Design as interactions of problem framing and problem solving: a formal and empirical basis for problem framing in design

Thesis

Doctor of Philosophy Thesis

DESIGN AS INTERACTIONS OF PROBLEM FRAMING AND PROBLEM SOLVING

(A formal and empirical basis for problem framing in design)

Martin Džbor

Knowledge Media Institute
The Open University
United Kingdom

Supervised by:	*Dr. Zdeněk Zdráhal*
	Dr. John B. Domingue
Examined by:	*Prof. Bashar A. Nuseibeh* (The Open University)
	Dr. David Robertson (AIAI, University of Edinburgh)

February 1999 – June 2002

Thesis submitted in partial fulfilment of the requirements for a PhD in Artificial Intelligence

Abstract

In this thesis, I present, illustrate and empirically validate a novel approach to modelling and explaining design process. The main outcome of this work is the formal definition of the *problem framing*, and the formulation of a *recursive model of framing in design*. The model (code-named ***RFD***), represents a formalisation of a grey area in the science of design, and sees the design process as a recursive interaction of problem framing and problem solving.

The proposed approach is based upon a phenomenon introduced in cognitive science and known as (*reflective*) *solution talkback*. Previously, there were no formalisations of the knowledge interactions occurring within this complex reasoning operation. The recursive model is thus an attempt to express the existing knowledge in a formal and structured manner. In spite of rather abstract, knowledge level on which the model is defined, it is a firm step in the clarification of design process. The RFD model is applied to the knowledge-level description of the conducted experimental study that is annotated and analysed in the defined terminology. Eventually, several schemas implied by the model are identified, exemplified, and elaborated to reflect the empirical results.

The model features the mutual interaction of predicates '*specifie*s' and '*satisfie*s'. The first asserts that a certain set of explicit statements is sufficient for expressing *relevant* desired states the design is aiming to achieve. The validity of predicate '*specifie*s' might not be provable directly in any problem solving theory. A particular specification can be upheld or rejected only by drawing upon the validity of a complementary predicate '*satisfie*s' and the (un-)*acceptability* of the considered candidate solution (e.g. technological artefact, product). It is the role of the predicate '*satisfie*s' to find and derive such a candidate solution. The predicates '*specifies*' and '*satisfies*' are contextually bound and can be evaluated only within a particular conceptual frame. Thus, a solution to the design problem is sound and admissible with respect to an explicit commitment to a particular specification and design frame. The role of the predicate '*acceptable*' is to compare the admissible solutions and frames against the 'real' design problem. As if it answered the question: "*Is this solution really what I wanted/intended?*"

Furthermore, I propose a set of principled schemas on the conceptual (knowledge) level with an aim to make the interactive patterns of the design process explicit. These conceptual schemas are elicited from the rigorous experiments that utilised the structured and principled approach to recording the designer's conceptual reasoning steps and decisions. They include

- the refinement of an explicit problem specification *within* a conceptual frame;
- the refinement of an explicit problem specification *using a re-framed* reference; and
- the *conceptual re-framing* (i.e. the identification and articulation of new conceptual terms)

Since the conceptual schemas reflect the sequence of the 'typical' decisions the designer may make during the design process, there is no single, symbol-level method for the implementation of these conceptual patterns. Thus, when one decides to follow the abstract patterns and schemas, this abstract model alone can foster a principled design on the knowledge level. It must be acknowledged that for the purpose of computer-based support, these abstract schemas need to be turned into operational models and consequently suitable methods. However, such operational perspective was beyond the time and resource constraints placed on this research.

Acknowledgements

This thesis talks about design as an interactive activity, emphasising the interaction of the different knowledge sources and types of knowledge. The knowledge-intensive work underlying this research and the thesis is no exception in this matter, and the outcomes argued in this document emerged in a process of knowledge communication and knowledge sharing. Let me herewith acknowledge the assistance provided by numerous contributors over the course of my research. As things turn out, I believe I was lucky in becoming a member of KMi so early in its existence. It is usually a great challenge to join a multidisciplinary team of renowned researchers. This challenge is twice that big if one joins a team that is under such a heavy development as KMi was (and still is).

For giving me the chance to tackle these challenges, I have to thank to Zdeněk Zdráhal who recognised some research potential in my early and rather fuzzy ideas. This research was conducted together with Zdeněk in a formal role of a primary supervisor, and a number of 'informal' roles. These informal ones were especially valuable because they made the life in KMi more interesting than one would expect from an academic establishment. They ranged from the debates inspiring and reassuring to provoking and challenging. I am also grateful for Zdeněk's ability to generate a lot of extra-curricular work, so that I did not have a chance to neglect my other ICT skills.

Zdeněk together with John Domingue provided invaluable comments throughout the entire research but especially at its final stage when they thoroughly read and commented on the earlier drafts of this thesis. The ideas for evaluating and visualising the gathered data, including the infamous Napoleon's March to Russia, originate in John's extensive expertise with program visualisation. I am also grateful for his patience to read all papers I produced, and provide comments to improve my language skills. If these skills look better now than they did, it is thanks to John and to Naomi Williams, my EFL tutor.

A number of other people provided comments on the ideas presented in this thesis at the different stages of my research, including Marc Eisenstadt, Marek Hatala, Joanna Kwiat, Paul MacEachern, Paul Mulholland, Tammy Sumner, Victoria Uren, Michael Valasek, and Stuart Watt. I cannot forget my former colleagues at the University of Technology in Košice, Slovakia, and Tokyo Metropolitan Institute of Technology, Japan, with whom I have a fruitful relationship although I left them a while ago. Especially Tomáš Sabol, Ján Paralič, and Shuichi Fukuda. To them belong thanks in respect to my project management, software design, and teaching skills, and the introduction into the scholarship. And obviously, I am grateful to my examiners, Prof. Nuseibeh from The Open University and Dr. Robertson from the University of Edinburgh for their helpful and encouraging feedback during my Viva.

