Boston.

Bjork, E. L. and R. A. Bjork (1996). *"Memory",* Academic Press California.

Blessing, L. M. (1994). "A Process-Based Approach To Computer Supported Engineering Design", *Ph.D.,* University of Twente, Enschede: The Netherlands.

Blessing, L. M. (1993). "A Process-Based Approach To Computer Supported Engineering Design", *International Conference on Engineering Design*, The Hague, Heurista:Vol.3 pp.1393-1400.

Boks, C. B. and E. Tempelan. (1997). "Delphi Forecasting of Recycling Technology to Support Design For End-of-life of Consumer Electronic Products", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology. Vol.1 pp.343-346.

Bonello, A. (1997). "Design exploration of assembly related life cycle consequences", *B.Eng.,* University of Malta, Msida.

Bonfield, T. and T. Lawlor-Wright (1997). "Towards a Multiple Perspective Design Advisory System For Printed Circuit Board Assemblies", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology. Vol. 2 p.p. 541-546.

Boothroyd, G. and P. Dewhurst (1991). "Product design for manufacture and assembly", *Design for Manufacture, Strategies, Principles and Techniques*. J. Corbett, M. Dooner, J. Meleka and C. Pym. Wokingham, Addison-Wesley Publishing Company: pp 165-173.

Borg, J. and K. J. MacCallum (1996). "Structuring Knowledge of Life-Cycle Consequences for Supporting Concurrent Design Exploration", *IFIP TC5 WG5.2, International Workshop on Knowledge Intensive CAD*, Carnegie Mellon University, Pittsburgh, PA, USA, 1996, Chapman & Hall:pp.208-223.

Borg, J.C. and X. T. Yan (1998)., "Design Decision Consequences: Key to 'Design For Multi-X ' Support", *Proceedings, 2nd International Symposium 'Tools and Methods For Concurrent Engineering'*, Manchester, UK: pp.169-184.

Borg, J. P (1996). "The effect of product and mould tool design on cost", *B.Eng.*, University of Malta, Msida, Malta.

Boston, O., A.W.Court, S. J. Culley, et al. (1996)., "Design Information Issues in New Product Development", *The Design Productivity Debate*. A. H. B. Duffy. London, Springer-Verlag: pp 231-254.

Bowen, J. (1991). "Aspects of Constraint Processing, (Technical Note)", *Artificial Intelligence in Engineering*, Vol.6, No.2., pp.100.

Bowen, J. (1995). "Using Dependency Records to generate Design Coordination advice in a Constraint-Based Approach to Concurrent Engineering", *Co-operation In Manufacturing:CIM AT WORK*, Kaatsheuvel, The Netherlands, CIMMOD/CIMDEV: pp.77-87.

Bowen, J., P. O'Grady, et al. (1990). "A constraint programming language for Life-Cycle Engineering", *Artificial Intelligence in Engineering*, Vol.5, No.4., pp206-220.

Bralla, G. J. (1996). *"Design for Excellence"*, McGraw-Hill, Inc. New York.

Brown, D. (1996). "Which way to KIC?", *IFIP TC5 WG5.2, International Workshop on Knowledge Intensive CAD*, Carnegie Mellon University, Pittsburgh, PA, USA, Chapman & Hall:pp.291-295.

Chakrabarti, A. and T. P. Bligh (1991). "Towards a decision-support framework for mechanical conceptual design", *International Conference On Engineering Design* Zurich, WDK:pp.384-389.

Chal, J. and H. Linde (1997). "Product Life thinking in product development", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol.3 pp.669-672.

Changchien, S. W. and L. Lin (1996). "A Knowledge-based design critique system for manufacture and assembly of rotational parts in concurrent engineering", *Computers in Industry*, Vol.32, No.2., pp.117-140.

Choy, C. M. and R. Balendra (1995). "The Influence of Tool Geometry On Blanking Characteristics", *3rd International Conference, Computer Integrated Manufacturing*, Singapore, World Scientific:Vol.1 pp.576-582.

Clausing, D. P. (1993). "World-Class Concurrent Engineering", *Concurrent Engineering Tools and Technologies for Mechanical System Design*. E. J. Haug. USA, Springer-Verlag: pp. 3-40.

Clausing, D.P. (1994). *"Total Quality Development - A Step-by-Step Guide to World-Class Concurrent Engineering"*, ASME Press, New York.

Cleetus, K. J. (1992). "Definition of Concurrent Engineering". *Report CERC-TR-RN-92-003*, Concurrent Engineering Research Centre, West Virginia University.

Cleetus, K. J. (1993). "Virtual Team Framework and Support Technology", *Concurrent Engineering Tools and Technologies for Mechanical System Design*. E. J. Haug. USA, Springer-Verlag: pp. 41-73.

CLIPS (1997). *"CLIPS Reference Manual, Volume 1 - Basic Programming Guide"*, Software Technology Branch, Lyndon B.Johnson Space Center.

Cook, R. D. (1995). *"Finite Element Modeling For Stress Analysis"*, John Wiley & Sons, Inc. New York.

Coyne, R. D., M. A. Rosenman, et al. (1989). *"Knowledge-based design systems"*, Addison-Wesley Reading, Massachusetts.

Cross, N. (1994). "Engineering Design Methods", *Engineering Design Methods*. Chichester, England, John Wiley & Sons Ltd.: pp.33-47.

Culley, S. (1998). "Designing with standard parts", *Engineering Designer*, September/ October, pp.8-11.

Cutajar, A. (1997). "Component Material and fabrication process compatability design considerations", *B.Eng.*, University of Malta, Msida, Malta.

Dagger, B. (1998). “New CD-ROM design aids”, *Engineering Designer*, September/ October, pp.22-23.

Dalgleish, G. F., K. G. Swift, et al. (1998). "Computer Support For Proactive DFA", *2nd International Symposium 'Tools and Methods For Concurrent Engineering'*, Manchester, UK, pp. 313-320.

Dan, P. (1992). “Concurrent Engineering at Martin Marietta Corporation”, *Concurrent Engineering Research in Review*, Vol.4, Special Issue, pp.6-7.

Dooley, K. J. (1994). "Quality Engineering", *Handbook of Design, Manufacture and Automation.* R. C. Dorf and A. Kusiak. USA, John Wiley & Sons Inc.: pp. 619-648.

Douglas, R. E. and D. C. Brown (1993). "Concurrent Accumulation of Knowledge: a view of concurrent engineering", *Concurrent Engineering.* H. R. Parsaei and W. G. Sullivan. London, Chapman & Hall: pp. 402-412.

Duckworth, A. P., R. W. Baines, et al. (1998). "An Eco-Design Framework For Small and Medium Sized Manufacturing Enterprises", *2nd International Symposium 'Tools and Methods For Concurrent Engineering'*, Manchester, UK, pp. 132-141.

Duffy, A. H. B. (1997), “The 'What' and 'How' of Learning in Design”, *IEEE Expert*, May/June, pp.71-76.

Duffy, A. H. B. and F. J. O'Donnell (1998). "A Design Research Approach", *Workshop on Research Methods in Artificial Intelligence in Design (AID'98),* Lisbon, Portugal.

Duffy, A. H. B. and M. M. Andreasen (1995). "Enhancing the Evolution of Design Science", *Proceedings, International Conference on Engineering Design*, Praha, pp.29-35.

Duffy, A. H. B., M. M. Andreasen, et al. (1993). “Design Co-ordination for Concurrent Engineering”, *Journal of Engineering Design*, Vol.4, No.4., pp.251-265.

Duffy, A. H. B., A. Persidis, et al. (1996). “NODES: a numerical and object based modelling system for conceptual engineering design”, *Knowledge-Based Systems*, Vol.9, pp.183-206.

Duffy, A. H. B., S. M. Duffy, et al. (1995). "Using Design Complexities in validating the Design Coordination Framework", *Co-operation In Manufacturing:CIM AT WORK*, Kaatsheuvel, The Netherlands, CIMMOD/CIMDEV. pp.16-40.

DuPont (1992). *"Design Handbook For Du Pont Engineering Polymers".* Wilmington, Du Pont.

Dym, C. L. (1994). *"Engineering Design - A Synthesis of Views",* Cambridge University Press.

Eastwood, M. A. (1995). "Implementing Mass Customization", *3rd International Conference, Computer Integrated Manufacturing,* Singapore, World Scientific:Vol.3, pp.12-16.

Eccles, W. (1994). “Quality Function Deployment”, *Engineering Designer*, January /February, pp.8-11.

Edwards, K. L. (1997). “A Perspective on Teamworking in Engineering Design”, Eng*ineering Designer*, September, pp.14-15.

Ellis, H. C. and R. R. Hunt. (1989). *"Fundamentals of Human Memory and Cognition"*, 4th Edition, Wm.C.Brown Company Publishers Iowa.

EMS (1996). *"Technical Data Sheet - Grivory GV"* . Domat, EMS-CHEMIE AG.

Eubanks, C. F. and K. Ishii (1993). “ AI methods for life-cycle serviceability design of mechanical systems”, *Artificial Intelligence in Engineering*, Vol.8, No.2., pp.127-140.

Fabricius, F. (1994). *"Design For Manufacture".* Lyngby, Denmark, Institute for Product Development (IPU).

Farish, M. (1992). *"New product development - The route to improved performance".* London, The Design Council.

Feng, C.-X. and A. Kusiak (1995). "Constraint-based design of Parts", *Computer-Aided Design*, Vol.27, No.5, pp.343-352.

Feng, C.-X., A. Kusiak, et al. (1996). "Cost evaluation in design with form features", *Computer-Aided Design*, Vol.28, No.11, pp.879-885.

Finger, S. and J. R. Dixon (1989). "A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Processes", *Research in Engineering Design*, Vol.1, No.1 pp.51-67.

Finger, S. and J. R. Dixon (1989a). "A Review of Research in Mechanical Engineering Design. Part II Representations, Analysis and Design for the Life Cycle", *Research in Engineering Design*, Vol.1, No.2 pp.121-137.

Fox, J. (1993). *"Quality Through Design"*, McGraw-Hill Book Company, London.

Franze, H. A. (1997). "New Life-Cycle Management Tools in the BMW Product Development Process", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol.1, pp.23-35.

French, M. J. (1985). *"Conceptual Design For Engineers"*, Design Council London.

Fuh, J. Y. H., C.-H. Chang, et al. (1996). "The Development of an Integrated and Intelligent CAD/CAPP/CAFP environment using logic-based reasoning", *Computer Aided Design*, Vol.28, No.3, pp.217-232.

Gadient, A. J., L. E. Hines, et al. (1997). "Agility through information sharing: Results achieved in a production setting", *Concurrent Engineering: Research and Applications*, Vol.5, No.2, pp.101-111.

Galea, A. (1997). "Component surface finish life cycle design considerations", *B.Eng.*, University of Malta, Msida, Malta.

Gardner, S. and Sheldon (1993). "Consensus Statement no.10.8.", *WDK Workshop on Design For X*, Technical University of Denmark, Lyngby, Denmark.

Geiger, T. S. and D. M. Dilts (1996). "Automated design-to-cost: integrating costing into the design decision", *Computer Aided Design*, Vol.28, No.6/7, pp.423-438.

Giarratano, J. C. and G. D. Riley (1994). *"Expert Systems: Principles and Programming"*, PWS, USA.

Grech, T. (1996). "Generating environmental design guidelines for injection moulded thermoplastic features", *B.Eng.*, University of Malta, Msida, Malta.

Groover, M. P. (1996). *"Fundamentals of Modern Manufacturing - Materials, Processes and Systems"*, Prentice-Hall Inc. New Jersey.

Grote, K.-H., J. L. Miller, et al. (1995). "Solid Freeform Manufacturing as Part of the Integrated Product Development Process", *3rd International Conference, Computer Integrated Manufacturing*, Singapore, World Scientific:Vol.1, pp.347-350.

Guan, X. and K. J. MacCallum (1995). "Handling of positional information in a system for supporting early geometric design", *AI System Support for Conceptual Design*, Ambleside, UK, Springer-Verlag:pp.55-70.

Gupta, S. K. and S. N. Dana (1995). "Systematic Approach To Analysing The Manufacturability of Machined Parts", *Computer Aided Design*, Vol.27, No.5, pp.323-342.

Gupta, S. K., C. R. William, et al. (1997). "Automated Manufacturability Analysis: A Survey", *Research in Engineering Design*, Vol.9, pp.168-190.

Hague, M. J., A. Taleb-Bendiab, et al. (1995). "An adaptive machine learning system for computer supported conceptual engineering design", *AI System Support for Conceptual Design*, Lancaster, UK, Springer-Verlag: pp.1-16.

Hales, C. (1993). *"Managing engineering design"*, Longman Scientific & Technical, Essex.

Hansen, C. T. (1997). "Towards a tool for computer supported structuring of products", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology.

Haslehurst, M. (1985). *"Manufacturing Technology"*, Hodder and Stoughton London.

Healey, J. (1994). "Failure Mode and Effects Analysis", *Engineering Designer*,. January /February, pp.4-7.

Hird, G. (1993). "Experiences in LUCAS with DFX", *WDK Workshop on Design For X*, Technical University of Denmark, Lyngby, Denmark.

Holtzman, S. (1988). *"Intelligent decision systems"*, Addison-Wesley Reading Mass.

Huang, G. Q. and K. L. Mak (1997). "Developing a Generic Design For X Shell", *Journal of Engineering Design*, Vol.8, No.3, pp. 251-260.

Hubka, V. (1985). "Attempts and possibilities for Rationalisation of Engineering Design", *Design and Synthesis*. H. Yoshikawa, Elsevier Science & Publishers B.V.:pp.133-138.

Hubka, V. and E. Eder (1988). *"Theory of technical systems: A total concept theory for engineering design"*, Springer-Verlag Berlin/Heidelberg.

Hubka, V., M. M. Andreasen, et al. (1988). *"Practical Studies in Systematic Design"*, Butterworths London.

Huh, Y.-J. and S.-g. Kim (1991). "A knowledge-based CAD system for concurrent product design in injection moulding", *International Journal Computer Integrated Manufacturing*, Vol.4, No.4, pp.209-218.

Ishii, K. (1991). "Role of Computerized Compatibility Analysis in Simultaneous Engineering", *International Journal of Systems Automation: Research and Applications (SARA)*, Vol.1, pp.325-345.

Ishii, K. (1995). "Life-Cycle Engineering Design", *Transactions of the ASME*, Vol.117, pp.42-47.

Jacobs, S. and S. Kethers (1994). "Improving Communication and Decision Making within Quality Function Deployment", *Concurrent Engineering Research & Applications*, Pittsburgh PA, CERA Institute:pp.203-213.

Jared, G. E. M., M. G. Limage, et al. (1994). "Geometric reasoning and design for manufacture", *Computer-Aided Design*, Vol.26, No.7, pp.528-536.

Jeang, A. and D. R. Falkenburg (1995). "Interactive Multiple Criteria Decision Making For Product Development", *3rd International Conference, Computer Integrated Manufacturing*, Singapore, World Scientific:Vol. 1, pp.253-260.

Jiafu, B., S. Pengfei, et al. (1995). "Numerical Simulation of Melt Flow Behaviour In Injection Moulding", *3rd International Conference, Computer Integrated Manufacturing*, Singapore, World Scientific:Vol.1, pp.569-575.

Jin, Y. and S. Lu (1998). "Toward a Better Understanding of Engineering Design Models", *Universal Design Theory*, Karlsruhe, Germany, Shaker Verlag:pp.73-84.

Jo, H. H., H. R. Parsaei, et al. (1993). "Principles of concurrent engineering", *Concurrent Engineering - Contemporary Issues and Modern Design Tools*. H. R. Parsaei and W. G. Sullivan. London, Chapman & Hall: pp. 3-23.

Joshi, S. P. (1991). "Decision-Making In Preliminary Engineering Design", *Artificial Intelligence in Engineering Design and Manufacture*, Vol.5, No.1, pp.21-30.

Karinthi, R., V. Jaganathan, et al. (1992). "Promoting Concurrent Engineering Through Information Sharing", *Engineering Database Program, ASME Winter Annual Meeting*, USA.

Kerr, S. M. (1992). "Human memory research developments and its influence on intelligent computer aided design". *Report CADC/R/92-04*, Glasgow, CAD Centre, Department of Design, Manufacture & Engineering Management, University of Strathclyde.

Kerr,S.M. and A.H.B. Duffy (1992) “Automated Rationalisation of Experiential Design Knowledge To Support Dynamic Memory”. *Internal report CADC/R/92-21.* Glasgow, CAD Centre, Department of Design, Manufacture & Engineering Management, University of Strathclyde.

Kerr, S. M. (1993). "Customized Viewpoint Support for Utilizing Experiential Knowledge in Design". *Ph.D.* Glasgow, CAD Centre, Department of Design, Manufacture & Engineering Management, University of Strathclyde.

Kerr, S. M. (1993a). "The Importance of Manipulating Past Design Knowledge by Group Rationalisation". *Report CADC/R/93-02,* Glasgow, CAD Centre, Department of Design, Manufacture & Engineering Management, University of Strathclyde.

Kim, G. J. (1997). “Case-based design for assembly”, *Computer-Aided Design*, Vol.29, No.7, pp.497-506.

Kiriyama, T. and A. Kubota (1996). "The RACE Asynchronous Collaboration Environment Project", *IFIP TC5 WG5.2, International Conference on Knowledge Intensive CAD*, Carnegie Mellon University, Pittsburgh, PA, USA, Chapman & Hall:pp.189-197.

Klatzky, R. L. (1980). *"Human Memory - Structures and Processes",* W.H.Freeman & Co. San Francisco.

Lawlor-Wright, T. and C. Gallagher (1995). "Development of a Design for In-Circuit Test Advisory System", *Co-operation In Manufacturing:CIM AT WORK*, Kaatsheuvel, The Netherlands, CIMMOD/CIMDEV:pp.220-232.

Lee, J. Y. and K. Kim (1998). “A feature-based approach to extracting machining features”, *Computer-Aided Design,* Vol.30, No.13, pp.1019-1035.

Levitt, R. E., Y. Jin, et al. (1991). “Knowledge-Based Support for Management of Concurrent, Multidisciplinary Design”, *Artificial Intelligence in Engineering Design and Manufacture*, Vol.5, No.2, pp.77-95.

Lindbeck, J. R. (1995). "Design for Producibility", *Product Design And Manufacture.* New Jersey, Prentice Hall:pp. 317-368.

MacCallum, K. J., A. Duffy, et al. (1987). "The Knowledge Cube - A Research Framework for Intelligent CAD", *IFIP Workshop on Intelligent CAD*, MIT, Boston, USA. pp.38-44.

MacCallum, K. J. (1991). "Expert System Architecture Requirements For Concurrent Design", *The World Congress on Expert Systems Conference*, Florida, USA.

MacCallum, K. J. (1992). "Requirements For Intelligent Support of Concurrent Engineering", Concurrent Engineering:Requirements for Knowledge-Based Design Support, *10th European Conference on Artificial Intelligence*, Austria.