In addition to the people mentioned earlier who helped with my PhD, there were numerous other friends who helped indirectly. Since it would be too long to write all their names, I will use the context in which we met, and they may be able to identify themselves. Thus, thanks to all KMi and Open University research students for their active participation in the seminars, and to all colleagues from KMi for creating a nice working environment and supporting me in various ways. The colleagues and students from the D309 summer schools taught me to express my ideas clearly and in a lay person-friendly manner. The experimental work would not be possible with the enthusiasm of Petr Horacek and his team from the Czech Technical University, Prague. Also, I greatly benefited from the comments of the anonymous reviewers and not-so-anonymous participants in prestigious scientific conferences I was privileged to present my work at.

Finally but definitely not lastly, my greatest debt is to my parents and brother. Although I was not sure myself whether or not to pursue this research career at KMi, their vision and trust have always been an endless source of my strength and confidence. It might be true that one cannot choose their family, but in my case, I am proud to have such wonderful and understanding parents as well as brother. The dedication of this work to them is only a little acknowledgement of all that support.

Ďakujem!

[illegible]

Martin Džbor

TABLE OF CONTENTS

1 EXECUTIVE SUMMARY

This thesis presents a perspective on the interpretation and solution of non-trivial problems in engineering design. Engineering design represents a class of problems that are collectively referred to as 'ill-structured' (Simon 1973). Among other characteristics of this type of problem, which we want to emphasise one that appears as a thread throughout the thesis, namely, a 'design solution'. This includes the actual designed artefact or product, as well as the refined or extended specification of the problem to be solved. Our claim is that the explicit problem specification is not 'given', but evolves to respond to the development of the design solutions (products). This is an important feature of all ill-structured problems – the problem can be better understood (i.e. structured) only while developing and reflecting on its (partial) solutions.

Another important corollary of the view presented in this thesis is the fact that the explicit specification of any (non-trivial) design problem is inherently incomplete both at the beginning and throughout the design process. A set of statements about the desired states becomes the 'problem specification' only at the moment the designer accepts the proposed artefact or product as a fully-fledged design solution. Let us discuss the essential statements of the thesis in this executive summary.

1.1 Class of problems

As we already mentioned, we are interested in engineering design. To be precise, we understand the task of design in accordance with Bylander and Chandrasekaran (1985), Qian and Gero (1996), or Goel (1997). Accordingly, design can be defined using the following tenets; the essence of the original research of this thesis is expressed in points d) and e):

a) a design problem can be specified as a set of requirements and constraints that refer to *functions* of the designed artefact and/or *properties* of the design elements;
b) a design solution is developed in terms of *structural* elements and *relations* among them assuring the proposed structure meets certain desired functions and/or properties;
c) knowledge of a particular domain consists of simpler or more complex *mappings* from the set of structural elements to the functions and properties, through various relations;
d) mappings that form a domain theory depend on a particular *design perspective* (*frame*), i.e. a point of view from which the designer describes the conceptual world of designing;
e) we define a design perspective as a *circumscribing frame* imposed on the current design problem and created using familiar concepts from the design cases tackled in the past – this corresponds to the observation of Schön (1983) that designers shape the problems by seeing them as more familiar situations

1.2 Problem solving strategies in design

Design, like other ill-structured problems, does not have a single, generally applicable strategy or problem solving method that would be effective in all cases. Nevertheless, it is possible to highlight a few typical strategies that are repeatedly observable throughout the different design problems. So far, research in the field of engineering design has produced many different models of the design process. However, the vast majority of these attended to the design process on the level of explicit knowledge. Let us explain what we mean by the 'level of explicit knowledge', first.

The vital difference between the proposed approach and the existing ones can be summarised in a sentence that reads approximately as follows. *The problem is given – these are the required features, and the task is to find a solution satisfying the requirements.* We claim that the initial premise of the previous sentence is flawed; i.e. the problem will 'be given' in explicit terms only at the end when we know its solution (the designed artefact). Thus, any robust model of a design process needs to take into account the fact that design is about *problem framing* (formulating, articulating, or interpreting) that subsequently leads to the construction of a solution (artefact, product).

In this thesis, we present a model of the design process that in our opinion satisfies this strong condition. Our Recursive model of Framing in Design (*RFD*) sees design as an interplay or interaction of two knowledge-level reasoning actions that are expressed by a pair of conceptual predicates – '*specifies*' and '*satisfies*'. The first predicate attempts to define the explicit vocabulary that would describe the vague design problem sufficiently for the subsequent problem solving. This articulation is bound by an entity we define as '*conceptual design frame*'. We argue that such a frame corresponds to a certain view of the problem at hand; it is an interpretation of the vague and incompletely described problem. A frame serves as a kind of pool of opportunities, from which various artefacts addressing the particular specification may be constructed. The process of artefact or solution construction is not blind and random. The available concepts are chosen and put together so that the final structure satisfies the current explicit specification of the design problem that is interpreted in a particular design frame.

The earlier-mentioned interplay of the two predicates occurs when for one reason or another the artefact that satisfies the explicit specification of the problem is not accepted as a design solution. Another reason for the interaction is that the used interpretation (frame) does not specify the design problem to the full extent. Consequently, the problem may be re-interpreted or re-framed, and a new conceptual design frame may be articulated. In a modified frame, new conceptual building blocks are available, entirely new products may be articulated and constructed that would satisfy the amended design frame and would be acceptable. A simple sketch of such interplay is depicted in Figure 1–1.

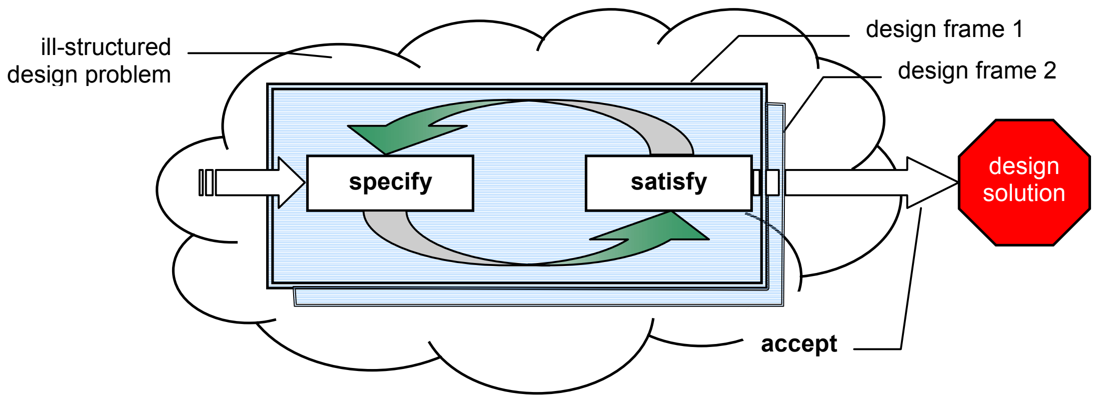


Figure 1–1. Interplay between problem specification and satisfaction within a particular design frame

1.2.1 Level of explicit knowledge

We associate most of the processes that can be modelled on the level of explicit design knowledge with the second predicate – box '*satisfies*' from Figure 1–1. We believe that this phase of the design process is sufficiently well researched, and therefore go through it only superficially. This is not the focus of the presented research. Therefore, we mention only one applicable model illustrating this predicate on the level of explicit design knowledge using the following basic reasoning strategies:

- *abduction from the domain theory* … the purpose of abduction is to assign a structure or a set of structural elements to the desired functions/properties (see assumption a) in section 1.1);
- *deduction in the domain theory* … the purpose of deduction is to ensure the speculatively abduced element (a set of elements) is indeed consistent with the remainder of the domain theory through generating additional consequences of choosing a particular structure;
- *consistency maintenance* … the purpose of consistency maintenance is to ensure the solution developed within a particular domain is a consistent and valid assignment/selection of structural elements

1.2.2 Level of (often) inarticulate processes

In addition to the explicit forms of reasoning that can be modelled using, for example, the earlier mentioned operations (section 1.2.1), there is a significant portion of reasoning in design that is not expressed in the same, formal language. Not only it is not expressed in such an explicit formalism, but also it is not even <u>expressible</u> explicitly! For instance, various subjective assumptions, designer's intentions, 'feelings' and expectations, may influence the design process – though typically, they are not (or cannot be) explicitly articulated to the full extent.

We believe that the <u>framing</u> of a design problem is often a typical representative of these 'inarticulate' or 'hard-to-articulate' reasoning processes. The ability to interpret a problem or to frame a vaguely defined design situation in the 'right' way is one of the most valued skills in the art of designing. Another process of great importance especially in creative and innovative design that resists an unambiguous explicit articulation, is a designer's <u>reflection</u> on his or her design decisions. As with framing, the ability to reflect in the 'right' direction distinguishes design masters from the novices (Schön 1983).

In this thesis, we focus mainly on this specific 'operation' of conceptual framing. We shall define the concept of 'design frame' as a knowledge-level operation that aims to circumscribe otherwise vaguely and incompletely defined design space. We will argue that the design frame cannot be given but must be constructed and re-constructed if an explicitly admissible solution is judged unacceptable. We define this re-construction as a recursive frame development or simply *re-framing*, and present several schemas featuring various forms of re-framing in design.

1.3 Organisation and structure of the thesis

In this section, we present the outline and the organisation of the subsequent chapters of this thesis. First, we identify five main parts of the thesis, and present them as a flowchart to show their mutual dependencies and causal order to simplify the reader's navigation through the document.

This thesis is divide into five major parts, each consisting of two or more chapters. The breakdown in Table 1–1 lists the five parts in their sequential order as a structured narrative. Figure 1–2 shows the individual chapters of the thesis grouped in their respective parts. Each part is depicted as a larger block with a colour background. The arrows in the figure show the chapters in a causal ordering. The arrow direction and orientation means that the chapter at the 'source' of the arrow directly informs the chapter at the 'target' of the arrow.

Table 1–1. Organisation of the thesis

Part	Chapter	Themes
Introduction	1. Executive summary	Setting up the stage, definition of foci, organisation of thesis, articulation and discussion of contributions, brief outline of the RFD model
	2. Outline of RFD model	
	3. Thesis roadmap	
Review of relevant literature	4. Perspectives on design	Review of literature on various perspectives for the investigation of design and literature on knowledge-intensive design models/tools
	5. Knowledge-intensive design	
Development of RFD theory	6. Theory of framing	Formal definition of main concepts of the theory of framing in design, basic predicates and their control flow, elaboration of the initial model based on the results of the experiments
	10. Extensions to RFD model	
Empirical validation of RFD theory	7. Background to experiments	Overview of empirical study, methodological notes, tools used in the experiments, annotation and RFD analysis of two design sessions, results aggregation, visualisation, evaluation
	8. Analysis of experiments	
	9. Results, implications	
Concluding remarks	11. Summary	Re-iteration of the essential contributions, concluding remarks, suggestions for further work
	12. Conclusions, further work	

Introductory chapters are shaded in gold; the concluding ones have a violet background. The body of the thesis consists of the three parts located in the centre of the flowchart. The review is shaded blue, the main contribution – the RFD model[1] of design is in orange, and finally, the chapters related to the empirical study have a green background.