MacCallum, K. J. (1992a). "New Challenges for CAD", *Joint IMechE/SERC Workshop on IT and Product Design*, St.Albans, UK, I.Mech.E.

Maher, M. L., M. B. Balachandran, et al., (1995). *"Case-Based Reasoning in Design",* Lawrence Erlbaum Associates Hove, UK.

Maher, M. L. and J. Poon (1996). “Modelling Design Exploration as Co-Evolution”, *Microcomputers in Civil Engineering,* Vol.11, No.3, pp.195-209.

Maher, M. L. and J. H. Rutherford (1997). “A Model for Synchronous Collaboration Design Using CAD and Database Management”, *Research in Engineering Design*, Vol.9, No.2., pp85-98.

Mantyla, M., S. Finger, et al. (1996). *"Knowledge Intensive CAD",* Chapman & Hall, Carnegie-Mellon University, Pittsburgh, PA.

March, J. G. (1994). *"A Primer on Decision Making - How Decisions Happen",* The Free Press New York.

McMahon, C. and J. Browne (1993). *"CADCAM From Principles To Practice",* Addison-Wesley Publishers Ltd. Wokingham.

Medland, T. (1997). "A decision directed design approach", *Engineering Designer*, May, pp.4-7.

Meerkamm, H. (1994). "Design for X - A Core Area of Design Methodology", *Journal of Engineering Design*, Vol.5, No.2, pp.145-163.

Meerkamm, H. (1997). "Information Management in Design Process - Problems and Approaches to their Solution", *Darmstad Symposium*, Darmstad, Germany, Springer-Verlag.

Mills, J. J. (1993). "A Taxonomy of the Product Realization Process Environment", *Research in Engineering Design*, Vol.4, pp.203-213.

Mistree, F., W. Smith, et al. (1993). "A decision-based approach to concurrent design", *Concurrent Engineering - Contemporary Issues and Modern Design Tools*. H. R. Parsaei and W. G. Sullivan. London, Chapman & Hall: pp.127-158.

Mohd-Hashim, F., N. P. Juster, et al. (1994). "A Functional Approach to Redesign", *Engineering with Computers*, Vol.10, pp.125-139.

Molina, A., A. H. Al-Ashaab, et al. (1995). "A Review of Computer-Aided Simultaneous Engineering Systems", *Research in Engineering Design,* Vol.7, No.1, pp.38-63.

Molloy, E. and J. Browne (1993). "A knowledge-based approach to design for manufacture using features", *Concurrent Engineering - Contemporary issues and modern design tools*. H. R. Parsaei and W. G. Sullivan. London, Chapman & Hall: pp. 386-401.

Molloy, E., L. Mannion, et al. (1994). "Towards an Architecture for Design for Manufacture", Galway, Ireland.

Mortensen, N. H. (1995). "Linking Product Modelling To Design Theory", *International Conference on Engineering Design (ICED'95)*, Praha:pp.1403-1408.

Mortensen, N. H. (1997). "Design Characteristics as Basis for Design Languages", *Proceedings, 11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol.2, pp.23-30.

Mortensen, N. and C. T. Hansen (1994). "Towards Semantic Modelling of Organs in a Computer based design support system", *VDI Tagung Datenverarbeitung in der Konstruktion 1994*, Dusseldorf, Germany, VDI Bersichte.

Mortensen, N. H. and M. M. Andreasen (1996). "Designing in an Interplay with a Product Model - Explained by Design Units", *International Symposium, Tools & Methods For Concurrent Engineering (TMCE)*, Budapest, Hungary.

Morup, M. (1993). "Total Life Considerations in Design for Quality", *DFX Research Workshop*, Technical Univeristy of Denmark, Lyngby, Denmark.

Morup, M., "Total Life Models - An Important Tool In Design For Quality", International Conference On Engineering Design ICED'93, The Hague, 1993, WDK.

Nilsson, M. (1997). "Which comes first: Form Synthesis, Material Selection or Process Selection?", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol.2, pp.741-744.

Nnaji, B. O., T.-S. Kang, et al. (1991). "Feature Reasoning for sheet metal components", *International Journal of Production Research*, Vol.29, No.9, pp.1867-1896.

Norell, M. (1993). "The Use of DFA, FMEA AND QFD as tools for Concurrent Engineering in Product Development Processes", *International Conference on Engineering Design (ICED'93)*, The Hague: Vol.2, pp.867-874.

Nowack, M. L. (1997). "Design Guideline Support For Manufacturability", *Ph.D.*, Cambridge University, Cambridge: UK.

O'Donnell, F. J. (1997). "The Support of Danish Design Methodology For Design Coordination", *PhD Workshop in Design Methodology*, Trondheim, Norway.

O'Grady, P., R. E. Young, et al. (1991). "An Advice system for concurrent engineering", *International Journal of Computer Integrated Manufacturing*, Vol.4, No.2, pp.63-70.

Oh, J. S., P. O'Grady, et al. (1995). "A constraint network approach to design for assembly", *IIE Transactions,* Vol.27, pp.72-80.

Olesen, J. (1992). "Concurrent Development in Manufacturing - based on dispositional mechanisms", *Ph.D.,* The Technical University of Denmark, Lyngby: Denmark.

Olesen, J. (1995). "Strengthening the Understanding of Conceptual Design", *International Conference on Engineering Design (ICED'95)* , Praha:pp.471-476.

O'Sullivan, B. (1997). "A Constraint-based support tool for early-stage design within a concurrent engineering environment", *Proceedings of the 4th International Conference on Concurrent Enterprising*, University of Notingham:pp.129-138.

Pahl, G. and W. Beitz (1996). *"Engineering Design - A Systematic Approach",* Second Edition, Springer-Verlag, London.

Park, H., M. R. Cutkosky , et al. (1994). "An Agent-Based Approach to Concurrent Cable Harness Design", *Artificial Intelligence in Engineering Design and Manufacture*, Vol.8, No.1. pp.45-61.

Parsaei, H. R. and W. G. Sullivan (1993). *"Concurrent Engineering - Contemporary issues and modern design tools"*, Chapman & Hall London.

Phillips, E. M. and D. S. Pugh (1996). *"How to get a PhD"*, Open Univeristy Press Buckingham.

Popovic, D. and V. P. Bhatkar (1994). *"Methods and Tools For Applied Artificial Intelligence"*, Marcel Dekker Inc. New York.

Portelli, J. I. (1995). "A CAD System Linking Plastic Product Design And Mould Tool Design", *B.Eng.,* University of Malta, Msida, Malta.

Prasad, B. (1996). *"Concurrent Engineering Fundamentals, Integrated Product and Process Organization"*, Prentice Hall New Jersey.

Prasad, B. (1996a). "Concurrent Engineering Definitions", *Concurrent Engineering Fundamentals, Integrated Product and Process Organization.* New Jersey, Prentice Hall: pp. 164-215.

Prasad, B., F. Wang, et al. (1997). "Towards a computer-supported cooperative environment for concurrent engineering", *Concurrent Engineering: Research and Applications*, Vol.5, No.3, pp.233-252.

Pugh, S. (1991). *"Total Design"*, Addison-Wesley Publishers Ltd. Essex.

Pye, R. G. (1989). *"Injection Mould Design",* Longman Scientific & Technical UK.

Ranky, P. G. (1994). *"Concurrent/Simultaneous Engineering (Methods, Tools & Case Studies)"*, CIMware Limited.

Reich, Y. (1992). "The value of Design Knowledge", *The Seventh Banff Knowledge Acquisition for Knowledge Based Systems Workshop*, Banff, Canada.

Rich, E. and K. Knight (1991). *"Artificial Intelligence",* McGraw-Hill Inc. New York.

Robinson, P., A. Matthews, et al. (1993). "A Computer Knowledge-Based System for Surface Coating and Material Selection", *Surface and Coating Technology*, Vol.62, pp.662-228.

Rogers, J. L. (1997). "Reducing Design Cycle Time and Cost Through Process Resequencing", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol. 1, pp. 193-198.

Roozenburg, N. F. M. and J. Eekels (1995). *"Product Design: Fundamentals and Methods"*, John Wiley & Sons Ltd. Wilshire.

Rychener, M. D. (1988). "Research In Expert Systems for Engineering Design", *Expert Systems for Engineering Design.* M. D. Rychener. London, Academic Press Inc.:pp.1-28.

Salomons, O. W., F. J. A. M. van Houten, et al. (1993). "Review of Research in Feature-Based Design", *Journal of Manufacturing Systems*, Vol.12, No.2, pp.113-132.

Salzberg, S. and M. Watkins (1990). "Managing Information for Concurrent Engineering: Challenges and Barriers", *Research in Engineering Design*, Vol.2, No.1, pp.35-52.

Schulte, M., C. Weber, et al. (1993). "Functional Features For Design In Mechanical Engineering", *Computers in Industry*, Vol.23, pp.15-24.

Schut, C. and B. Bredeweg (1996). "An Overview Of Approaches To Qualitative Model Construction", *The Knowledge Engineering Review*, Vol.11, No.1, pp.1-25.

Seliger, G., E. Zussman, et al. (1994). "Integration Of Recycling Considerations Into Product Design - A System Approach", *Information and Collaboration Models of Integration*. S. Y. Nof. Netherlands, Kluwer Academic Publishers: pp. 27-41.

Sen, P. and J. B. Yang (1995). "Multiple-Criteria Decision-Making in Design Selection and Synthesis", *Journal of Engineering Design*, Vol.6, No.3, pp.207-230.

Shah, J. J. and M. Mantyla (1995). *"Parametric and Feature-based CAD/CAM"*, John Wiley & Sons, Inc. New York.

Shankar, R. S. and D. G. Jansson (1993). "A Generalized Methodology For Evaluating Manufacturability", *Concurrent Engineering*. H. R. Parsaei and W. G. Sullivan. London, Chapman & Hall: pp.248-263.

Sharples, M., D. Hogg, et al. (1989). *"Computers and thought"*, The MIT Press Massachusetts.

Sivaloganathan, S. and N. F. O. Evbuomwan (1997). "Quality Function Deployment - The Technique: State of the Art and Future Directions", *Concurrent Engineering: Research and Applications*, Vol.5, No.2, pp.171-180.

Smith, B. (1997). "Profiles: Pascoe Engineering Ltd.". The Glasgow Technologist, April:pp. 4-5.

Smith, R. (1990). *"Collins Dictionary of Artificial Intelligence"*, Collins, Glasgow.

Smithers, T. and W. Troxell (1990). "Design is Intelligent Behaviour, But What's the Formalism?", *Artificial Intelligence in Engineering Design and Manufacture*, Vol.4, No.2, pp.89-98.

Spath, D., M. Hartel, et al. (1996). "Recycling-oriented Information Management: A prerequisite for Life Cycle Engineering", *Journal of Engineering Design*, Vol.7, No.3, pp.251-264.

Starvey, C. V. (1992). *"Engineering Design Decisions"*, Edward Arnold London.

Stenros, A. (1997). "Design is a tool for competition", *Form Function Finland*, Vol.1, pp.1-5.

Su, C.-J. and M. M. Tseng (1997). "EKAMD - A Knowledge-Based Concurrent Engineering Support System", *Concurrent Engineering: Research and Applications*, Vol.5, No.1, pp.59-76.

Subramaniam, B. L. and K. T. Ulrich (1998). "Producibility Analysis Using Metrics Based on Physical Process Models", *Research In Engineering Design*, Vol.10, No.4, pp.210-225.

Swift, K. G. (1987). *"Knowledge Based Design For Manufacture"*, Kogan Page London.

Swift, K. G. and A. J. Allen (1992). "Techniques in Design for Quality and Manufacture", *Journal of Engineering Design*, Vol.3, No.1, pp.81-91.

Swift, K. G. and A. J. Allen (1994). "Product variability risks and robust design", *Proc. Instn. Mech.Engrs.*, Vol.208, No.B, pp.9-19.

Swift, K. G., M. Raines, et al. (1997). "Design Capability And The Costs Of Failure", *Proc. Instn. Mech.Engrs.*, Vol.211, No.B, pp.409-423.

Tang, M. X. and K. Wallace (1997). "A Knowledge-Based Approach to CAD Systems Integration", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol. 2, pp.185-190.

Tang, M. X. (1996). "An AI-Based Architecture For Concurrency Management In Engineering Design", *National Design Engineering Conference*, Chicago IL, ASME:pp.1-9.

Terpenny, J. P. and B. O. Nnaji (1993). "Feature Based Design Evaluation For Machine/Tool Selection For Sheet Metal", *2nd Industrial Engineering Research Conference Proceedings*, USA, Institute of Industrial Engineers:pp.26-30.

Tezuka, A. (1992). "Adaptive Remeshing Process With Quadrangular Finite Elements", *Advances in Engineering Software*, Vol.15, pp.185-201.

Thomson, S. (1995). "Design Activities", *M.Sc.*, CAD Centre, University of Strathclyde, Glasgow UK.

Tichem, M. (1993). "Design For Manufacturing And Assembly; A Closed Loop Approach", *International Conference on Engineering Design (ICED'93)*, The Hague, Heurista:Vol.2, pp.1033-1040.

Tichem, M. (1997). "A Design Coordination Approach To Design For X", *Ph.D.*, Delft University of Technology.

Timperi, A. (1997). "Design of Environmental Friendly Forest Machines", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol. 1, pp.43-48.

Tizzard, A. (1994). *"Numerical Methods", An Introduction To Computer Aided Engineering*. T. Andrew. London, McGraw-Hill Book Company: pp.125-155.

Tjalve, E. (1979). *"A Short Course in Industrial Design"*, Newnes-Butterworths London.

Tomiyama, T., Y. Umeda, et al. (1995). "Knowledge Systematization For A Knowledge Intensive Engineering Framework", *First IFIP Workshop on Knowledge Intensive CAD*, Helsinki University of Technology, Espoo, Finland:pp.55-80.

Tomiyama, T. (1996). "Concurrent Engineering: A Successful Example for Engineering Design Research", *The Design Productivity Debate*. A. H. B. Duffy. London, Springer-Verlag: pp.175-186.

Tomiyama, T. (1998)., "An Overview of The Modelling of Synthesis Project", *International Symposium on Modelling of Synthesis*, The University of Tokyo, Tokyo, Japan, The Modelling of Synthesis Project, Japan Society for the Promotion of Science (JSPS) & Research for the Future Program:pp. 3-12.

Turban, E. (1993). *"Decision Support And Expert Systems - Management Support Systems"*, Macmillan Publishing Company New York.

Ullman, D. G., S. Wood, et al. (1990). "The Importance of Drawing in the Mechanical Design Process", *Computer & Graphics*, Vol.14, No.2, pp.263-274.

Uschold, M. and M. Gruninger (1996). "Ontologies: Principles, Methods And Applications", *The Knowledge Engineering Review*, Vol.11, No.2, pp.93-136.

Vajna, S. and B. Wegner (1997). "Support The Product Life Cycle By a Computer Aided, Integrated Feature Based Modelling Scheme.", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology:Vol.2, pp.301-306.

van den Kroonenberg, H. H. (1987). "CAD Applications In The Creative Phase Of The Methodical Design Process", *Computer Applications in Production and Engineering*. K. B. Estensen, P. Falster and E. A. Warman. North-Holland, Elsevier Science Publishers B.V.:pp. 373-382.

Victor, S. K., D. C. Brown, et al. (1993). "Using multiple expert systems with distinct roles in concurrent engineering system for powder ceramic components", *Applications of Artificial Intelligence in Engineering*. G. Rzevski, J. Pastor and R. A. Adey. Southampton, Computational Mechanics Publications. VII:pp.83-96.

Wallace, K. (1997). "Product Development and Design Research", *11th International Conference on Engineering Design*, Tampere, Finland, Tampere University of Technology (Keynote Paper).

Walters, J. R. and N. R. Nielsen (1988). *"Crafting Knowledge Based Systems - Expert Systems Made Realistic"*, John Wiley & Sons New York.

Wang, M. H. and M. R. Johnson (1995). “Design for Disassembly and Recyclability:A Concurrent Engineering Approach”, *Concurrent Engineering:Research and Applications*, Vol.3, No.2, pp.131-134.

Watson, B., D. Radcliffe, et al. (1996). "A Meta-Methodology For The Application Of DFX Design Guidelines", *Design For X*. G. Q. Huang. London, Chapman & Hall:pp. 441-462.

Weber, C. (1996). "What Do We Call A 'Feature' And What Is Its Use? - Results Of FEMEX Working Group 1", *FEMEX Working Group 1, 'Feature Definition and Classification'*, Germany.

Wierda, L. S. (1991). “Linking Design, Process Planning and Cost Information by Feature-based Modelling”, *Journal of Engineering Design*, Vol.2, No.1, pp.3-19.

Willemse, M. A. and T. Storm (1995). "Designer Support For Assembly Decisions", *Co-operation In Manufacturing:CIM AT WORK*, Kaatsheuvel, The Netherlands, CIMMOD/ CIMDEV:pp.206-219.

Willemse, M. A. (1997). "Interactive Product Design Tools For Automated Assembly", *Ph.D.*, Delft University of Technology, Delft.

Williams, M. and A. Taleb-Bendiab (1998). "Software Support For Agile Manufacturing Systems and Virtual Enterprises Through the Use of Multi-Agent Systems", *2nd International Symposium 'Tools and Methods For Concurrent Engineering'*, Manchester, UK:pp.456-463.

Winner, R. I. (1988). "The Role of Concurrent Engineering in Weapons Systems Acquisition", *Report R-338,* Institute of Defense Analysis, Alexandria, Va.

Winston, M. E., R. Chaffin, et al. (1987). “A Taxonomy of Part-Whole Relations”, *Cognitive Science*, Vol.11, pp.417-444.

Wood III, W. H. and A. M. Agogino (1996). “Case-Based Conceptual Design Information Server For Concurrent Engineering”, *Computer Aided Design*, Vol.28, No.5, pp.361-369.

Woodhead, N. (1990). *"Hypertext & Hypermedia",* Sigma Press Wilmslow, England.

Xia, S. and N. Smith (1996). “Automated modelling: a discussion and review”, *The Knowledge Engineering Review*, Vol.11, No.2, pp.137-160.

Yan, X. T. and J. E. E. Sharpe (1995). "Reasoning and Truth Maintenance of Causal Structures in Interdisciplinary Product Modelling and Simulation", *International Workshop on Engineering Design: AI System Support for Conceptual Design*. England, Springer: pp. 403-425.

Young, R. I. M. (1996). "Supporting Multiple Views in Design For Manufacture", IFIP TC5 WG5.2, Internationa Conference on Knowledge Intensive CAD, Carnegie Mellon University, Pittsburgh, PA, USA, Chapman & Hall:pp.259-268.

Zhang, Y. (1998). "Computer-based Modelling and Managment for Current Working Knowledge Evolution Support", *Ph.D.,* University of Strathclyde, Glasgow.