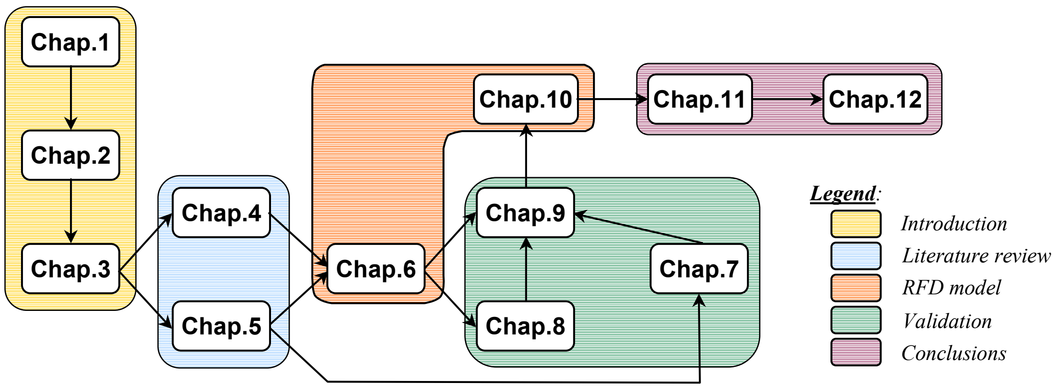


Figure 1–2. Interrelation of the individual chapters

Below, we show yet another organisational structure of the thesis. This time it takes into account the logical relations and the contextual associations of the individual chapters with a particular issue or set of issues. We state shortly what is the problem investigated in the following chapters, what are the assumptions and expectations, and goals of the thesis. Finally, we briefly summarise the validating empirical study, and mention how we evaluated the findings.

[1] RFD is an acronym frequently used in this thesis, and stands for ***R****ecursive model of* ***F****raming in* ***D****esign.

We would like to recommend a 'minimal route' for reading this thesis as follows. Assuming the reader reached this point, the next 'essential' chapter is number 3. Then we recommend reading at least chapter 4 that gives serves as a foundation for and gives a background to the remainder of the thesis, and concludes with the enumeration of the characteristics of design problems. Next, we recommend reading chapter 6 because it contains the essential definitions of the RFD model and the theory of framing in design. Chapter 7 follows from chapters 4 and 5; it introduces the experiment but is not vital for understanding the subsequent chapters. Chapter 8 is recommended in order to understand the experimental work. Finally, chapters 9 and 10 are essential, because they contain the results and the extension schemas to the RFD model.

Problem investigated in the thesis (see chapters 3, 4 and 5):

- design (engineering design) → conceptual phase

The goals of the thesis (see chapters 6 and 10):

- formal definition of the terminology (incl. terms such as 'design frame', 'problem framing', 'acceptability', and 're-framing');
- formulation of a knowledge-level model of the design process featuring the formally defined concepts;
- articulation of the schemas or patterns for (reflective) re-framing of design problems

Expectations/assumptions (see chapters 3 to 10):

- design is conducted on the explicit level with the occasional (but essential) input from the inarticulate (tacit and implicit) knowledge levels;
- in order to design, a problem and its solutions must be expressed in explicit terms that can be checked against the explicit requirements/constraints;
- the inarticulate level typically acts in the problem (re-)interpretation and reflection, and this input exhibits certain typical patterns;
- the result of re-interpretation and reflection is an amendment of the explicit specification(s) of the problem and/or its admissible solution

Validation and empirical study (see chapters 7 and 8):

- 24 design experiments recorded, annotated and analysed;
- detailed annotation and analysis of a design of an active suspension (task T11);
- detailed annotation and analysis of a design of a paper-smoothing plant (task T21)

Results and evaluation (see chapters 9 and 10):

- analyses of 24 design sessions aggregated and depicted visually;
- re-occurring patterns identified and exemplified, first on the examples from literature, then the analysed results of the empirical study

2 OUTLINE OF THE MODEL OF FRAMING

Before immersing into the depths of the subject of this thesis, let us draw a quick sketch in order to give a picture of what a reader may expect to find in the subsequent text. This short chapter can be seen as a kind of scenario or a screenplay for doing design as we envisage it in the proposed model. In section 2.1, we look at what decisions are typically made during the process of designing. The claims and observations made in section 2.1 are expanded, refined, and justified in chapters 4 to 6, later in the document. Section 2.2 highlights the processes forming the subject of the enquiry in the documented research. Again, a superficial sketch serves as an introduction to a complex research field, and further chapters work on the sketched subject more in depth.

2.1 Foreseen scenarios of design(-ing)

Many existing models of design assume that the designer is given an explicit specification of the problem to which a solution is sought. According to Chandrasekaran (1990), such specification typically contains a list of desired functions, properties and various constraints on the product or manufacturing process. Furthermore, these models usually assume that the designer has also a certain repository of components, modules and structures at his disposal. It is then the designer's task to search and choose such a structure that would meet the desired functionality.

There were developed numerous methods and techniques for this search for a design solution. They were constructed using different paradigms, but in the end, they all aimed at satisfying the given explicit specification. For the purpose of illustrating our point, we chose two such methods that are generally applicable in design – abduction from a logical theory, and logical deduction. More details on these explicit methods are mentioned elsewhere in this document, see section 6.4. Thus, the designer may abduce a sufficient structure that according to his or her domain theory has a potential to deliver the desired functionality.