Appendix A – Glossary of Terms

Term	Definition
Abstraction	A process by which individual items of knowledge (concepts) are organized into a 'super-concept' having less detail than the original [Kerr 1993].
AI	Artificial Intelligence, a field in computer science – see [Rich et al. 1991]
Artefact	The product i.e. a man-made, physically realized, material object.
Artefact life	This is total elapsed time from when the need of an artefact was established to when the artefact is removed from existence – see section 2.2 for details.
Artefact life actors	Human beings involved during an artefact's life e.g. designers, tool makers, production engineers, inspectors, users and service engineers.
Artefact life model	A model collectively reflecting the artefact solution and solutions of life-phase systems that the artefact will interact with during its life.
$CAD_sF\Sigma X$	'Computer Aided Design Synthesis For Multi-X' system – a design tool;
Cavity	The female portion of a mould which gives the component its external form
CC	Concurrent consideration of total life-cycle issues – see section 1.1.1.
Class	A collection of elements with common characteristics
Component	A single material element realized without any assembly operations (see section 2.1)
Concept	A chunk of related information about some entity or object (e.g. the concept of a car).
Consequence	An outcome resulting from a decision commitment
Constraint	A restriction on a set of possible options
Core	The male portion of a mould tool which forms the internal shape of the moulded component
Core-pin	A pin placed in a mould tool for forming a hole or opening in a plastic moulded component
CWK	Current (design) working knowledge (see Figure 7.1) - see [Zhang 1998].
CX	Concurrent execution of independent/semi-independent product development activities – see section 1.1.1.
Decision commitment	The option selected from the given alternatives
Design characteristic	Design solution attributes e.g. material = cast iron
DFΣX	Design for Multi-X
DFA	Design For Assembly
DFC	Design For Cost
DFE	Design For the Environment
DFM	Design For Manufacture
Draft angle	A taper angle of form features in components that are to be injection moulded, required to facilitate the removal of the moulded component from the mould;
$D_sF\Sigma X$	Design Synthesis for Multi-X
Feature	An information unit (element) representing a 'region of interest' within a product [Weber 1996].
Generalization	The process of deriving information about a class of objects/elements/ concepts that also pertains to members of that class [Coyne et al. 1989].
HTML	HyperText Markup Language, used for generating Wold Wide Web Documents – see [Bacon et al. 1997].
Inference	The process of reaching a conclusion from an initial set of propositions, the truths of which are known are assumed [Smith 1990].
Inheritance	The mechanism of passing information from one level of abstraction of organized objects/elements/concepts to a lower level [Walters et al. 1988].
Instance	A specific concept of a class of concepts e.g. Suzuki Swift is an instance of the concept 'car'.
KC	**K**nowledge of life-cycle **C**onsequences
KICAD	A *'Knowledge Intensive CAD'* system - see [Brown 1996; Mantyla et al. 1996];
Knowledge	Data, information and experience that support the inference of new facts from given facts;

LCC	Life-Cycle Consequence – a consequence influencing artefact life-phases – for details see section 2.3.5;
LCC_i	An interacting type LCC (see 6.3.2)
LCC_{ni}	A non-interacting type LCC (see 6.3.1)
LCPE	Life Cycle Phase Element - an element (e.g. mould tool) that forms part of a life-phase system (see 3.3). LCPEs can be used to synthesize life-phase system models (see 7.3.3);
Life-oriented design	A design process concerned with the generation of a solution that simultaneously caters for a host of multiple artefact life-phase requirements;
Life-phase	A time segment in the artefact's life, during which there occurs a change in the state of the artefact. This work considers main four phases involved in a mechanical artefact's life : design, realization, use and disposal – see 2.2.4;
Life-phase model	A decomposable model consisting of life-phase systems with which the artefact interacts during that particular life-phase;
Life-phase system	A system that delivers effects to transform an operand in state 1 to an operand in state 2. For instance a lathe is fabrication system employed in the realization life-phase;
Means	'That by which a result is brought about' In the case of design, 'means' can be techniques, methods or tools [Duffy et al. 1998];
Non-directional inference	Inference that can take place both from an initial set of propositions to reaching a conclusion *and* vice-versa, from a given conclusion to a set of propositions. Also known as multi-directional inference - (see [Bowen 1995]).
Parting surface	The part of both mould plates, adjacent to the impression, which butt together to form a seal and prevent loss of plastic material from the impression;
PDE	Product design element - an element that forms part of a mechanical artefact system. Different system levels (e.g. component) have different PDEs (see 2.1);
Premise	Observations collected, given information;
Product development	A segment in an artefact's life during which a production plan, an artefact solution and a marketing plan are generated to enable the realization phase to commence (see Figure 2.6; [Roozenburg et al. 1995]);
Providence	To foresee and take into account aspects of the total life that are fixed during design – see [Olesen 1992];
Reasoning	The drawing of inferences or conclusions from known or assumed facts – see [Popovic et al. 1994] for details;
S.E.L.	Synthesis Elements Library – a library of PDEs and LCPEs. See section 7.3.2
Shrinkage allowance	The additional dimensions that must be added to a mould tool to compensate for shrinkage of the plastic moulding material on cooling;
Sink mark defect	A depression (defect) on the surface of moulded plastic component caused by internal shrinkage;
Split cavity	A general term for a part of a mould tool cavity which can be withdrawn at right angles to the mould's axis. A split cavity permits the moulding of certain component form features which have a projection below the parting surface;
Sub-class	A collection of elements with common characteristics, that are a sub-set of an existing class of elements e.g. 'sedan cars' is a sub-class of the class 'cars'.
Synthesis decision commitment	Decision commitments that are reflected in the artefact's solution model. These cause a change in the concretization or detailing of a solution; see section 6.2;
System	A finite set of elements collected to form a whole under certain well-defined rules, whereby certain definite relationships exist between the elements, and to its environment [Hubka et al. 1988];
Taxonomy	A hierarchical classification of objects/concepts/elements [Smith 1990].
Weld line defect	A mark (surface defect) formed by the incomplete fusion of two or more streams of plastic flowing together inside a mould tool;

Appendix B – Representative Computer-Based Providence Means

This appendix provides a review of *representative* computer-based means that directly support providence. This review is not intended to provide an exhaustive enumeration of systems. Rather, it aims to disclose the characteristics of selected computer-based systems with respect to the criteria established in Chapter 4, in order to provide a basis for establishing the strengths and limitations of different class of means supporting providence. Table B.1 provides representative *AI-based* support means, whilst Table B.2 provides representative *feature-based* means.

Table B.1a – Comparative Matrix for AI Based Providence Means

Tool/ Author(s)	*Solutions generated: (N)one, (A)rtefact, (L)ife-phase*	**Artefact Life Knowledge**			**Other Comments**
		Revealed during (s)nthesis or (a)nalysis	*View Single (X) = 1(X) Multiple (X) = Σ(X)*	*Type (G)eneric or life (S)pecific*	(Domain, support for artefact life exploration, techniques)
Constraint Networks					
Klaus3 [Bowen 1995]	(A)	During (s) but requires co-operation from other team members	Σ(X) but segmented views (different perspectives of the network)	(G) – no underlying artefact life-phase models	• Domain: printer wire board • requires a number of team members working together • the constraint network can be divided into different fields of view to reflect the different perspectives of various members of a concurrent engineering team • Uses the Galileo2 language
Larry [O'Grady et al. 1991]	(A)	During (a) - constraining influences between the functional design solution and manufacturing have to be first captured as a network	1(X) – manufacturing issues	(S) but is based on manufacturing knowledge rather than models of the manufacturing system	• Printed wiring boards • Constraining influences between the functional design solution and manufacturing have to be first captured as a network • designers can explore the 'artefact solution' to fit the manufacturing process
DFA system [Oh et al. 1995]	(A)	During (a) – key design evaluation factors and constraints have to be first identified by the user and then expressed as a constraint network	1(X) – assembly issues	(S) but user needs to declare relationships between the artefact & life issues	• Domain: generic and applicable to different artefact system levels; • Employs the constraint modelling system SPARK; • exploration is limited to exploring different values of parameters in the existing constraint network;
Case-Based Reasoning Design Tools					
CDIS [Wood III et al. 1996]	(A)	(a) – providence separate from synthesis decision making	Σ(X) but segmented views	(G)	• Domain: generic • employs hybrid case-based reasoning system with SQL database • case indexing aided by an intelligent thesaurus to retrieve cases from different artefact viewpoints and at different levels of abstraction
REV-ENGE [Kim 1997]	(A)	(a) – the input to a tool is a candidate solution	1(X) – assembly issues	(G)	• Domain: mechanical artefacts • designers can explore the 'artefact solution' to fit the assembly process • tool focuses on reviewing assembly issues to support 'artefact redesign' • artefact life exploration is not supported – only artefat solution can be explored • rather than storing 'final design solutions' as cases, REV-ENGE stores cases of DFA re-design methods

Table B.1b – Comparative Matrix for AI Based Providence Means

Tool/ Author(s)	Solutions generated: (N)one, (A)rtefact, (L)ife-phase	Artefact Life Knowledge			Other Comments
		Revealed during (s)nthesis or (a)nalysis	*View Single (X) = 1(X) Multiple (X) = Σ(X)*	*Type (G)eneric or life (S)pecific*	(Domain, support for artefact life exploration, techniques)
Knowledge Based Systems					
DFS [Eubanks et al. 1993]	(A)	During (a) – requires a description of the candidate solution.	1(X) - life-cycle service costs	(G)	• Domain: Mechanical systems; • Commitments concerning the service process can be explored by the user– however, there are no underlying service system models; • Candidate solution represented as a semantic networks;
DAISIE [Ishii 1991]	(A)	During (a) – requires a description of the candidate artefact solution.	Σ(X) but segmented views – assembly, manufacture use (serviceability)	(S) to the solution generated, but artefact life-phase knowledge is fixed	• Domain: Injection moulded or forged components • only artefact solution can be explored • artefact life actor's design knowledge is stored as a Compatability Knowledge Base • no underlying life-phase models i.e. artefact life exploration is not supported
I3D [Victor et al. 1993]	(A)	During component concept (a)	Σ(X) but segmented views – manufacture, inspection, cost & reliability	(G)	• Domain: ceramic componetns • only artefact solution commitments can be explored • employs intelligent agents (expert systems) with different artefact life roles • system also provides 're-design advice'
DESIGN CRITIQUE SYSTEM [Changchien et al. 1996]	(A)	During (a) – tool requires first a description of the candidate solution	Σ(X) views – manufacture & assembly	(G)	• Domain: rotational machined parts • no underlying life-phase models i.e. artefact life exploration is not supported • rotational parts topologically represented by features and relationships • provides the designer with suggestions of correction
mfk [Meerkamm 1994]	(A)	During candidate component solution (a)	Σ(X) but segmented views	(G)	• Domain: mechanical components • no underlying life-phase models, thus no artefact life exploration
MIDAS System [Bonfield et al. 1997]	(A)	During solution (a) – requires a CAD data as input	Σ(X) but segmented views (manufacture, testing, servicing)	(S) to the realization phase (has a company specific process information)	• Domain: printed circuit board assemblies • some artefact life exploration – designer can acquire information on production facilities available, but there is no underlying models of production systems • Implemented in Poplog & uses HiPWorks to provide multi-media rule explanation • Object-oriented representation of PCB solution • Provides rule explanation and redesign suggestions
Knowledge-based CAD system [Huh et al. 1991]	(A)	During conceptual solution (a) – reqiures artefact's primary geometry as input	Σ(X) but segmented views – manufacturing (moudability), use (mechanical behaviour)	(S) to the artefact solution but not to the life-phase systems that will be encountered	• Domain: thermoplastic components • no life-phase system modelling is supported • provides recommendations for appropriate rib and boss parameters • employs an expert system (RIBBER) loaded with relevant production rules and a geometric modeller (PRO/ENGINEER)

Table B.2 – Comparative Matrix for Feature Based Providence Means

Tool/ Author(s)	*Solutions generated: (N)one, (A)rtefact, (L)ife-phase*	**Artefact Life Knowledge**			**Other Comments**
		Revealed during (s)nthesis or (a)nalysis	*View Single (X) = 1(X) Multiple (X) = Σ(X)*	*Type (G)eneric or life (S)pecific*	(Domain, support for artefact life exploration, techniques)
Feature-Based Design Tools					
Sheet Metal Design System [Terpenny et al. 1993]	(A)	(a) – during sheet metal solution analysis	1(X) - realization phase issues- compatible machine tools	(G)	• Domain: Sheet metal components • no underlying life-phase models, thus no artefact life exploration • Tool also support process planning • Uses object-oriented technology
[Fuh et al. 1996]	(A)	(a) – during candidate solution analysis; a 3D model of the part is required as input	1(X) – realization phase issues: process plan, fixture plan	(G)	• Domain: mechanical components • no underlying life-phase models, thus no artefact life exploration • Tool generates a process plan and a fixturing plan
[Lee et al. 1998]	(A)	(s) – machining operations revealed during the evolution of component solution	1(X) – realization phase issues: machining operations required, ease of machining, tolerance	(G) – no underlying realizaiton phase model	• Domain: mechanical components • no underlying life-phase models, thus no artefact life exploration • tool provides rapid feedback on manufacturing implications during solution evolution • Tool employs the concept of extracting machining features from form features
[Gupta et al. 1995] (a system approach)	(A)	(a) – requires a solid model of solution	1(X) – realization phase (manufacturability in terms of cost & quality)	(G)	• Domain: machined parts • assumes information about machining operation is fixed i.e. no underlying life-phase models • employs the concept of 'machining feature' recognition.
CBD [Feng et al. 1995]	(A)	(s) – during synthesis of a feature based model	1(X) – realization phase (machining)	(G) – no life-phase models	• Domain: machined components • Employs Object-oriented technology for features & production rules for 'machining constraints"
[Geiger et al. 1996]	(A)	(a) – during candidate solution analysis; a feature based solution description is required	1(X) – realization phase costs	(S) for the realization phase	• Domain: mechanical components • no underlying life-phase models, thus no artefact life exploration – but plant manufacturing data is used to determine component's process plan & costs • Tool generates a process plan and a fixturing plan

Appendix C – Notation

Scope

For formal explanation of the phenomena model (Chapter 6) and the LCC knowledge model (Chapter 8) established in this research, the following notation is employed. Some notation (e.g. connectives) is based on the 'alphabet' of predicate logic [Popovic et al. 1994] for describing *elements* and *relationships* between them. However, to specifically explain synthesis decision making concerning artefact and life-phase models, further notation established for the purpose of this research is employed, as defined in this appendix. Section C.1 concerns symbolic notation, section C.2 concerns logical relationships, section C.3 concerns component modelling, section C.4 concerns life-phase modelling and C.5 concerns decision-making notation.

C.1 Symbolic Notation

	Notation	***Meaning***
Symbol	Σ	= summation
	$\forall$	= for all
	[a]	= an element that can be defined by the designer; for concurrent synthesis these include PDEs, LCPEs;
	⌊a⌋	= 'a' is a class; e.g. ⌊ferrous materials⌋
	[a-b]	= ranging from 'a' to 'b' i.e. constrained from 'a' to 'b'
	$\delta(x)$	= a change in the value of 'x'
Connectives	$a \wedge b$	= 'a' and 'b' (conjunction)
	$a \vee b$	= 'a' or 'b' (disjunction)
	$\neg$	= 'not' e.g. $\neg$ b = not 'b'
Set	{....}	= a set of some elements e.g. { [a] } ;
	$\{a\} \subset \{a,b\}$	= 'a' is a subset of {a, b}

	a ∈ {b}	= 'a' *member of* set {b}
	a ∉ {b}	= 'a' not a *member of* set {b}

C.2 Logical Relationship Notation

	a => b	= 'a' implies 'b', i.e. If 'a' then 'b'
	a <>---b	= 'a' is attained by 'b' i.e. 'a' requires 'b'
	a >-- b	= 'a' is suitable for 'b'
	a ⇐ b	= 'a' has 'b'
Structural	a •→ b	= 'a' is a *kind_of* 'b', i.e. 'a' inherits from 'b'
	a → b	= 'a' is *part_of* the structure of 'b' [Winston et al. 1987]

C.3 Component Modelling Notation

	<model>$_{component}$	A compositional model of the component made up of [F], [F_a] and [M] related with part_of (→) relationships.
Form features	[F]	= form feature *e.g. [F] = [hole]*
	$[F]_p$	= a parameter associated with [F] *e.g.*$[hole]_{depth}$ *is the parameter 'depth' associated with form feature [hole];*
	$[F]_{pv}$	= is a feature parameter value *e.g.* $[F]_{pv}$= $[hole]_{depth=20mm}$;
	$[SF]_{i/e}$	= a suface finish value (*i = internal; e = external*) associated with a form feature
	[T]	= Tolerance on feature parameter value *e.g.*$[hole]_{depth}$ ⇐ *([T]=± 0.01mm) is the tolerance for the hole's depth parameter*
Assembly features	$[F_a]$	= assembly feature *e.g* $[F_a]$ = *[snap-fit]*
	$\{[F_a]_c\}$	= a set of assembly feature characteristics *e.g.* $\{[snap\text{-}fit]_c\}$ = *{non-permanent, ….. , weak-bond}*
Material	[M]	= component material *e.g. [M] =[aluminum];*

{m} = set of material [M] properties *e.g. {ductile, …, conductive}*

C.4 Life-phase Modelling Notation

Technical processes

[P] = a life-cycle process *e.g. [P] = [milling]; subscript 1 if primary; subscript 2 if secondary: e.g.* $[P]_1$ *= [sand casting];* $[P]_2$ *= [drilling];*

$[P]_p$ = a parameter associated with [P] *e.g .* $[moulding]_{temperature}$;

$[P]_{pv}$ = is a parameter value for $[P]_p$ *e.g* $[moulding]_{temperature=1000^oC}$

$[P]_t$ = life-cycle process technology, based on the model of a technical process by [Hubka et al. 1988]; *e.g. for [P] = Spark Erosion,* $[P]_t$ *= Material erosion by electric sparks*

{ t } = set of process technological properties *e.g. for* $[P]_t$ *= Material erosion by electric sparks, {t} = {conductive* $\wedge$ *…* $\wedge$ *low hardness values}*

$(Q)_e$ = an integer range [a - b] indicating the process economic quantity - *e.g.* $([P]=[Sand\_casting]) \Leftarrow (Q)_e \geq 1$

Process technical systems

[TS] = a physical, technical system [Hubka et al. 1988] that realizes the transformation effects of process [P] *e.g. for ([P]=[injection moulding]), ([TS]=[injection moulding machine]);*

$[TS]_p$ = a parameter associated with [TS] *e.g .* $[core\text{-}pin]_{diameter}$;

$[TS]_{pv}$ = is a parameter value for $[TS]_p$ *e.g.* $[core\text{-}pin]_{diameter\ =\ 12mm}$

Life-phase modelling

$<Phase>_i$ = a compositional model for life-phase 'i' e.g. $<Phase>_{Realization}$. This consists of [P] and $[TS]_p$, related with *part_of* relations. Embedded in this model are $]P[_t$, $[P]_p$, $[P]_{pv}$, { t } and $[TS]_{pv}$

[Q] = quantity of components required in the 'use' phase

<life model> = artefact's life compositional model, consisting of $<model>_{component}$ and $<phase>_i$ models.

Performance measures

(PM) = Performance Measure e.g. cost, throughput time, quality, efficiency, flexibility and risk [Olesen 1992].

(PM)<Phase>$_i$ = PM is a performance measures for life-phase '*i*'.

e.g. (Cost)<Phase>$_{Disposal}$.

(PM)[P] = PM is a performance measure of [P] e.g. (Time)([P]=[milling]);

δ(PM) = a change in PM, value being relative and between –10 to +10

C.5 Synthesis Decision-Making Notation

Decision making

D = a decision query arising due to a set of alternatives *e.g. D = which form feature?*

{O} = a set of decisoin query (D) options ; the set includes reusable PDEs and LCPEs; *e.g. {O} = {[hole] $\vee$ [rectangular_hole] $\vee$ [triangular_hole] }*

(i) the set {O} can be continuous over a range {[a – b]} *e.g. orientation angle values range from {[20^O - 180^O]}* or

(ii) the set {O} can be discrete of the type {a $\vee$ b $\vee$ c} *e.g. material options are: {[mild steel] $\vee$ [copper] $\vee$ [cast iron] }*

D{O} = decision proposal: a query (D) being operated on a set of options {O}; *e.g. what value for the angle?{[20^O-180^O]}*

$[d]_E\{O\}$ = a synthesis decision commitment i.e. the alternative *selected* from the proposal D{O};

e.g. D=what value for the orientation angle?

with {O} = {[20^O - 180^O]} and e.g. $[d]_E\{O\}$ =45^O

$[d]_E\{O\} \rightarrow$ <model>$_x$ = a synthesis decision commitment made to <model>$_x$

CN {0} = a constraint (i.e. a restriction) on the members of set {O} forming part of the decision proposal D{O}.

e.g. the set of unconstrained values for the parameter angle (Y) is {[0^O - 360^O]} but due to some constraint CN(Y) = ((Y $\geq$ 45^O) $\wedge$ (Y $\leq$ 180^O)), the valid set being { [45^O - 180^O] };

e.g. For a discrete set of options {O} of the type {a ∨ b ∨ c} a typical constraint is CN{O} = ¬ b ;

S = {D} = decision space i.e. a *set* of decision queries {D} i.e. S={D} = { $D_1 \wedge D_2 \wedge D_3$ } that have to be all considered and eventually committed i.e. not optional;

For instance, if {[F]} = {[hole]∨ [rectangular_hole] ∨ [triangular_hole]} and D {[F]} results in $[d]_E$*{[F]} = [hole], then, S={D'} = { what* $[hole]_{radius}$*? ∧ what* $[hole]_{depth}$*? ∧ what* $[hole]_{angle}$*? ∧ what* $[hole]_{centre\_point}$*? }*

LCC_j = 'x' 'x' is a LCC of type 'j' resulting from a synthesis decision commitment. If j='ni', then it is a non-interacting LCC; If j='i', then it is an interacting LCC.

Appendix D – Knowledge Representation Schemes

Scope

This appendix provides a non-exhaustive background to different knowledge representation schemes *commonly* supported by Artificial Intelligence (AI) development tools. This understanding provides a foundation as to why a hybrid representation scheme was employed in FORESEE. Extensive details on representation schemes can be found in [Walters et al. 1988; Coyne et al. 1989; Rich et al. 1991; Giarratano et al. 1994; Popovic et al. 1994]. From an AI point of view, recognized knowledge is viewed as a collection of facts about the world to be formally represented in a computer. It forms the knowledge base of an AI system such as a KICAD tool. *Facts* are fragments of knowledge defining something about the *objects (e.g. PDEs and LCPEs)* in the world we would like to represent in the computer. They usually express the *state* of the objects in the selected domain and the *relationships* between them [Coyne et al. 1989]. *Knowledge representation* is concerned with *how* recognized knowledge (Figure 8.1) can be formally codified into some computable syntax [Rychener 1988; Popovic et al. 1994; Tomiyama et al. 1995]. Due to the basic types of knowledge categories [Popovic et al. 1994], AI development tools provide two classes of knowledge representation schemes: *declarative* and *procedural,* these disclosed next.

D.1 Declarative Schemes

These schemes emphasize the representation of facts concerning *objects*, which can be *static* (do not change with time) or *dynamic* [Popovic et al. 1994]. Common schemes are *semantic nets, scripts* and *frames.*

Semantic nets

These rely on graph theory and use nodes and labelled links for descriptions of *objects* and the *interrelationships* between them. A typical example representing facts about a 'dog' (an object) is:

- 'the dog' is an 'animal'
- 'the dog' is 'my dog'
- 'my dog' is a 'dachshund'
- 'the dog' is a 'female'
- 'the dog' has a 'short tail'
- 'the dog' has 'black hair'

Network representation

As reflected, 'the dog' object has a number of binary relations e.g. 'is a', 'has a' with other objects. Semantic nets are useful for the *graphical* representations of such interrelationships. Such a network is represented in computer form using predicate logic of the type:

Example

- isa (dog, animal)
- isa (my dog, dachshund).

As argued in [Popovic et al. 1994], semantic networks are suitable for the representation of a taxonomy between different objects.

Reasoning with semantic networks

A common reasoning technique with semantic networks is based on *matching network structures* i.e. on constructing a network fragment containing both the nodes and specific values and those whose values are to be inferred as variables [Popovic et al. 1994].

Scripts

These are specialized structures primarily used in representing typical *sequences* of *events* or causal chains of events. They store the knowledge about what basically happens when for example, a person goes to the cinema or to a restaurant. A non-exhaustive script of the sequence of events for a person going to a cinema is:

Script representation

- First event — Select restaurant
- Second event — Enter selected restaurant
- Third event — Take a seat in the restaurant
- Fourth event — Chose and order food
- Fifth event... — ....

Consequently, a script does not contain knowledge about an object (e.g. the person) but rather about the behaviour of the object in a given situation. Scripts are useful for representing situations in which a *fixed* sequence of events is known *before hand* unlike with synthesis where decisions (events) are rarely carried out in a fixed sequence.

Reasoning with Scripts

Reasoning with scripts requires first selecting an appropriate script – one which best matches the given *specific* conditions. Once a feasible script is matched, it can be used for the prediction of occurrences of non-explicitly specified events. For example, if we know that somebody has taken a seat in restaurant, we can infer that the person has selected and entered a restaurant and that later on he/she will choose and order food.

Frames

These are basically a *structure* (see Figure 9.6, Chapter 9) for holding various types of knowledge about a class of similar *objects/concepts.* In this sense, a class describes a prototypical object [McMahon et al. 1993]. As stated by [Walters et al. 1988] some authors refer to frame-like structures as *units*, *objects*, *concepts*, *schema* or *entities*. Frames are given names, with the presumption that the knowledge contained within the frame is interrelated. Thus frames can represent a relatively large chunk of knowledge about a particular real world object (e.g. a car) describing it in the required detail (e.g. has four wheels, doors, an engine etc.).

Frame slots

In frames *'slots'* are used for knowledge storage. Slots describe some *attribute* such as name of car, manufacturer, model etc. which can have a *value* of different types e.g. symbolic or numeric. Consequently, two frames can represent different objects e.g. a sedan car or a lorry. Mechanisms are available for restricting the content *type* of a slot (e.g. numeric integer values only) or the *range* (minimum and maximum).

Relationships between frames

Slots can also represent *relationships* between frames. One such relationship is that of refinement, with related frames being more *specific* or *general.* This makes it possible to arrange frames into classes, sub-classes and instances i.e. frames can be structured into *kind_of* taxonomies. A specific frame stores knowledge describing a specific class of object (e.g. fuel bowsers). A general frame stores knowledge common to a class of objects (e.g. the class vehicles). An *instance* exists when no further subdivisions are possible without loosing the concept of a class that represents multiple objects – it is *a* specific case. Such a network of interconnected frames can be considered as a *semantic network* [Rychener 1988]. Most frame systems allow for slot attributes and values to be filled in by *inheritance* rather than by specifically declaring attributes or values [Rychener 1988]. Relating frames with *'part_of'* relationships makes frames also useful for representing a compositional solution model.

Static knowledge

Frames enable knowledge about an object to be stored at a fixed state, or what is known as a perspective. For example, a frame about a material would contain slots storing knowledge about the material's properties. However, in the real world, such properties can change with for example a change in the environmental temperature in which the material is being employed. Thus frames are useful for representing static, declarative knowledge [Rychener 1988]. A *relationship* between different frames is similarly represented in a

static way. Thus as argued by (Kerr et al. 1992), the process of organizing concepts into *kind_of* taxonomies is implicitly based on some *perspective*. From this Ph.D. research point of view, this limitation is acceptable because at any one time, the designer would be making use of one type of perspective.

Embedded functions - methods

Certain tools allow slots to also contain *functions* this permitting procedural type knowledge to be incorporated within the frame representation [Walters et al. 1988]. This is the case with the use of a frame representation employed in a style of programming termed *Object-Oriented Programming (OOP)* [Rychener 1988]. As argued in [Walters et al. 1988], OOP is inextricably intertwined with a frame-based representation. In OOP systems, objects have associated with them knowledge about how they behave. Rather than a function, such procedural type knowledge associated with objects is termed a *method* in OOP [Walters et al. 1988; Coyne et al. 1989].

D.2 Procedural Schemes

Procedural knowledge about the world concerns knowledge about *what* to do, *when* and *how*. The most common way or representing procedural knowledge in a computer is through programming languages such as PASCAL, LISP and C. Unfortunately, this way of representing procedural knowledge gets low scores with respect to the properties of *inferential* and *knowledge management* adequacy [Rich et al. 1991]. Because of these difficulties in representing procedural knowledge, attempts have been made to find other ways of representation. The most commonly used technique for doing this is the use of *production rules* [Coyne et al. 1989; McMahon et al. 1993].

Production rules

These provide knowledge representation scheme for building a set of transformations. The rules have essentially two parts [Popovic et al. 1994]:

- A *conditional* part on the left-hand side (LHS) showing when the rule should be applied;
- a *transformational* or *actional* part on the right hand side (RHS) showing the actions to be taken *whenever* the conditional part of the rule is applicable.

Thus, each production rule can be seen as a statement of the form in (D.1) :

Rule D.1

IF < a given condition is true>

THEN < an action(s) has to be undertaken>

The *action-part* of a rule is generally a list of actions to be performed to the objects/facts stored in the *working memory* of the knowledge-based system. Such actions could include addition, deletion and/or modification of objects/facts stored in the working memory.

Certainty

Some inference engines permit the use of a *certainty factor (CF)* to be associated with each rule [Walters et al. 1988], of the form:

Rule D.2

IF < a given condition is true>

THEN < an action(s) has to be undertaken> **CF**

meaning that if the *conditional part* is true, the *actional part* is usually true for **CF%** of the time e.g. 60%.

Appendix E – FORESEE Implementation Files

Scope

The name and content of the different FORESEE implementation files are outlined in Tables E.1, E.2, E.3 and E.4. The scope of providing information about these files is to support other researchers interested in reusing the FORESEE prototype system as a test-bed for further research.

Table E.1 FORESEE system files

Filename	**Content description**
main.clp	a batch file that loads other CLIPS code files;
main_new.clp	this file is loaded if the design session is new;
main_old.clp	this file loads any classes and instances created in a previous session;
master.htm	this contains HTLM master code for initializing the LCC browser;
FORESEE.clp	this contains the clips code for the user interface
General.clp	this defines system variables and a set of functions for the *Consequence browser, Performance Measures, Library access module, Session history module, Multi-X behaviour module, Solution model viewer, Solution model manipulator, Solution filing functions.*

Table E.2 – Synthesis element models & *Kind_of* taxonomy definition files

Filename	**Content Description**
cl_assem.clp	Contains the definition of assembly feature models and taxonomy
cl_form.clp	Contains the definition of form feature models and taxonomy
cl_manuf.clp	Contains the definition of manufacturing process models and taxonomy
cl_mater.clp	Contains the definition of engineering material models and taxonomy
cl_pdo.clp	Definition of various PDE/LCPE specific to the implemented case-study

Table E.3 – Files defining default instances available with FORESEE prototype

Filename	**Content Description**
in_assem.clp	Definition of default assembly feature instances
in_manuf.clp	Definition of default fabrication process instances
in_mater.clp	Definition of default engineering material instances
in_pdo.clp	Definition of a set of default PDE/LCPEs e.g. standard mould parts

Table E.4– Knowledge base filenames

Filename	**Content Description**
rule_ass.clp	Consequence inference and action rules concerning assembly features
rule_form.clp	Consequence inference and action rules concerning form features
rule_mat.clp	Consequence inference and action rules concerning material PDEs
rule_mold.clp	Consequence inference and action rules concerning mould design & construction
rule_pdo.clp	Consequence inference and action rules concerning concurrent synthesis
rule_pds.clp	Consequence inference and action rules concerning requirements (PDS)
rule_proc.clp	Consequence inference and action rules concerning fabrication processes
rule_sf.clp	Consequence inference and action rules concerning surface finish

Appendix F – Typical Synthesis Element Representation

Scope

This section provides an example of how a *synthesis element* model is represented with frames in the FORESEE prototype. The representation uses the COOL[1] syntax. As demonstrated, the relationship of a synthesis element to other elements in the respective *kind_of* taxonomy is defined by the *'is-a'* element of the *defclass construct* provided by COOL. Figure F.1 discloses the default classes defined in FORESEE, used in declaring synthesis element models as those enclosed in this Appendix.

Definition of top-most class in the synthesis element library

```
(defclass FORESEE (is-a USER)
    (role concrete)
    (pattern-match reactive)
    (slot part_of  (create-accessor read-write)
                   (type SYMBOL) (visibility public) )
)
```

Definition of Fasteners

Frame representation for class 'FASTENER'

```
(defclass FASTENER (is-a NON-PERMANENT))
    (defmessage-handler FASTENER init after ()
    (slot-insert$ ?self consequences 1 requires_hole
                                       assembly_slow
                                       increases_number_of_components
                                       bonds_dis-similar_materials
                                       complex_assembly_automation)
    (slot-insert$ ?self tech_properties 1 ductile)
    (if (next-handlerp) then (call-next-handler))
)
```

Taxonomy sub-classes definition

```
(defclass SCREW (is-a FASTENER))
(defclass SELF_TAP_SCREW (is-a SCREW))
```

Definition of sub-class specific LCC_{ni}

```
(defmessage-handler SELF_TAP_SCREW init after ()
    (slot-insert$ ?self consequences  bad_for_repetitive_ass)
    (if (next-handlerp) then (call-next-handler))
)
```

Note that the frame for a FASTENER does not have the slot *'part_of'* defined as part of its model, as this is inherited from the top-most class FORESEE in the synthesis elements library, quoted above.

[1] COOL: CLIPS Object Oriented Language - see [Giarratano et al. 1994; CLIPS 1997].

Definition of A Circular Opening

Sample CLIPS code representation

```
(defclass OPEN_FORM (is-a FORM)
  (role concrete)
  (pattern-match reactive)
  (slot depth  (create-accessor read-write) (visibility public) )
  (slot shape  (create-accessor read-write) (visibility public) )
  (slot angle     (create-accessor read-write) (visibility public)(type
NUMBER) )
  (slot x_centre  (create-accessor read-write) (visibility public) )
  (slot y_centre  (create-accessor read-write) (visibility public) )
)
```

Consequence accumulation

```
(defmessage-handler OPEN_FORM init after ()
   (slot-insert$ ?self consequences 1 less_volume)
   (if (next-handlerp) then (call-next-handler))
)
```

Sub-class definition

```
(defclass CIRCULAR_OPEN (is-a OPEN_FORM)
    (role concrete)
    (pattern-match reactive)
    (slot depth  (create-accessor read-write) (visibility public) )
    (slot diameter  (create-accessor read-write)
                (visibility public)
                (type NUMBER)
                (default 0.00)
   )
    (slot angle     (create-accessor read-write) (visibility public)(type
NUMBER) )
    (slot x_centre  (create-accessor read-write) (visibility public) )
    (slot y_centre  (create-accessor read-write) (visibility public) )
)
```

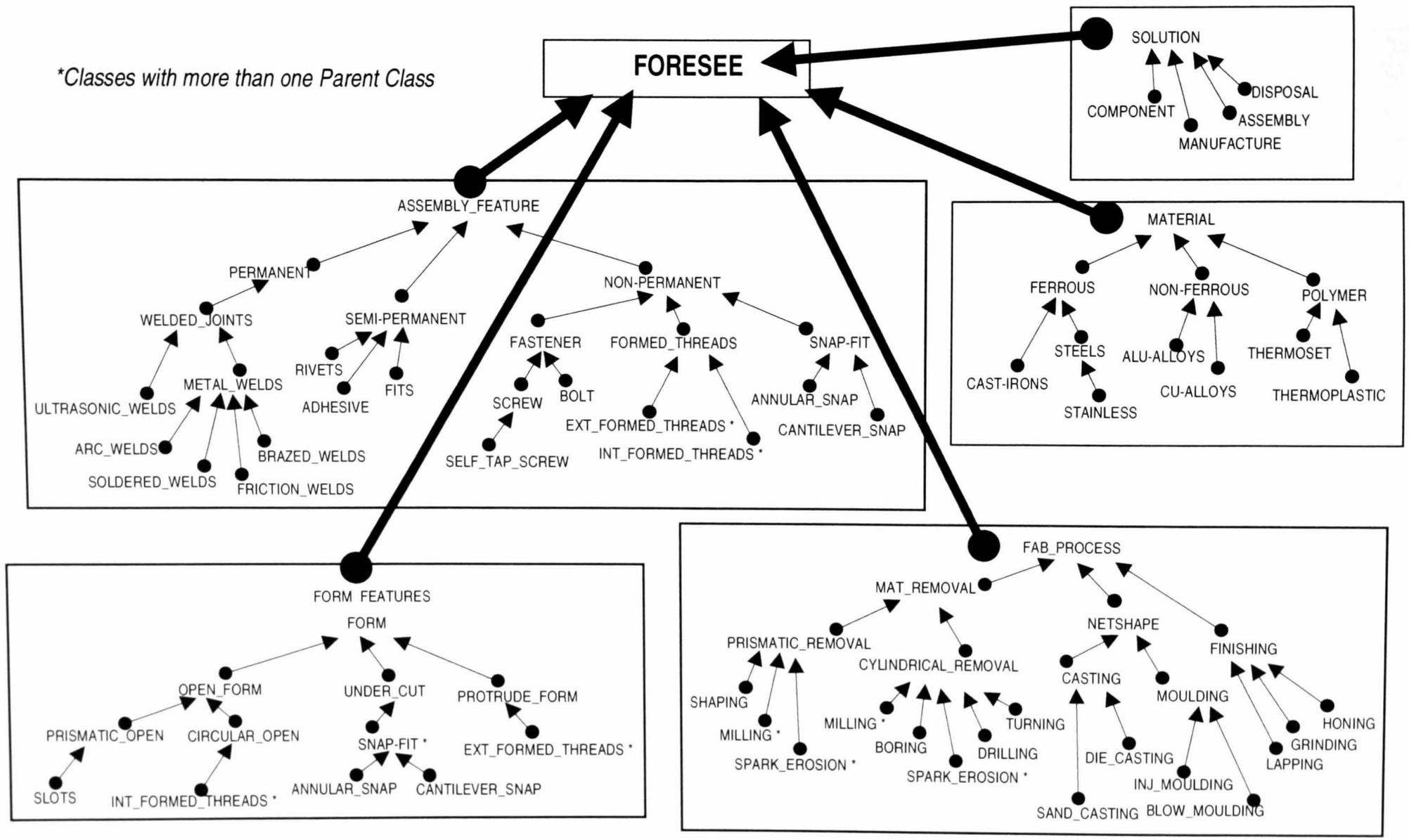


Figure F.1 – Default Classes Defined in FORESEE

Appendix G – Performance Measure Functions

As outlined in relationship (8.5), this reproduced below, a LCC can influence a performance measure of a life-phase technical process:

(8.5) (LCC) => (δPM)[P]

Relationship 8.5 has been implemented in FORESEE as part of the 'actional part' of a production rule, this principle explained in G.1.