However, reasoning by abduction is not logically sound; the abduced structure can be used only if it and its consequences are consistent with the logical theory and all the requirements and constraints. It is clear that the abduced structure must be evaluated against various explicitly formulated constraints, e.g. in respect to the compatibility, safety or manufacturing. Typically, these constraints are related to the solved problem, and by referring to them the designer seeks an answer for a question: "Are the given requirements and constraints met by the proposed structure (product, partial solution)?"

In addition to the evaluation of compliance with constraints as described in the previous paragraph, the designer must ensure that nothing inconsistent can be derived from the abduced structures. Therefore, it may be necessary to deduce the inevitable consequences of believing a particular set of the abduced statements. Unlike abduction, reasoning by deduction infers only the *necessary* consequences that are implied by certain believed axioms, decisions, etc. These derivations typically produce statements in form of logical theorems. Nevertheless, both operations mentioned so far seek a solution that is plausible for the purpose of the given design problem – a design solution or product that *satisfies* the explicit specification of the problem.

Now, there may be several situations happening as a result of the abduction, deduction, and consistency maintenance. The first and most desirable one is that at least one consistent solution exists in the given domain theory. According to the explicit models of design, this is the desired state, and the design process may be stopped. We leave this case aside for a moment. Second, a less desirable, albeit very frequent, situation occurs if no suitable, i.e. consistent structure can be abduced from the given domain theory. In other words, there is an explicit conflict or a contradiction between two or more constraints interpreted in the particular domain theory. Any such contradiction needs to be removed and rectified in order to develop a consistent solution, and to that extent, we may assume the following two paths for the conflict resolution:

a) current abduction can be discarded, the reasoning method backtracks to the last consistent state, and presents *another consistent abduction*;
b) current abduction is again discarded but the reasoning method *cannot present any alternative* suggestion in the current domain theory

The situation described in point a) is rather simplistic, and can be achieved through various existing, algorithmic or heuristic techniques for *backtracking* (Dechter 1992; Sabin and Freuder 1994; Black 1999; Chen and van Beek 2001). An alternative approach to the resolution of an explicit conflict is available in numerous works on truth maintenance – e.g. contextual extension of a common logic (de Kleer 1986a). The second situation, described as b) is very common in the real-world design, and requires a significantly different approach. We believe that one of the potentially applicable techniques is, for instance, TRIZ – an algorithm proposed by Altshuller (1984). This algorithm tries to identify the minimal contradicting set (usually a pair) of requirements, and resolve it by an application of less usual analogy.

TRIZ argues that the designer has to perceive the problem from a different, perhaps novel and innovative perspective in order to overcome the inconsistency that hampers his or her efforts. In other words, this resolution does not lie inside the problem solving theory. By referring to the analogous problem, the designer brings in new concepts, axioms and assumptions into the logical theory. Eventually, the inconsistency and conflict between the requirements are resolved not by backtracking but rather by re-interpreting the conceptual perspective. Unlike monotonic backtracking, such re-interpretation or perspective shift is already a non-monotonic operation. This is an important finding because it bridges the gap between the theoretical ‘perspective shift’ argued by Schön (1983) and a practical ‘inventive algorithm’ of Altshuller.

Thus, looking at a design problem from a different angle, a designer may appreciate new features, new conceptual objects s/he was unaware of before. As Schön puts it, a designer learns more about the design problem by *shaping* (framing) and *re-shaping* (re-framing) it. Nonetheless, a shift of perspective or an operation of re-framing triggered by an explicit contradiction is only one of the challenges that is not entirely accounted for by the many existing design models. Even bigger issues arise from the fact that it is not always possible to identify and articulate the conflict explicitly. Let us return to the ‘simple’ situation from point a) above, where there was at least one consistent solution existing in the domain theory. Suppose the abduced structures are indeed logically consistent with the current domain theory, and there are no logical contradictions or inconsistencies present. However, the designer *reflects on* ('looks at and analyses') the achieved state of the design, and declares that 'he is not happy with such a solution' or 'he does not like it'. In other words, a designer declares the achieved state being 'unacceptable' as a design solution. Is such a situation possible?

We believe that such a situation is not only possible, but it is very typical for the class of problems described as *ill-structured* (see also section 1.1). The incompleteness and ambiguity of the ill-structured problems literally calls for an extension of the problem solving theories. We argue that the designer's 'tacit dissatisfaction' with the current solution is caused by a conflict between his or her *expectations* and the explicit problem solution and/or specification. The declaration of unacceptability reflects the inarticulate need for a perspective shift. Similarly as it happened with the explicit contradictions (see paragraphs above), the shift may lead to the extension of the logical theory, which by definition introduces non-monotonic reasoning to the problem solving theory. This is a very tangled situation, difficult to describe – there are no explicit contradictions in the original domain theory but in spite of this, there is an inarticulate perception of insufficiency.

We suggest seeing this 'inarticulate lack of consistency' as an issue of selecting an incorrect or incomplete conceptual perspective (*frame*). The externalised (explicit) design frame that was used for the derivation of the 'unacceptable' solution is not synchronised with all the implicit or tacit expectations the designer may have. In order to resolve this 'inconsistency', a synchronisation between the explicit and tacit is desired. Intuitively, a remedy for the tacit conflict may lie in the perspective shift or *re-framing* of the design problem. However, unlike the situation with an explicit conflict, in this case, a 'tacitly perceived' inadequacy must be formulated explicitly before any changes to the perspective may occur.