(G.1) **IF** *(LCC)$_j$* **THEN** *pm_function$_j$ (x)*

where *pm_function$_j$ (x)* causes a relative change of value (x) in a performance measure (stored as a variable in the computer's memory) of a life-phase process due to $(LCC)_j$. A number of such *pm_function$_j$ (x)* functions[2] have been implemented for *design, assembly, manufacturing, use, service* and *disposal*. Examples of assembly *(ass_)* related functions are listed in Table G.1

Table G.1 – Typical Performance Measure Functions

pm_function$_j$ (x)	**Function's action**
ass_time_plus(x)	Increases the value of the assembly time by 'x' units
ass_time_minus(x)	Decreases the value of the assembly time by 'x' units
ass_cost_plus(x)	Increases the value of the assembly cost by 'x' units
ass_cost_minus(x)	Decreases the value of the assembly cost by 'x' units
ass_qual_plus(x)	Increases the value of the assembly quality by 'x' units
ass_qual_minus(x)	Decreases the value of the assembly quality by 'x' units

[2] A *function* in CLIPS is a set of instructions defined with a procedural programming paradigm similar to that found in languages such as C, Pascal and LISP.

Appendix H – LCC Truth Maintenance Mechanism

Scope

Due to some synthesis decision commitment retraction, a LCC will not be true anymore. In such situations, the KICAD tool should automatically update the list of LCCs associated with the current solution state. This appendix describes the mechanism of how LCC knowledge has been represented to address this truth maintenance requirement.

Production rules

Basically a production rule (see Appendix D) representing LCC inference knowledge has *conditional elements* on the left-hand side (LHS) of the rule and a list of *actions* to be performed on the right hand side (RHS). Using *IF...THEN*[3] syntax, this is formally:

(H.1) **IF** <conditional elements> **THEN** <list of actions>.

Based on the formalism in Chapter 8, in the case of a LCC_{ni}, the conditional element is a synthesis commitment. In the case of LCC_i, a conditional element is an interacting relationship (IR). The *list of actions* will be the co-evolution of a new fact i.e. the LCC_{ni} or LCC_i respectively. Consider the simple LCC_i example in (H.2):

(H.2) **IF** $(A \wedge B \wedge C)$ **THEN** new fact is (LCC='z')

If at any time, the interacting relationship $(A \wedge B \wedge C)$ exists, then the fact (LCC='z') is added to the fact list. Then excluding other facts that may be present in the *working memory*, the current facts list will be:

(H.3) [facts list: A, B, C, (LCC='z')]

Assuming that for some reason, the synthesis element 'A' is retracted, by the designer, than the new current fact list will be:

(H.4) [facts list: B, C, (LCC='z')]

Now the facts list state in (H.4) contradicts the piece of LCC inference knowledge in (H.2) as the LHS of the rule is *not true* anymore. To cater for such truth maintenance, wxCLIPS supports *logical dependency* in production rules, of the form:

(H.5) **IF <logical>** <conditional elements> **THEN** <list of actions>

[3] The *IF...THEN* syntax is not wxCLIPS syntax - this is being used for explanation purposes. For wxCLIPS representation syntax, see Appendix J.

By the use of the additional condition *<logical>*, the resulting new facts generated by the *<list of actions>* become *logically dependent* on the pattern of the *<conditional elements>* i.e. if the pattern is not true anymore, than the facts introduced by the *<list of actions>* will be retracted. Building up on (H.2), we can say:

(H.6) - Detection

IF <logical>(A ∧ B ∧ C) **THEN** new fact is (LCC='z')

If at any time, the interacting relationship (A ∧ B ∧ C) exists, then the fact (LCC='z') is added to the fact list, this *now* being *logically dependent* on the elements forming the interacting relationship on the LHS of rule (H.6). Then, excluding other facts that may be present in the working memory, the current facts list will be as in (H.3). If this time element 'A' is retracted, then due to logical dependence, fact (LCC='z') will also be retracted, with the fact list becoming now as an (H.7):

(H.7)

[facts list: B, C,]

Limitations

However logical dependency support provided by CLIPS limits the truth maintenance capability required with FORESEE. This is because as discussed in Chapter 9, LCC knowledge embedded in FORESEE includes consequence *action* knowledge that cause fluctuations in performance measures, evolves models through concurrent synthesis, updates the consequence list in the hypermedia browser etc. *Actions* executed are not facts stored in the working memory. Thus, retracting a logically dependent consequence fact is not enough if actions executed are to be maintained. A mechanism is required that takes care to *undo* the actions performed when a consequence fact is retracted. The problem is that this truth maintenance mechanism will need to execute when the consequence fact is *not* present anymore, this causing a difficulty with production rule pattern matching.

Mechanism

This section demonstrates the mechanism of how knowledge was represented in FORESEE to maintain the truth of such *LCC action knowledge*. The mechanism developed is based on segmenting LCC knowledge into three parts:

i. a rule *detecting* the existence of a co-evolving LCC – this similar in structure to that in (H.6);
ii. a rule performing *actions* due to the existence of a LCC, this similar in structure to that in (H.8);
iii. a rule undoing actions due to the retraction of a LCC i.e. performing *truth maintenance* this similar in structure to that in (H.9);

(H.8)
Action rule

IF fact is (LCC='z')

THEN

- display relevant message to attract designer's attention;
- create a LCC entry in the *consequence browser* with details about the consequence including meaning, source commitments and guidance to its avoidance/relaxation;
- record consequence & details in *session history* file;
- make the relevant fluctuations to performance measures e.g. *ass_time_plus (5);*
- make consequence specific actions (e.g. evolve a model)
- add a fact of the type **retract_(LCC=z)** for action truth maintenance.

(H.9)
Action truth maintenance rule

IF fact (LCC='z') is not present in the facts list

AND fact *retract_(LCC=z)* is present in the facts list

THEN

- delete relevant LCC entry from the *consequence browser*;
- record retraction of consequence in *session history* file;
- undo the relevant fluctuations to relevant performance measures e.g. *ass_time_plus (5);*
- undo consequence specific actions (e.g. delete an element from a model);
- remove the fact **retract_(LCC=z)** from the working memory.

Mechanism Operation

When all elements forming part of an interacting relationship are present in the evolving artefact life model, *detection rule H.7* is fired, generating facts as in H.4. This new state of facts causes *action rule H.11* to fire, resulting besides a number of actions in the state of the facts list as:

(H.10) [facts list: A, B, C, (LCC='z'), retract_(LCC='z')]

If for some exploratory reason, retracts the commitment 'A', due to logical dependency in H.6, the fact (LCC='z') is retracted from (H.10), with the current facts list now becoming:

(H.11) [facts list: B, C, retract_(LCC='z')]

This new state in (H.11) causes the action truth maintenance rule (H.9) to fire, this undoing the actions performed with action rule (H.8). As structured, the truth maintenance rule (H.9) will only fire when *(LCC='z')* is *not* present *but* on the condition that fact *retract_(LCC='z') is* present. In rule (H.8) the fact *retract_(LCC='z')* is purposely not logically dependent on the fact *(LCC='z')*. If this were the case, then the retraction of fact *(LCC='z')* will be accompanied by the retraction of fact *retract_(LCC='z')* in which case the truth maintenance rule (H.9) would never fire.

Appendix I – LCC Browser Maintenance

Problem

The *LCC browser* plays an important role in the FORESEE architecture. For appropriate support, the designer needs to be provided with LCCs *only* relevant to the current solution state, meaning that if any synthesis decision commitments are retracted, related LCCs should be retracted and deleted from the list displayed in the LCC browser.

Implemented solution

Maintaining the truth of the LCC list in the LCC browser required the implementation of a function termed *html_tms (?csq)* described in the flowchart in Figure I.1. Basically, a LCC action rule (see rule H.8) introduces a truth maintenance fact with a unique number #NO, of the form:

(I.1)

retract (LCC=z) #NO

When an action truth maintenance rule (see H.9 in Appendix H) executes, the function *html_tms (?csq)* is executed with the argument *?csq* being #NO, this causing the LCC list to be appropriately maintained. Typical examples are listed in Appendix J.

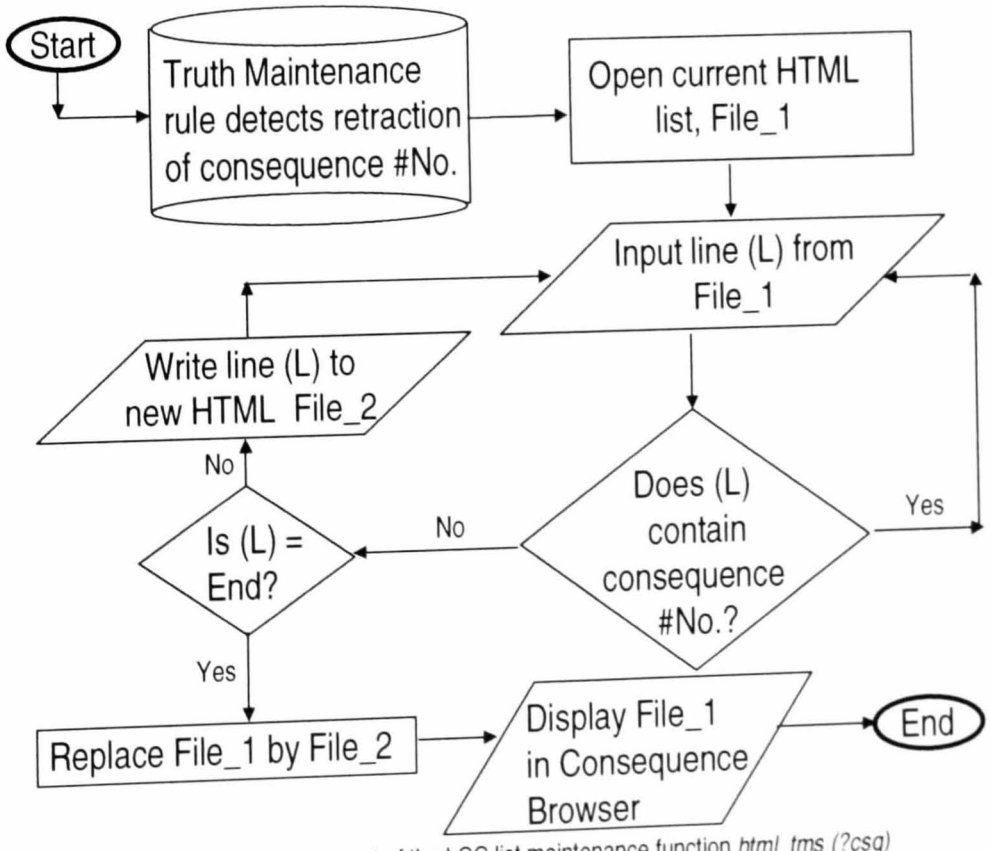


Figure I.1 – The concept of the LCC list maintenance function *html_tms (?csq)*

Appendix J – LCC Knowledge Representation

Scope

This Appendix presents examples of how LCC knowledge employing the truth maintenance mechanisms discussed in Appendices H and I, are actually represented in FORESEE with the COOL syntax (CLIPS Object-Oriented Langauge – see [Giarratano et al. 1994; CLIPS 1997]).

Example

The first example quoted is that of a sink mark defect, this based on the LCC_i knowledge structuring concept presented in relationship (8.61).

RULE J.1 ***Consequence Inference (Detection) Rule***

Description — *CLIPS code*

Rule name

```
(defrule sense_sink_marks
```

Dependency

```
(logical
    (object (is-a THERMOPLASTIC) (name ?mname) )
        (material-used (name ?mat_name) )
        (test (eq ?mname (symbol-to-instance-name ?mat_name) ) )
```

Interacting commitments

```
        (object (is-a PROTRUDE_FORM)(name ?fname) )
        (form_featureused ?fname2 ?feat_id )
        (test (eq ?fname (symbol-to-instance-name
                    (string-to-symbol (str-cat ?fname2 "_" ?feat_id) )  ) )
        )
        (process-used (name ?NAME1) )
        (object (is-a INJ_MOULDING) (name ?pname) )
        (test (eq ?pname (symbol-to-instance-name ?NAME1) ) )
    )
```

Inferred LCC_i

```
    =>
        (assert (consequence ?fname ?feat_id ?mname ?pname
            sink_mark_defect))
        (record_consq "SINK_MARK_DEFECT")
)
```

RULE J.2 ***Consequence Action Rule***

Description — *CLIPS code*

Rule name

```
( defrule sink_mark_defect_formation
```

LCC_i detected

```
    (consequence $?anything sink_mark_defect)
=>
```

Explanation on consequence by reporting related information into consequence browser and session history

```
(begin_warning "SINK MARK DEFECTS")
 (printout consq "  Because of the interacting commitments, " crlf)
 (printout consq "  " $?anything crlf)
 (printout consq "  then Sink Marks can form.  This will depend" crlf)
 (printout consq "  on the rib-height:component_wall_thickness" crlf)
 (printout consq "  ratio.  The greater the rib-height, the" crlf)
 (printout consq "  more chances of sink marks forming." crlf)
 (printout consq "         [Dow Plastics Design Guideline]" crlf)
 (printout consq " " crlf)
 (printout consq "  This can have an influence on the: " crlf)
 (printout consq "  " crlf)

 (printout consq " USE "  crlf)
 (printout consq " Reason: " )
 (printout consq " If component needs to have a cosmetic finish " crlf)
 (printout consq " or a good structural performance... " crlf)
 (printout consq "   " crlf)

 (printout consq " MANUFACTURE " crlf)
 (printout consq " Reason: " )
 (printout consq " If the component " (upcase ?*isem_disinn*) " requires a good finish," crlf)
 (printout consq " then more components need to be produced and the defective " crlf)
 (printout consq " ones scrapped..... " crlf)
 (printout consq "   " crlf)

 (printout consq " DISPOSAL " crlf)
 (printout consq " Reason: " )
 (printout consq " More defective parts to be disposed/recycled..." crlf)
 (printout consq "   " crlf)
(end_warning sinkmark.gif 400 300)
```

Performance measure fluctuations

```
(use_qual_minus 9)
(use_time_minus 6)
(manuf_time_plus 5)
(manuf_cost_plus 4)
(manuf_qual_minus 4)
(disp_cost_plus 7)
(disp_time_plus 7)
```

Fact for action truth maintenance

```
(assert (retract sink_mark_defect # ?*consq_no*))
```

Display fluctuations in Mult-X behaviour window

```
(show_pm)
)
```

RULE J.3 ***Truth Maintenance Rule***

Description	*CLIPS code*
Rule name	(defrule truth_sink_mark_defect (declare (salience 5000))
Pattern detecting LCC$_i$ retraction	(**not** (consequence $?anything **sink_mark_defect**)) ?f1<- (**retract sink_mark_defect** # **?csq**)
	=>
LCC fact maintenance	(retract_fact SINK_MARK_DEFECT)
Consequence list maintenance	(html_tms ?csq)
Maintenance of performance measures	(use_qual_minus -9) (use_time_minus -6) (manuf_time_plus -5) (manuf_cost_plus -4) (manuf_qual_minus -4) (disp_cost_plus -7) (disp_time_plus -7)
Display updated fluctuations in Mult-X behaviour window	(show_pm) (retract ?f1))

Typical Concurrent Synthesis Knowledge Representation

Example background

Rules J.4, J.5 and J.6 demonstrate a typical representation of concurrent synthesis knowledge. It concerns the addition of a core-pin once a form feature, a *kind_of* CIRCULAR_OPEN is added to a thermoplastic component that is going to be injection moulded.