The whole exercise of frame shifting aims to improve the designer's understanding of the 'given' problem, and to refine its interpretation. In our opinion, the frame shift occurs by an articulation of possibly new or novel concepts from which a different explicit problem specification and/or a solution may be constructed. We justify our opinion by belief that the explicit formulation of new design assumptions, requirements or constraints based on the tacit expectations, is actually an opportunity for the *exploration* of the ill-structured design space (Smithers, Conkie *et al.* 1990). Exploring the opportunities of the 'given' design problem by framing and re-framing, the designer refines his or her knowledge of the domain, the problem, as well as the criteria for declaring certain product a design solution. This refinement is an essential feature of non-trivial design problems, and eventually fosters a *co-evolution* of a problem specification together with a design solution (Nidamarthi, Chakrabarti *et al.* 1997).

The two techniques for conflict resolution are entirely different in spite of looking very similar. The former technique that deals with the explicit contradictions can be translated into solving a task of '*finding an alternative solution*' (point a) above) or '*solving an alternatively interpreted (framed) problem*' (point b) above). In other words, using a not-too-familiar but different analogy may help interpret the contradictory statements and/or the axioms of the problem solving theory differently. In the latter case, what the designer tackles is a problem of '*what is an alternative interpretation of the current design state*'. Instead of resolving the conflicts, s/he is trying to learn more about the problem, discover the opportunities of the current task, and find what can be done with the existing case.

We believe that the two stages of addressing design problems mentioned earlier are the foundation of the complexity of design. The points raised in the discussion make design a worthwhile research topic. On one hand there is a quest for a solution (structure) *satisfying* the problem specification, and on the opposite, a quest for an appropriate conceptual frame, in which a problem can be explicitly *specified*, and interpreted (and hopefully, also satisfied). In this thesis, we justify this belief by elaborating a formal reasoning framework we refer to as a '*Recursive model of Framing in Design*'.

2.2 Models of reasoning techniques in design

In the previous section, we mentioned several reasoning techniques applicable to design, such as for instance, frame articulation and modification, conflict identification, and design re-formulation. We stated also that these operations use to a large extent knowledge sources that are often complementary to the explicit and formal domain theories. These operations may go 'beyond' the classic logic and account for the earlier-mentioned 'inarticulate' or 'hard-to-articulate', non-monotonic reasoning processes. Listed in Table 2–1 below, is a brief overview of our proposals for modelling different reasoning operations that are typically observed in design.

The reader may note that the table summarises both explicit and tacit 'operations' from the previous section. These knowledge-level reasoning steps are listed in the order they appeared in the discussion conducted in section 2.1. Roughly speaking, our recursive model of framing in design (RFD) will address mainly those reasoning steps that have a grey background in Table 2–1. These steps and concepts are defined formally in chapter 6, and further extended in chapter 10. Nonetheless, we will briefly mention also the explicit operations but only as an illustration of the interrelations between our model and the existing ones (see chapter 6).

Table 2–1. Operations on different knowledge sources in design.

Operation	Formality	Knowledge manipulated	Description
framing	mixed	requirements, conceptual frame (ontology)	Interpretation of given requirement in terms of familiar situations, selection of a suitable design perspective, frame
abduction	formal	functional requirements, domain theory (mappings), structural elements	Production of candidate solutions in terms of structural elements from given requirements and domain theory
deduction	formal	domain theory (axioms, mappings), candidate solutions, functions	Derivation of necessary consequences in terms of functions/properties from assumed candidate solutions and current domain theory
consistency check	formal	domain theory (axioms, mappings), candidate solutions, constraints	Assurance that domain theory does not contain contradictions, esp. introduced by the abduced candidate solutions
conflict identification	formal (usually TMS)	domain theory (axioms, assumptions), constraints	Trace a conflict to its source – violated axiom, domain constraint, assumption?
refinement of design specification	mixed	conceptual frame, requirements, constraints	Explicit formulation of a commitment to a particular requirement/constraint that was implicitly present but not 'activated'
frame modification	inarticulate, informal	requirements, constraints, assumptions, candidate solutions, conflict, previous design cases	Search for another, relevant conceptual frame, and a formulation of analogous matches between the current design and suggested new frame
design re-formulation	mixed	conceptual frame(s), requirements, constraints, assumptions	Transfer of relevant knowledge from a modified frame in terms of additional requirements/constraints/assumptions

We propose indexing the operations in Table 2–1 using a reference to the level of explicitness of the knowledge sources that are used for a particular operation. We believe that a type of knowledge source also informs the character of the operation. According to the expectations, this character may range from formal to vague, and perhaps tacit and inarticulate. In order to avoid a narrow perspective we define different reasoning operations by emphasising what knowledge roles are manipulated rather than formulating some prescriptive methods. As we shall see later, these 'functional' definitions reveal several interesting similarities between the different operations.

When looking at those knowledge-level operations that contain other than formal means of reasoning, we may observe two important features. The first is a repeated reference to the previous knowledge (e.g. knowledge of past designs, knowledge transfer, etc.). The second striking feature is a presence of design perspective or context, which we call '*conceptual frame*' among the other knowledge sources. Though these two features appear to be mutually independent, they are closely related in the design reasoning. Perception of similarity, familiar interpretation of a design problem, or transfer of previously acquired knowledge are a few examples of reasoning based on the designer's experience. We believe that analogy plays a far more important role in the design than the usual and well researched component re-use (Watson and Perera 1997). Its influence can be found throughout many different operations, and is particularly well observable in the conceptual phase of the design process and framing of the design problem in the familiar conceptual terms.

The important role of a 'conceptual design frame' was already hinted in section 2.1 earlier in this chapter. This is an essential concept of the proposed recursive model of framing in design(-ing), and we devote more space to it in chapters 6 and 10. The articulation of a conceptual design frame is often possible thanks to the designer's rich experience and knowledge of past design cases. We would even argue that the process of 'understanding a design problem' (i.e. *framing it*) features what Simon (1973) calls 'extra effort designers spend before being able to solve the problems'. This process would not be possible without such an experiential knowledge and reasoning based on the similarity and familiarity.