RULE J.4 ***Consequence Inference (Detection) Rule***

Description	*CLIPS code*
Rule name	; Due to mould required and circular opening form feature ; a Core-pin is Required in the mould tool ; ********** (defrule sense_mould_tool_requires_core_pin

Dependency

```
(logical (consequence ?anything mould_required)

        (object (is-a CIRCULAR_OPEN)(name ?fname) )
            (or  (form_featureused ?fname2 ?feat_id )
                 (ass_featureused (name ?fname2) (pdo_id ?feat_id) )
            )
            (test (eq ?fname (symbol-to-instance-name
                        (string-to-symbol (str-cat ?fname2 "_" ?feat_id) )))
            )
    )

    =>

        (assert (consequence mould_required ?fname ?feat_id core-pin_required ))

        (record_consq "CORE-PIN REQUIRED")
        ;;; Check available suppliers & their diameter ranges
        (assert (check_core-pin_suppliers))
        (run)
)
```

Pattern detecting concurrent synthesis

New LCC fact

Keep a record in the Session History

RULE J.5 Consequence Action Rule

Description

CLIPS code

Rule name

```
(defrule action_mould_requires_core-pin

(consequence mould_required ?fname ?feat_id core-pin_required )

=>
    (begin_warning (str-cat "CORE" ?feat_id " IS A CORE-PIN") )
     (printout consq "  Since the feature " ?fname "   added, "  crlf)
     (printout consq "  has a circular cross-section, then" crlf)
     (printout consq "  the core required is a core-pin." crlf)
     (printout consq " " crlf)
     (printout consq "  *************************" crlf)
     (printout consq "  Design Guidance" crlf)
     (printout consq "  *************************" crlf)
     (printout consq "  Core-pins are available in standard sizes" crlf)
     (printout consq "  sizes from suppliers such as HASCO and" crlf)
     (printout consq "  DME.  It makes sense to use a standard core-pin" crlf)
     (printout consq "  as it avoids machining." crlf)
     (printout consq " " crlf)
     (printout consq "  As a result, FORESEE suggests the definition of" crlf)
     (printout consq "  a hole diameter value in the component that" crlf)
     (printout consq "  when material shrinkage factor is taken into" crlf)
     (printout consq "  consideration yields a standard core-pin diameter." crlf)
     (printout consq "  " crlf)
     (end_warning kcm.bmp 100 100)
     (run)
```

Detection of concurrent synthesis fact

Reporting of 'concurrent synthesis' consequence state and related information into consequence browser

Fact for PM truth maintenance

```
(assert (retract core-pin_required ?feat_id # ?*consq_no*))
(run)
```

Action that adds a 'core-pin' object to the 'mould' model

```
;; remove CORE object from mould tool
   ;;; and replace with CORE-PIN object
   (delete_pdo "CORE" ?feat_id )
   (add_copy core ?feat_id  core-pin  mould)
   (run)
```

Updates the display of the mould tool model

```
   ;;; show evolving mould design
   (consists_of mould FORESEE)
)
```

RULE J.6 ***Truth Maintenance Rule***

Description — *CLIPS code*

Rule name

```
(defrule truth_core-pin_required
```

Pattern detecting LCC retraction

```
   (not (consequence $?anything core-pin_required))
   ?f1<-(retract core-pin_required ?feat_id # ?csq)
=>
```

Records maintenance in Session History

```
   (retract_fact CORE_PIN_REQUIRED)
```

Consequence list maintenance

```
   (html_tms ?csq)
```

Maintenance of mould tool model

```
   (delete_pdo "CORE" ?feat_id )
   (retract ?f1)
```

Updates the display of the mould tool model

```
   ;;; show evolving mould design
   (consists_of mould FORESEE)
)
```

Appendix K – Evaluation Background Information

This appendix provides a copy of the background information submitted to the evaluators prior to the evaluation via the FORESEE prototype.

FORESEE System Demo

Jonathan Borg
CAD Centre
Department of Design, Manufacture & Engineering Management
University of Strathclyde
Glasgow G1 1XJ
email: jonathan@cad.strath.ac.uk

Background

The purpose of this demonstration is to evaluate a prototype implementation of a proposed '*knowledge intensive approach' as* a means by which to effectively allow designers to engage in *provident* [1] thinking so as to 'optimize the conditions during production and product life'. The knowledge intensive approach, implemented here in a Knowledge Intensive CAD (KICAD) prototype nicknamed FORESEE is proposed as a means allowing designers to:

- *explore* life commitments and their interactions;
- take a *multiple-life phase* view (rather than a narrow view);
- do all this during candidate solution *synthesis* and not later during candidate design analysis.

Knowledge Application

Thermoplastic component design has been chosen to demonstrate the utility of employing the proposed knowledge intensive approach as a means supporting life design providence and exploration, as *from the early design stages.* It is however possible, with the population of more knowledge from other domains, to apply the proposed knowledge approach for component life design providence & exploration.

Feedback on Demonstration

During this demonstration, you will not be asked to use the FORESEE system yourself, as (i) it lacks a number of user-friendly functionalities and (ii) training is required in using the system. Rather, based on your impressions of the case-study and specific examples you might want to be performed, you will then be kindly asked to provide feedback on:

- the effectiveness of the knowledge intensive approach and
- the KICAD prototype implementation.

For this purpose, a few questions on the above, will be explained and discussed with yourself at the end of this demonstration.

[1] Providence is defined as "to foresee and take into account aspects which are fixated or determined at the design stage"

Appendix L – Evaluation Questionnaire

This appendix provides a sample of a filled-up questionnaire discussed during a structured interview with on of the evaluators.

4C

Structured Interview Questionnaire

Please tick where appropriate & feel free to add reasons to your answers

1. Does the FORESEE demonstrate how a designer can build up an abstract component compositional *(part_of)* model from *Product Design Objects (PDOs)*?

Yes ✓ __ No __

2. Did FORESEE make you aware of life-cycle consequences *during* component model definition?

Yes ✓ __ No __

3. Were you made aware of life-cycle consequences when the artefact model was still *imprecise? (e.g. 'opening' is less precise than a 'slot opening')*

Yes ✓ __ No __

4. Were you made aware of life-cycle consequences when the artefact model was still *incomplete? (e.g. a 'slot opening' without parameters (lxbxh) defined is incompletely defined)*

Yes ✓ __ No __

5. Were you made *aware* of consequences :

- *for one life-phase/system only* __
- *for multiple but segmented life-phases* __
- *for multiple & interacting life-phases* ✓

6. Do you consider the life-cycle consequences *awareness* provided to be:

- *only generic* __
- *company oriented (i.e. also takes into consideration company resources, policies etc.)* ✓
- *candidate design solution oriented* ✓

4C

7. Do you consider that the concept of *concurrent synthesis* of the 'artefact' and 'different life-phase models' is demonstrated by the support provided by the system?

Yes ✓ __ No __

8. Would you consider the *awareness* of life-cycle consequences:
   - to motivate you *exploring* alternative commitments?

Yes ✓ __ No __

   - to *hinder* your design freedom?

Yes __ sometimes __ No __

   - to make you eventually confirm decision commitments more *consciously*?

Yes ✓ __ No __

9. Do you find that the use of *Cost, Time & Quality* performance measures as a suitable means for comparing the *relative* influence of a consequence on the different life-phases?

Yes __ ✓ cost and time but not quality. __ No __

10. Do you find consequences detected :
   - to be *reported* in an adequate way?

Yes ✓ __ No __

   - easily to *locate* and *browse* upon?

Yes __ ✓ __ No __

11. Do you consider the system provided advice to the *avoidance*, or *relaxation* (i.e. reduction) of *consequences* ?

Yes ✓ __ ✓ __ No __

12 Do you think that keeping track of the fluctuations in performance measures for the different life phases, with different decision commitments is a useful way of formatting *life design history*?

Yes __ ✓ __ No __

Focus on major fluctuations Graphical displays.

4C

13. Does designing with FORESEE demonstrate the approach of how *"knowledge of consequences"* can be employed *during* component design synthesis?

Yes ✓ __ No __

14. Does FORESEE demonstrate how new *objects* can be added and knowledge content manually changed (i.e. not by machine learning techniques), *during* the use of the system:

Yes ✓ __ No __

15. Assuming *more knowledge* was embedded within such system, would you consider the proposed 'knowledge of consequences' approach to life design, to be useful in practice?

Yes ✓ __ No __

16. Do you have any recommendations to:

- the 'knowledge of consequences' design approach:
  Need to prioritise consequences.
- the FORESEE prototype system:
  Able to dynamically interact with other design tools such as proIE.
  Able to access other knowledge databases.
  Easily customisable.
- the demonstration/case-study:

17. If in your opinion, the demonstration highlights *other issues* not mentioned above, what are these:

4C

ABOUT YOUR DESIGN EXPERIENCE

18. Your current role: Product Design Consultant.

19. Type of Product Design Experience : *(e.g. mechanical parts, thermoplastic components etc.)*

Thermoplastic pressed metal parts, Housings, ~~etc~~ casings, Complex assemblies eg. Vaccum cleaners, machines e.g. paper handling devices.

20. Number of years working in design:

<2 years 2< years < 5 years > 5 ✓

21. Do/did you form part of a *Product Design Team*?

Yes ✓ Sometimes Varies No __

22. On average, *how frequently* are/were Product Design Team meetings held, in which *all* team members were present?

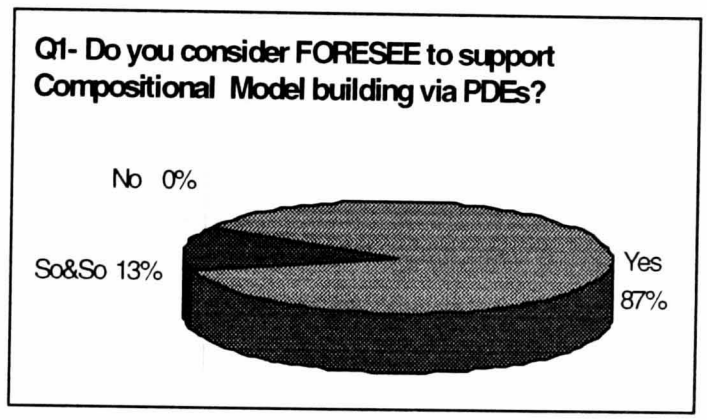
Daily Weekly Monthly Other (specify) Varies

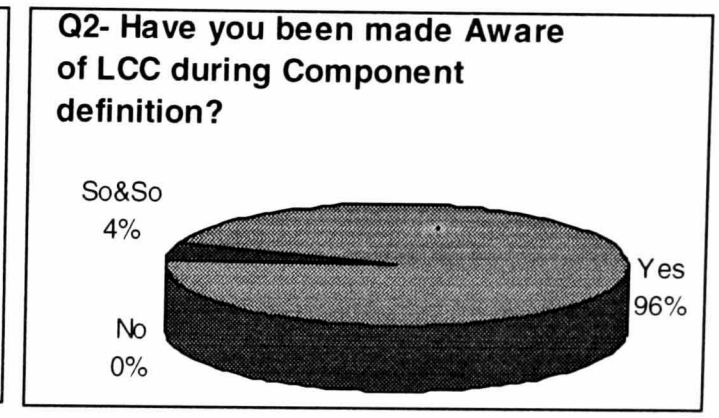
Thank you for your time in discussing and answering this questionnaire.

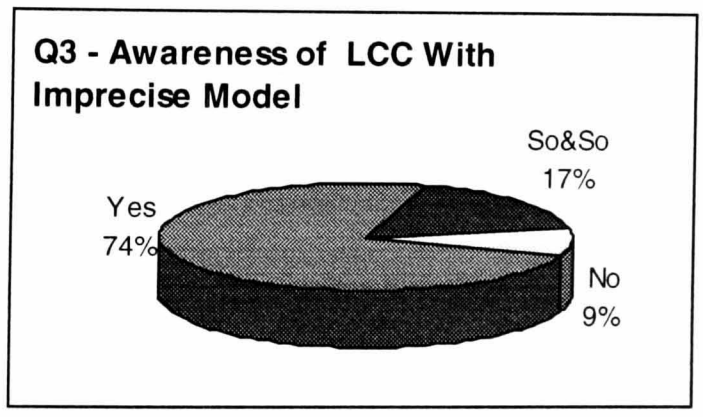
Appendix M – Evaluation Results

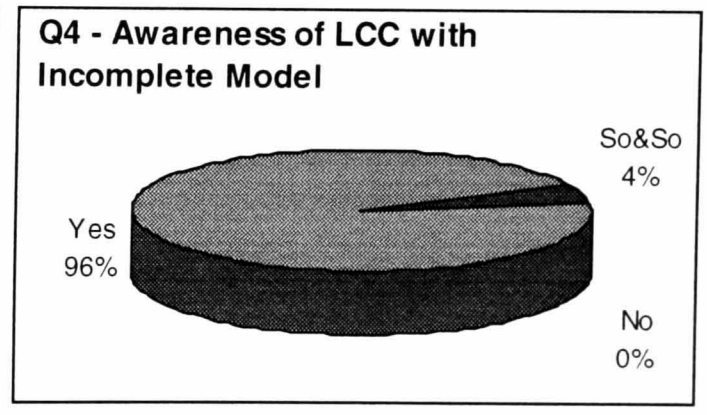
This appendix contains the demonstration evaluation results. In section M.1, results are presented globally for all the evaluation participants. Section M.2 presents separate results for the three different categories of participants involved in the exercise i.e. for *designers, researchers* and *students.* The results, presented in graphical form, relate to the questions (shown in full in Appendix L) raised during the structured interview following the FORESEE demonstration.

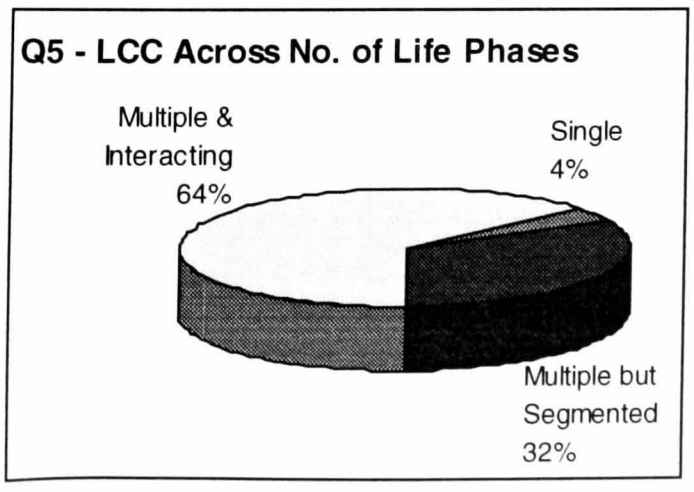
M.1 – Global Evaluation Results

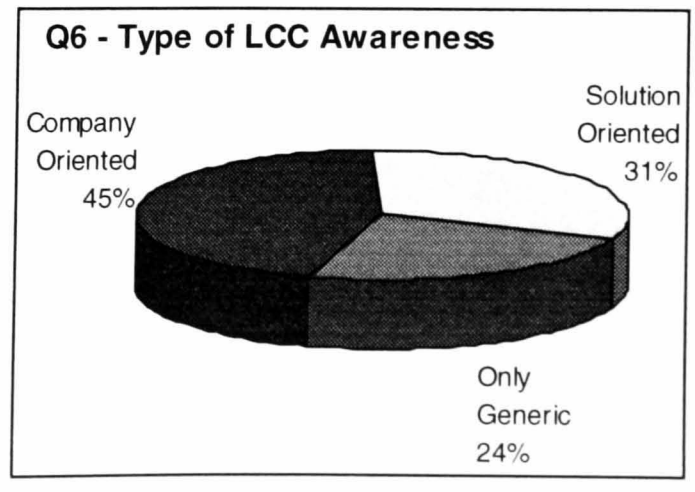


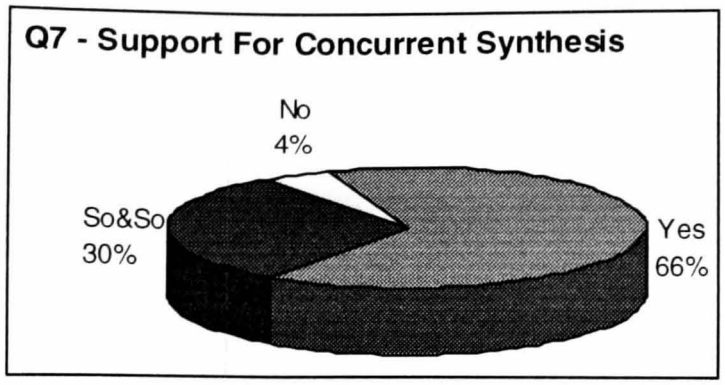


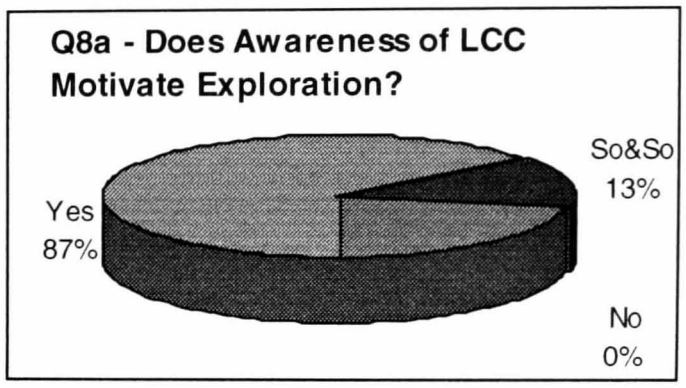


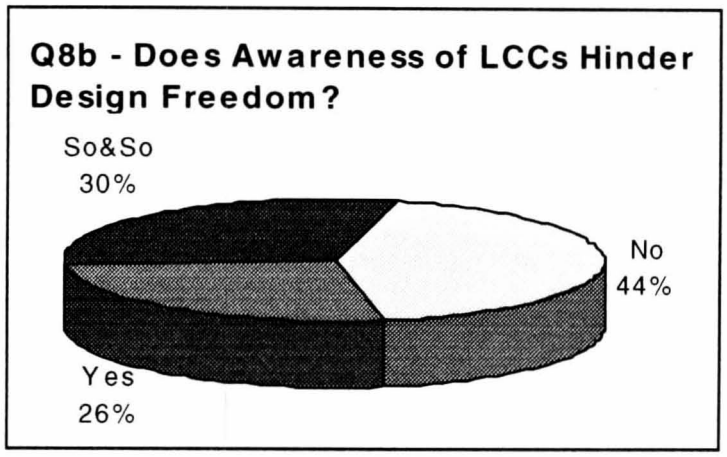


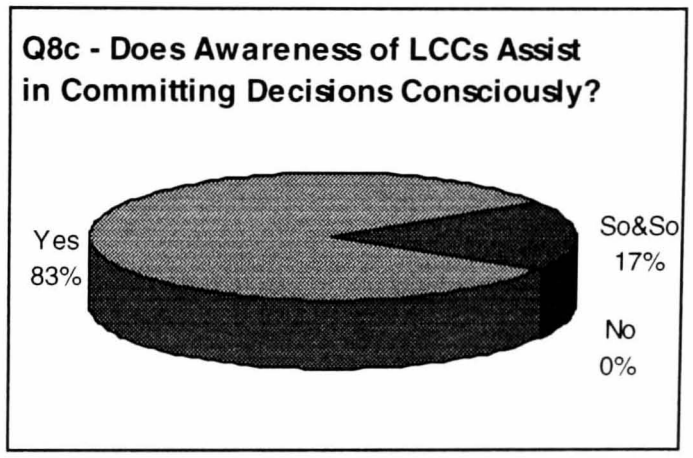





Q7 - Support For Concurrent Synthesis
No
4%
So&So
30%
Yes
66%

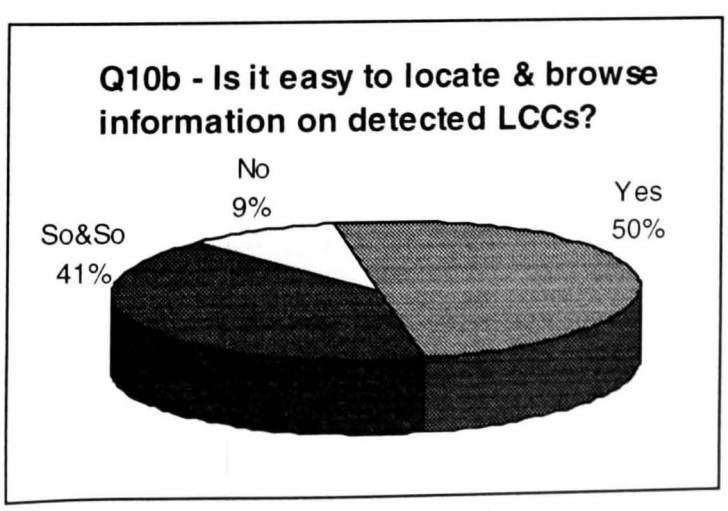

Q8a - Does Awareness of LCC Motivate Exploration?
So&So
13%
Yes
87%
No
0%

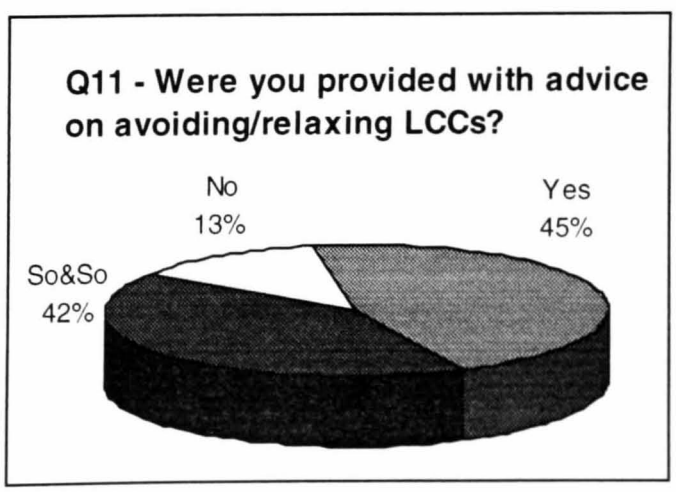

Q8b - Does Awareness of LCCs Hinder Design Freedom?
So&So
30%
No
44%
Yes
26%

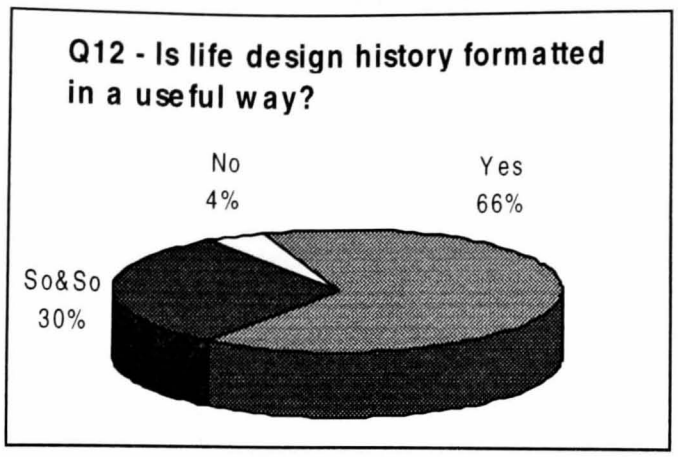

Q8c - Does Awareness of LCCs Assist in Committing Decisions Consciously?
Yes
83%
So&So
17%
No
0%

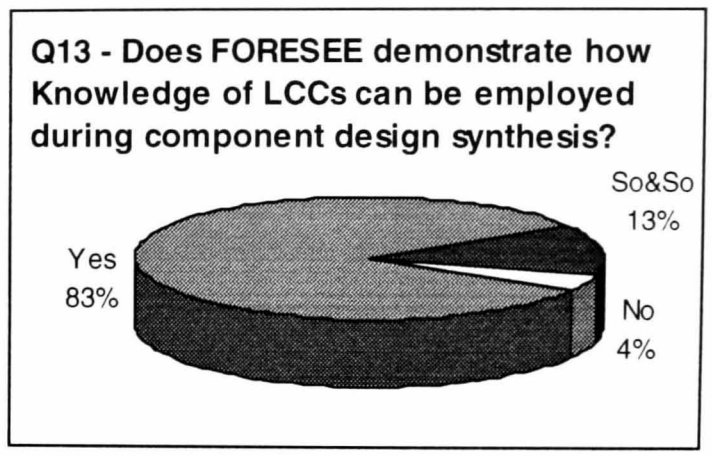
Q9 - Are Perf.Measures Suitable Metrics For Comparing the Relative Influence of LCCs?
So&So
26%
Yes
65%
No
9%

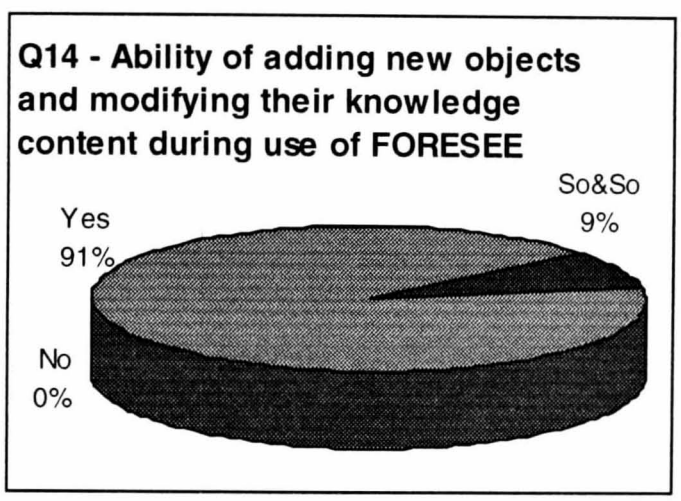
Q10a - Are LCCs reported adequately?
So&So
26%
Yes
57%
No
17%

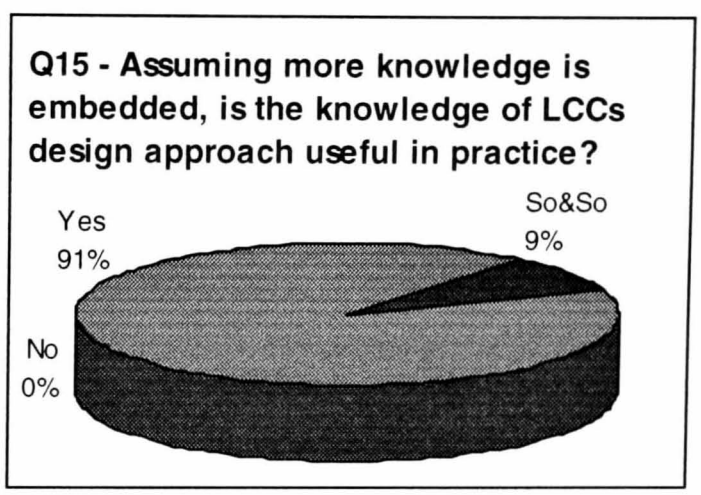

Q10b - Is it easy to locate & browse information on detected LCCs?
No
9%
Yes
50%
So&So
41%


Q11 - Were you provided with advice on avoiding/relaxing LCCs?
No
13%
Yes
45%
So&So
42%

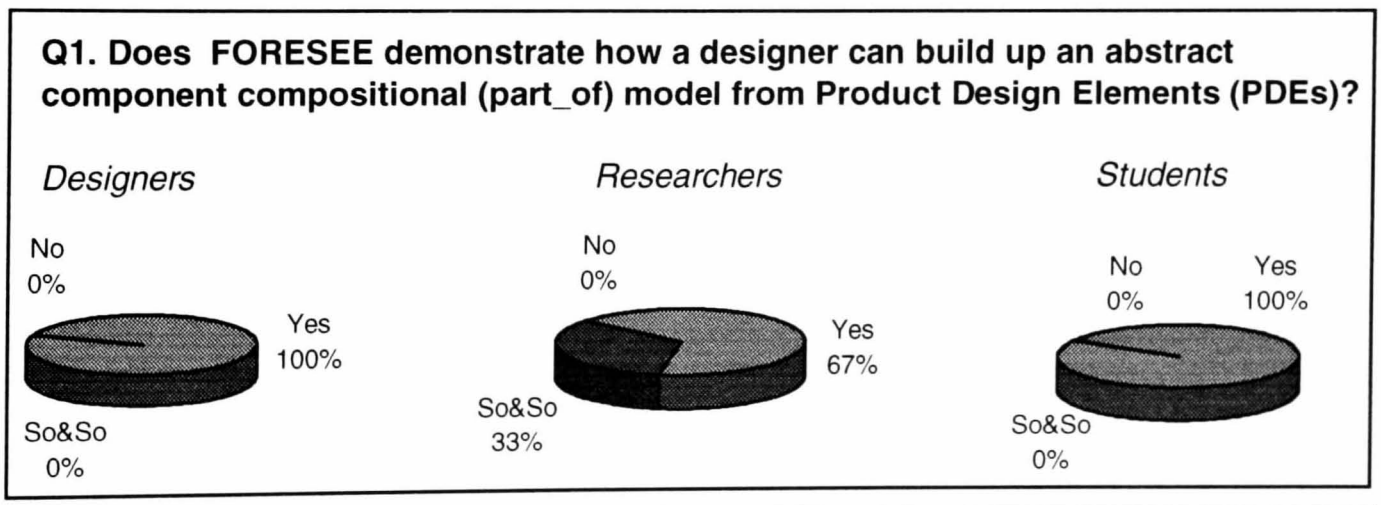


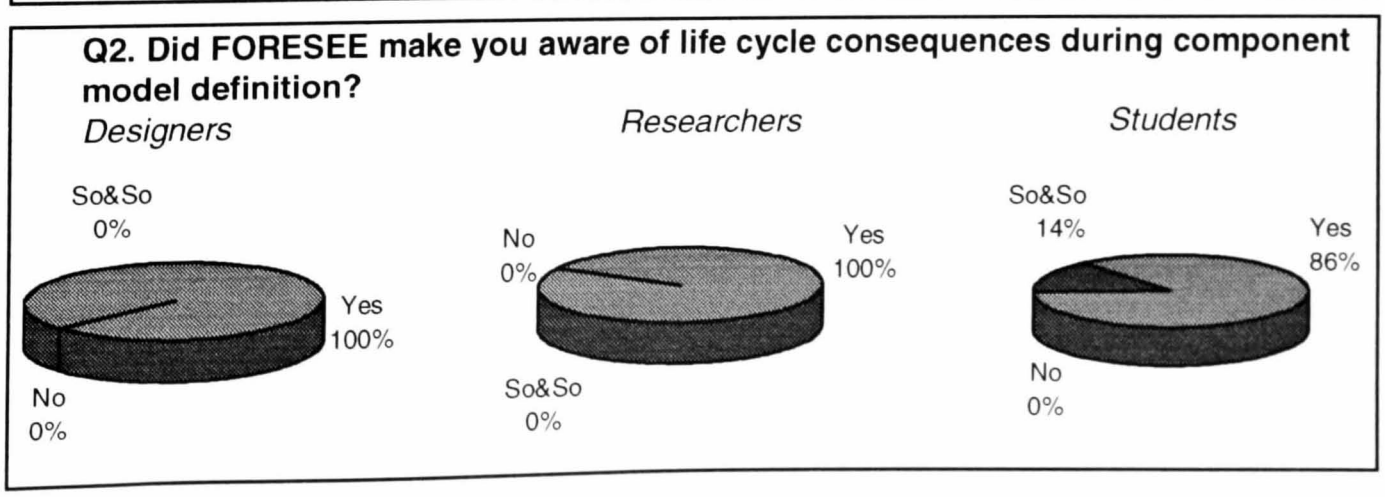




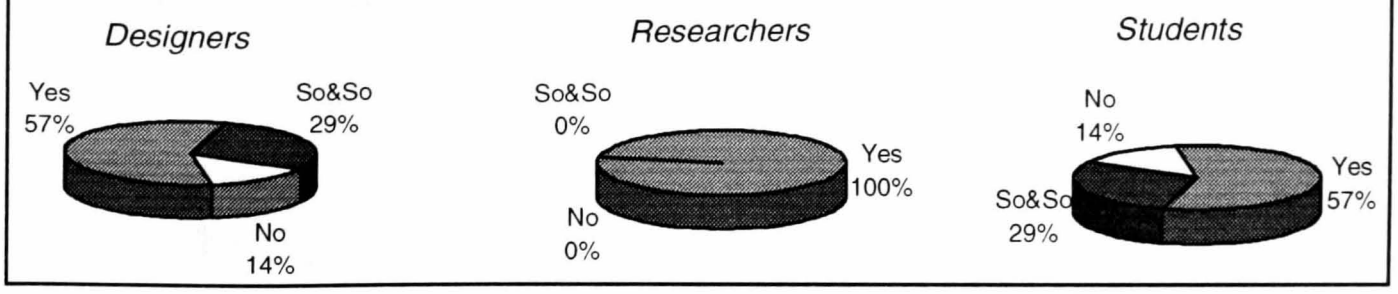


M.2 – Results by Evaluator Category

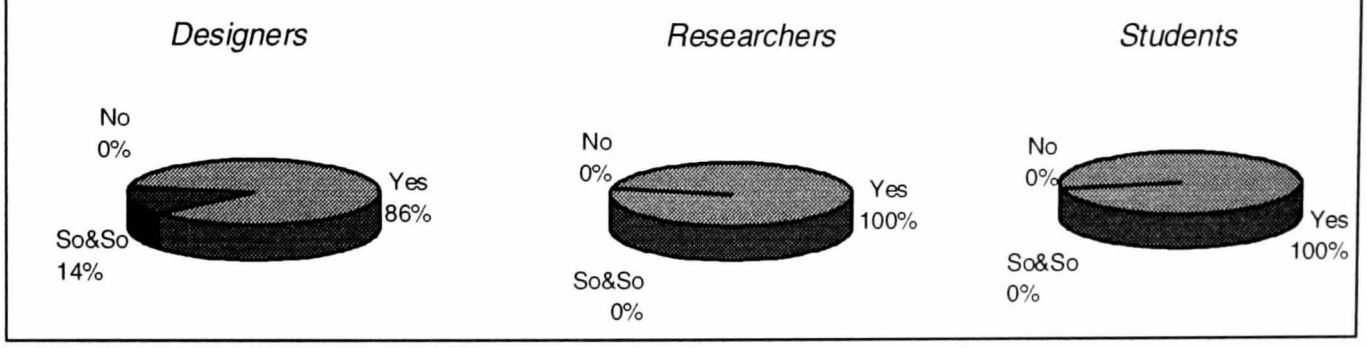




Q3. Were you made aware of life-cycle consequences when the artefact model was still imprecise?

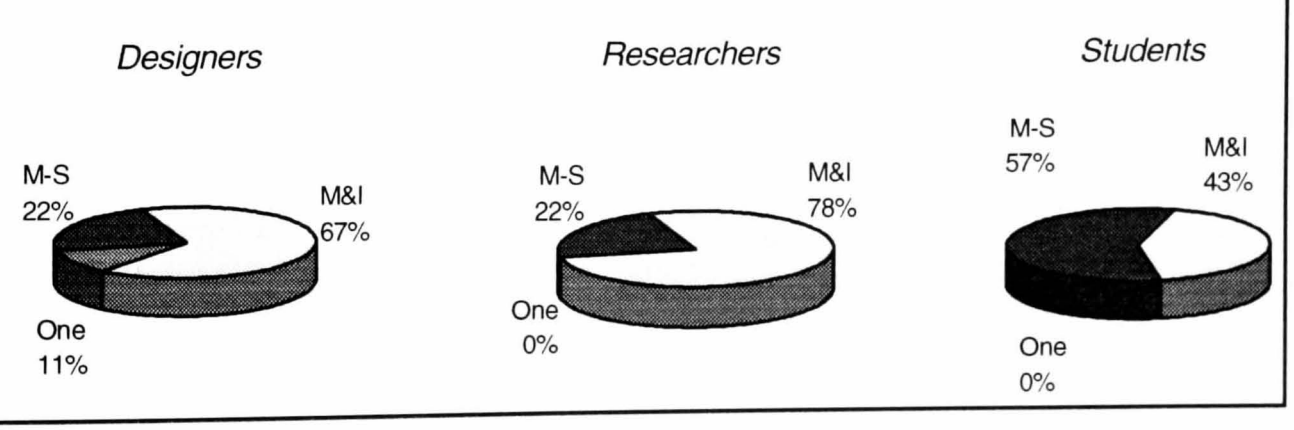


Q4. Were you made aware of life-cycle consequences when the artefact model was still incomplete?



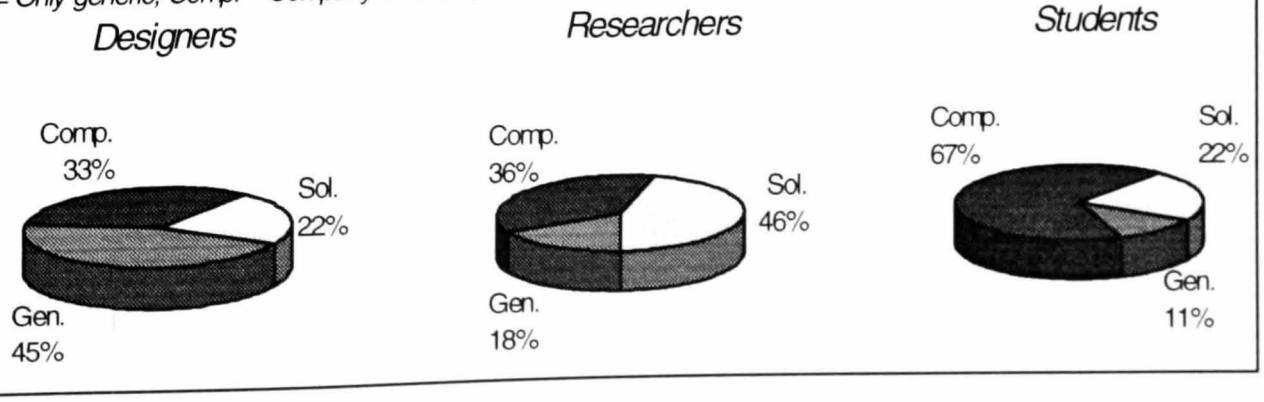
Q5. For how many phases were you made aware of consequences?