In the literature, different terminology is used to refer to that 'extra effort'. For instance, Coyne, Rosenman *et al.* (1990) talk about '*problem interpretation*', Dominowski and Dallob (1995), and Candy and Edmonds (1996) use term '*immersion*' and 'act of insight by which certain design elements are chosen'. Nakakoji, Sumner *et al.* (1994) present this issue as an '*explication of tacit intentions*', and Schön (1983) refers to this operation as '*situation shaping*'. Among so many different terms, Schön emphasises that 'conceptual shaping' (or framing) is a fully-fledged operation that serves as a 'pre-processor' for the rest of the design. A pre-processor that, in general, influences both, the understanding of what is the problem the designer tackles, as well as deciding how to solve the problem (i.e. how to construct a solution to it).

A similar notion appears also in the works of Jackson (2001), whose '*problem frames*' represent classes of familiar sub-problems that can be decomposed and analysed in a particular way. Jackson argues that the problem frames may play the same role in developing an understanding of a design problem and its explicit specification, as design patterns or design prototypes (Gero 1996) play in the solution development. In other words, a conceptual frame affects the entire design process, and the construction of such a frame deserves as much attention as design prototypes and other techniques.

In this thesis, we argue that the articulation of a conceptual design frame is an essential operation that informs any subsequent interpretations of the design as well as selections and applications of problem solving methods. We present a recursive model of framing and re-framing that provides a holistic picture, and attempts to see different ‘design problems’ as specialised instances of one complex model. Among the ‘instances’ covered by the proposed model are a routine design with constraints and fixes (Yost 1992), and the inventive design with contradictory requirements (Altshuller 1984; Sushkov, Mars *et al.* 1995). The proposed model also contains schemas and patterns applicable to innovative designs with evolving specifications (Gero 1996; Goel 1997), and conceptual refinements of the design solutions (Nidamarthi, Chakrabarti *et al.* 1997). We argue that these specific instances occur in accordance with a few predictable and describable patterns. This awareness of the different patterns is important and beneficial for the purposes of knowledgeable (or knowledge-intensive) and adaptable design support. We believe that further research into the occurrence of different reasoning schemas (patterns) may indeed leverage a more realistic and more natural interaction between a designer and a computational support tool.

The author of this text is fully aware that the research described in the proposed thesis only scratches the tip of a massive iceberg. However, it is our firm belief that the proposals made can be used and further enhanced in a future research thus benefiting the research into design theory and support.

3 THESIS ROADMAP

The research addressed in this thesis is concerned with selected features of engineering design problems. Many books and research articles were written about design and designing that investigated these topics from several different perspectives. Design as a problem solving activity has been practised for many centuries; however, as a scientific discipline it was acknowledged relatively recently. Unlike other, mature sciences (e.g. mathematics or physics), the science of design is still relatively young, and it is still developing its own scientific apparatus, methods of enquiry and terminology. One of the important facets that need to be addressed by any science is to understand and formally ground its subject of enquiry – in our case 'design'. It is the intention of the presented thesis to contribute to defining and shaping the science of design on multiple levels.

First, a recursive model of framing in design is proposed as a feasible reasoning strategy. Its main purpose is to contribute to the understanding of our subject of enquiry. We shall investigate the design process, and devote our effort to the description of different reasoning strategies that may underlie such a complex activity as design. Second, we investigate the deployment of the proposed model and the theory of framing for the representation of design problems. The ultimate goal is to propose a seed of a methodology for supporting the designers who are tackling non-trivial design problems. It is the objective of this chapter to set up the stage for the rest of the thesis, and present motivation, background, and the main contributions of the conducted research.

We begin with a few illustrative examples from the real-world situations that serve as a motivation for the research work (see section 3.1). We believe that these examples show what part of the design process and the designer's reasoning is in the focus in the subsequent parts of the thesis. Before going into details we roughly draw the boundaries for the positioning of this research. The subsequent text uses several terms that are common in our everyday speech, and may have certain charge. The terminology includes such commonly used terms as for instance, '*design*', '*knowledge*', or '*modelling*'. In order to avoid any additional ambiguity, section 3.2 in this chapter clarifies the terms that are used later for the construction of our proposals.

3.1 Motivation

One of the godfathers standing at the birth of the 'science of design', Herbert Simon (1973) includes design among so-called '*ill-structured problems*'. He argues that most of the real-life problems belong to this category, which is in fact an essential reason why something like 'science of design' is needed. Simon and other researchers name several typical features that characterise the ill-structured problems we encounter in the real world. Among others, it is the incompleteness and vagueness of the initial specification of the problem that very often contributes to seeing design as an artistic exercise. In addition to the incomplete beginnings, we can mention the absence of a well-structured problem solving space, as well as the absence of clear criteria for the determination of solution suitability. First, we shall look at these selected features of design, and relate them to the existing research and a real-world scenario. Throughout the motivating section, we concentrate on three issues that are interesting from the knowledge-level point of view: incompleteness of knowledge, applicability of knowledge, and knowledge dynamics.

Despite the fact that the above-mentioned statements are widely accepted as correct, many existing approaches to the design theory take into account the 'ill structure' only superficially. A lot of research has been concerned with the application of various formal reasoning techniques to various real-life problems. Unfortunately, only very few techniques and reasoning methods can be successfully transplanted from a well-structured mathematical world to an ill-structured reality. Most objections that emerge in this context have something to do with the necessity of making decisions with incomplete information at hand. With real-world problems, there are many exceptions to the well-defined rules; the real world has plenty of 'typical' and 'default' but also 'exceptional' features, which are able to overturn the logic. Further objections include the human ability to learn from their previous experience, create new objects and concepts in contrast to processing and re-using the existing ones, etc.