(M-S = Multiple but Segmented, M&I = Multiple & Interacting)

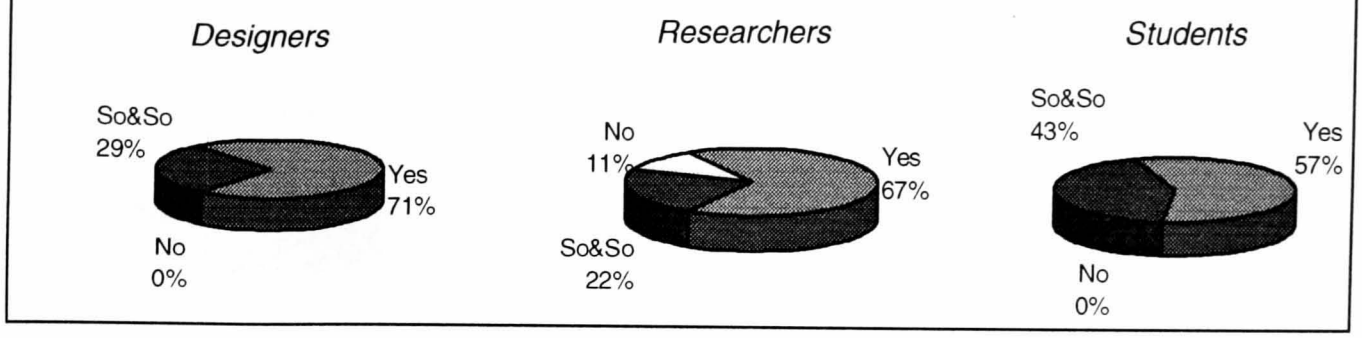


Q6. Type of life-cycle consequence awareness provided:

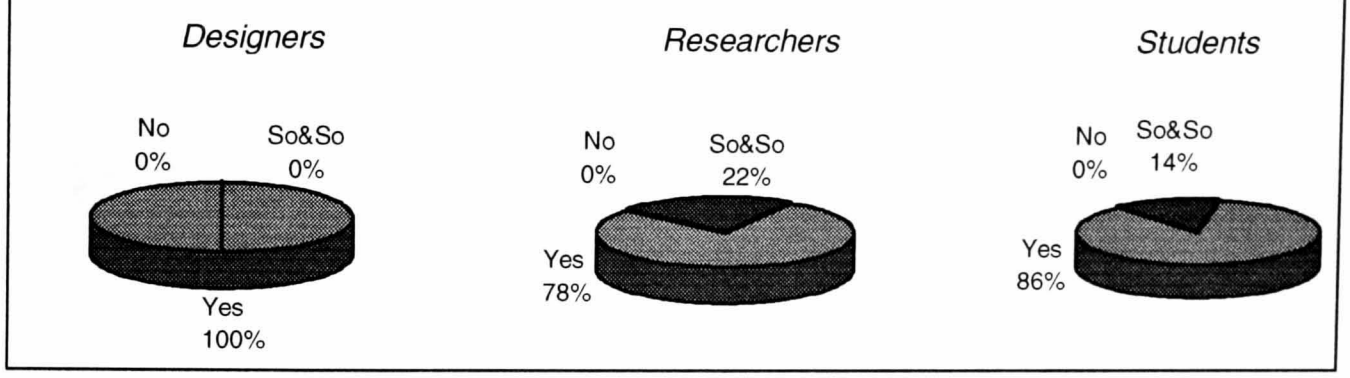
(Gen = Only generic, Comp. = Company Oriented, Sol. = Solution Oriented)



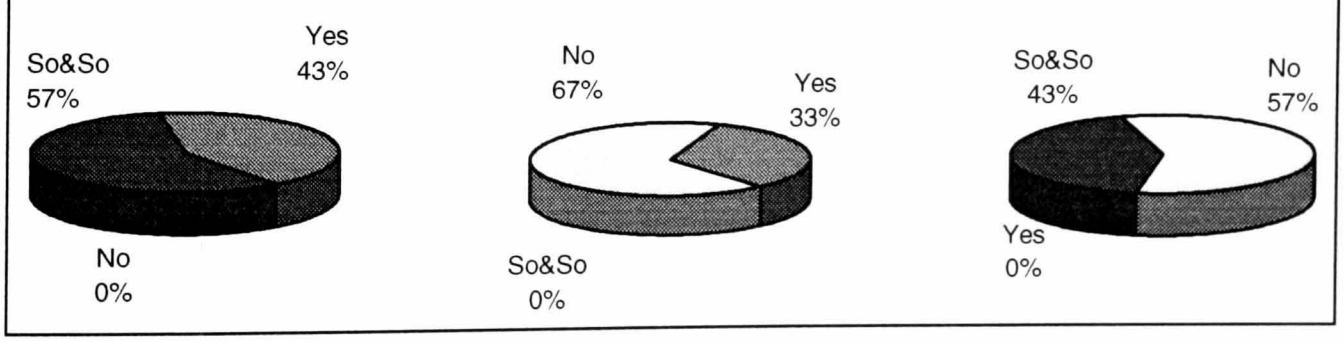
Q7. Do you consider the concept of 'concurrent synthesis' is demonstrated by the provided by the system?



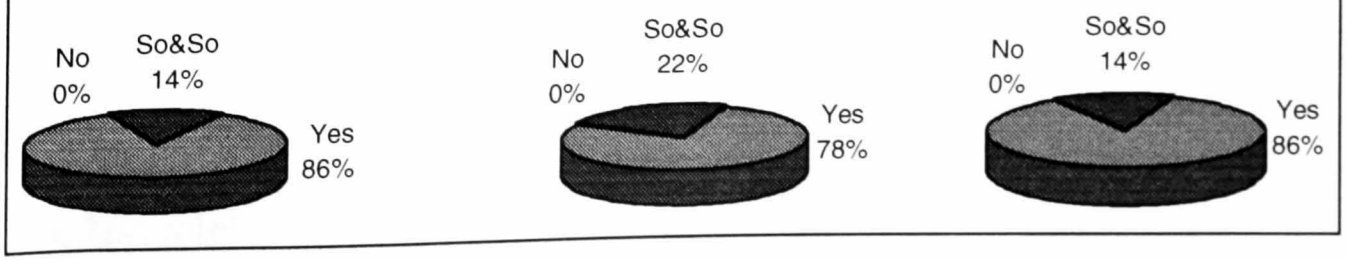
Q8a. Would awareness of LCC motivate you to 'explore' alternative commitments?



Q8b. Would awareness of LCC 'hinder' your design freedom?

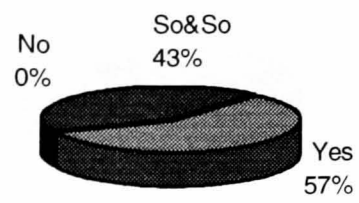


Q8c. Would awareness of LCC make you confirm commitments more consciously?

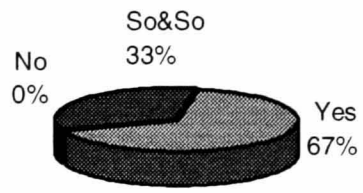


Q9. Do you find the use of Cost, Time & Quality performance measures as a suitable means for comparing the relative influence of a consequence on the different life-phases?

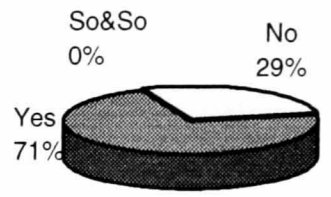
Designers



Researchers

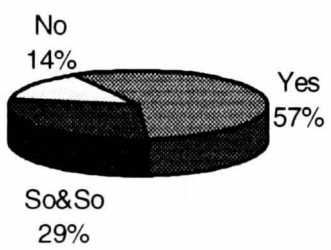


Students

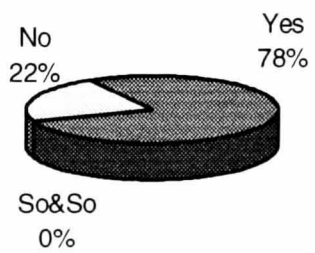


Q10a. Do you find consequences detected to be reported in an adequate way?

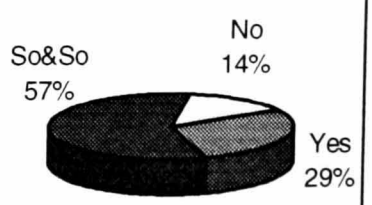
Designers



Researchers

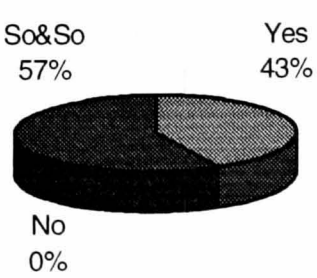


Students

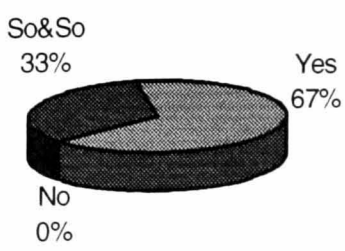


Q10b. Do you find consequences detected easily to 'locate' and 'browse' upon?

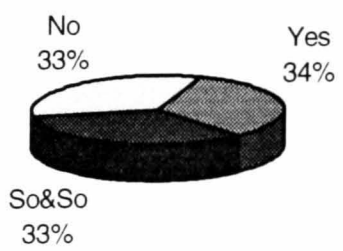
Designers



Researchers

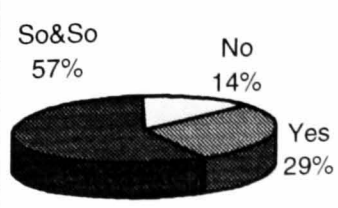


Students

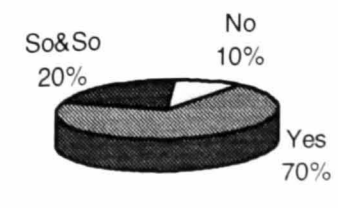


Q11. Do you consider the system provided advice to the avoidance, or relaxation of consequences?

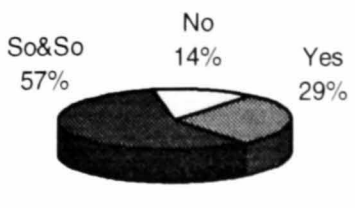
Designers



Researchers

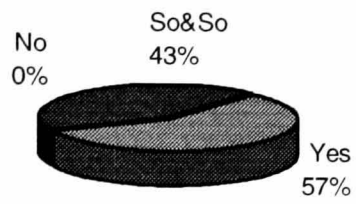


Students

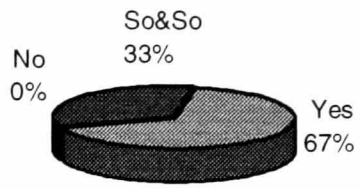


Q9. Do you find the use of Cost, Time & Quality performance measures as a suitable means for comparing the relative influence of a consequence on the different life-phases?

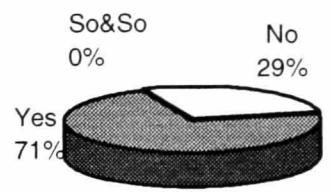
Designers



Researchers

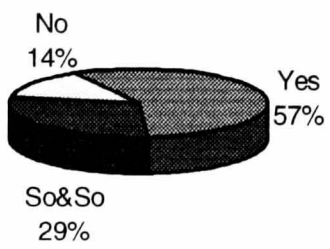


Students

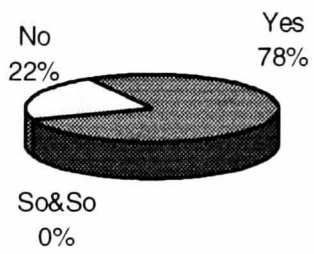


Q10a. Do you find consequences detected to be reported in an adequate way?

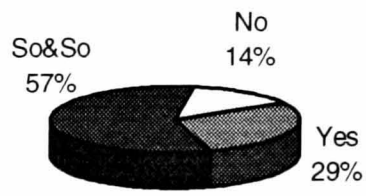
Designers



Researchers

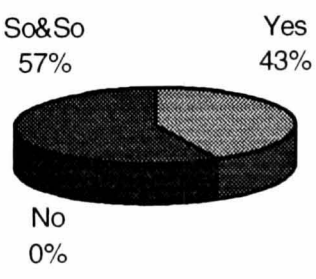


Students

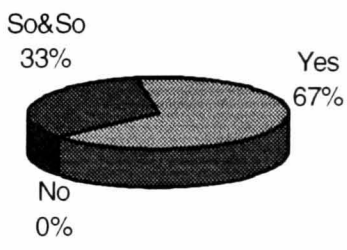


Q10b. Do you find consequences detected easily to 'locate' and 'browse' upon?

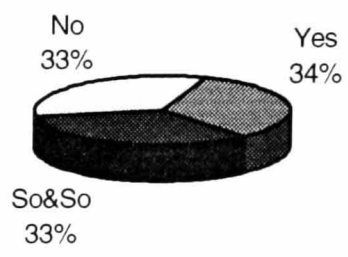
Designers



Researchers

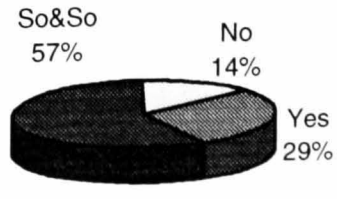


Students

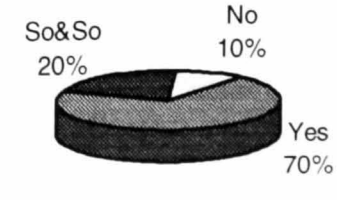


Q11. Do you consider the system provided advice to the avoidance, or relaxation of consequences?

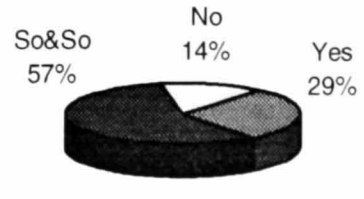
Designers



Researchers

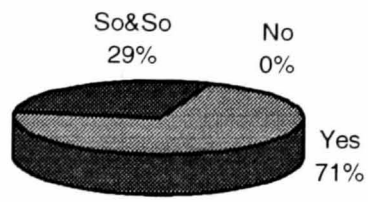


Students

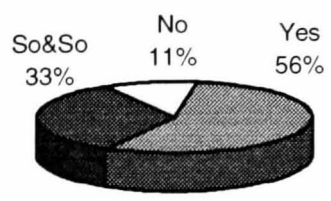


Q12. Do you think that keeping track of the fluctuations in performance measures for the different life phases, with different decision commitments is a useful way of formatting life design history?

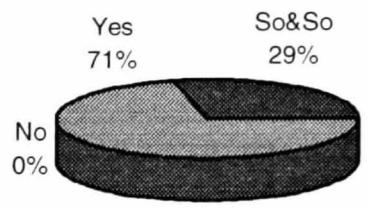
Designers



Researchers

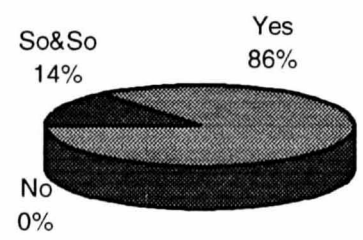


Students

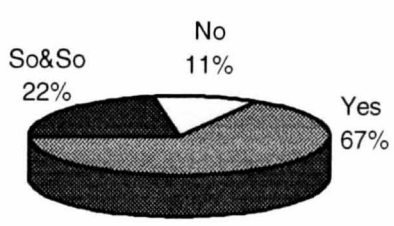


Q13. Does designing with FORESEE demonstrate the approach of how 'knowledge of consequences' can be employed during component design synthesis?

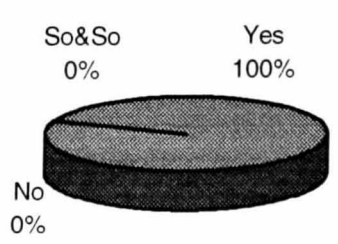
Designers



Researchers

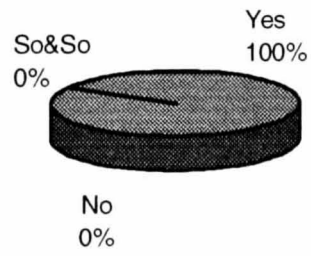


Students

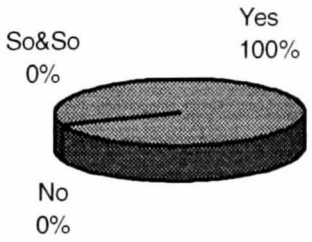


Q14. Does FORESEE demonstrate how new objects can be added and knowledge content manually changed during the use of the system?

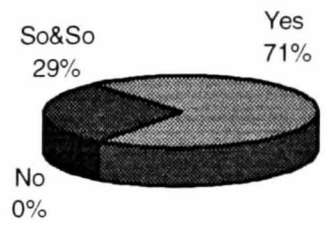
Designers



Researchers

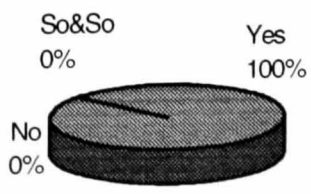


Students

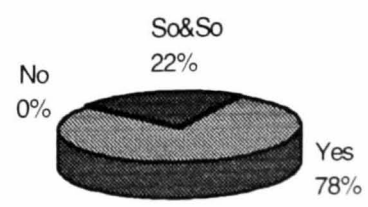


Q15. Assuming more knowledge was embedded within such a system, would you consider the proposed 'knowledge of consequences' approach to life design, to be useful in practice?

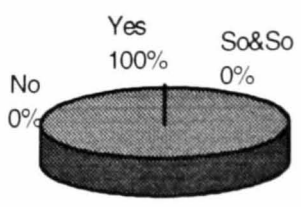
Designers



Researchers



Students



Appendix N – Ph.D. Research Publications

This appendix provides a list of research publications concerning original work carried out during this Ph.D.

	Year	***Publication***
1.	Sep. 1996	Borg J. and K.J., MacCallum. (1996). *"Structuring Knowledge of LCCs for Supporting Concurrent Design Exploration".* Proceedings of the 2nd International Knowledge Intensive CAD workshop organized by IFIP WG5.2 at Carnegie Mellon University, Pittsburgh, USA, 16-18 September 1996. Published by Chapmann & Hall, ISBN 0-412-81450-1, pg.208-223.
2.	Aug. 1997	Borg J. and MacCallum K.J. (1997). *"A Life-cycle consequences Model Approach To The Design For Multi-X of Components",* in Proceedings of the 11th International Conference on Engineering Design (ICED97), Vol. 2, edited by Riitahuhta A. Published by Tampere University of Technology. pg. 647-652.
3.	April 1998	Borg J.C. and X.T. Yan. *"Design Decision Consequences: Key to Support for 'Design for Multi-X' "*, Proceedings of the 2nd International Symposium on 'Tools & Methods for Concurrent Engineering', held in Manchester, UK, 21-23 April, 1998, pg. 169-184.
4.	Dec. 1998	Borg J.C, X.T. Yan and N.P.Juster. *"A KICAD Tool For Pro-Active Exploration Support To 'Design Synthesis for Multi-X' "*, Proceedings of the 3rd International Knowledge Intensive CAD workshop organized by IFIP WG5.2, held at Tokyo University, Tokyo, Japan, 1-4 December 1998, pg. 274-295.
5.	Aug. 1999	Borg J.C, X.T. Yan and N.P.Juster. *"Exploiting Decision Consequences to Support 'Design for Multi-X' – An Evaluation"*, accepted for publication in the Proceedings of the 12th International Conference on Engineering Design (ICED99), being held in Munich, Germany, August 1999.
6.	1999	Borg J.C, X.T. Yan and N.P.Juster. *"Guiding Component Form Design Using Decision Consequence Knowledge Support"*, accepted for publication in the Journal of Artificial Intelligence in Engineering Design and Manufacture, Vol.13 (5).