Nevertheless, the 'defaults' do not pose the most difficult obstacle for understanding design. Whether a statement is 'typical' or 'exceptional' it can be explicitly formulated and expressed using a suitable language. A statement may be forgotten in the initial problem specification, and may need to be added there later. However, in addition to the facts that are untold, designers take advantage of knowledge that is not only untold but in general cannot be expressed in the same (formal) language as the explicit design knowledge. The following examples from the different real-world situations show what we mean by the statement '*knowledge that may be used but resists formalisation*'. The examples are drawn from different domains to illustrate the commonality of 'non-explicit knowledge'.

3.1.1 A few examples from practice

The first two examples described in section 3.1.1.1 – cycling and shawl weaving, do not require any specialised knowledge, but still are able to describe the issues that are familiar to many of us. The next example in section 3.1.1.2 is taken from the domain of medicine, and illustrates the presence of some knowledge sources that complement the explicit formal techniques developed for the construction of a decision about a particular illness.

3.1.1.1 Can you tell how you do it?

Michael Polanyi was one of the first pioneers to draw the attention of the research community to the term 'tacit knowledge'[2]. He defined this 'hidden knowledge' as something that is inherently present and used when tackling a problem, though it may not be possible to express or explain it explicitly. The original example from Polanyi's book illustrating tacit knowledge describes cyclists who are able to cycle on any bike they are given. Many different explicit models can be developed in physics in order to address the dynamics of cycling as a motion. These models may include various formal definitions of different forces acting on the cyclist, the bicycle, the road, and so on. Nevertheless, any such explicit physical models will be of little use to the beginners who want to learn cycling.

[2] Term 'tacit knowledge' was introduced by Michael Polanyi in his 1966 book 'The Tacit Dimension'; we have borrowed the example of cycling from (Cook and Brown 1999).

Our ability to cycle includes among other issues also the ability to stay upright on a bike, keep our balance, and not fall off the bike. The vast majority of people when asked to explain the principle of balancing to a beginner, will be *able to do* it and show it. However, usually they will claim that they are *unable to say* how exactly they turn the handlebars, how they measure the forces acting on them, what forces are relevant for them, or what strength they apply. Simply, they can keep their balance and avoid falling down without having to know the physics and various explicit mechanical models. Although they are not able to describe what they are doing, neither when nor how they react, they clearly must have this knowledge because it is a prerequisite for riding a bike.

Polanyi claims that what people are talking about, is an *explicit dimension* of their knowledge. People may attempt to deliver some explicit explanation based on their knowledge of mechanics or physics, but this explicit statement will be insufficient for a novice cyclist to learn cycling and not to fall. The above-mentioned inexpressible knowledge, according to Polanyi and others, belongs to the *tacit dimension*. Cook and Brown (1999) emphasise that explicit and tacit knowledge exist in the different dimensions of the same problem space. They are also very distinctive forms of knowledge and neither is a variant of the other one. They complement each other, and typically, one type can be used to acquire the other one but never transformed to the other type (Cook and Brown 1999)!

Back to the cycling example, tacit knowledge of balancing is required to allow one to ride a bicycle. Whatever explicit description of what is happening when one rides a bicycle, may serve as *an aid* in acquiring the necessary tacit knowledge of staying upright. On the other hand, however precise the explicit description of 'cycling theory' is, it does not guarantee that a novice who masters it will actually be able to keep balance and ride a bike without falling off. It is not a problem of the amount of explicit knowledge one has about a particular task, or its quality. As Cook and Brown claim, the explicit form by itself simply cannot do all the necessary work; 'other dimensions' are clearly playing their roles in the game. The authors mentioned above understand *knowledge* as an explicit category that can be 'possessed', and tend to distinguish it from tacit *knowing*; i.e. knowledge that can be practised, knowledge as a competence (Newell 1982).

Design is in its nature far more similar to the example with cycling than it seems at the first glance. Similarly as cyclists use their tacit knowledge of staying upright to maintain their balance, the designers often have 'tacit expectations' when assessing the suitability of designed artefact. Similarly, they may use various implicit or inarticulate assumptions in interpreting the design problem or considering a particular decision in the solution construction. It is obvious that the designed artefact has to be suitable in respect to the given explicit requirements and constraints, such as colour, size, compatibility, etc. However, designers often go beyond the explicit assessment of the considered artefacts in order to develop a better impression of their work. A nice example of such a deeper form of assessment can be found in (Schön 1983). Schön describes a case of Slovakian peasants producing special and unique shawls woven of yarns dipped into home-made dyes. These dyes have been prepared for centuries using knowledge that was passed from one generation to the next. The inherited knowledge included both the skill (of practising the handcraft) and knowledge (of additives). However, when the weavers were introduced to the modern aniline dyes, which were supposed to simplify their work, the beauty of the shawls was '*spoiled and lost*'.

This happened not because the aniline shades were poorer or completely unsuitable for the shawl weaving business. On the contrary – they were richer and cheaper, and could replace the ‘natural’ ones. Unfortunately, such a straightforward logic completely failed in this particular scenario. Schön suggests another reason why this failure may have occurred. Namely, the actual design process was crudely disrupted by the ready-made artificial dyes, and the weavers were not able to recognise good or bad products because they lost some information that was vital for them. Something that they knew from previous generations, and commonly used when preparing hand-made dyes, was left out by the explicit problem of choosing and applying the artificial dyes.

In our context, we would argue that the weavers’ rich understanding of a complex procedure of dye preparation was abruptly replaced by explicit knowledge that had a far narrower scope of validity. The explicit knowledge might have been correct with regard to the colours and dye shades but apparently that was only a part of the information the weavers needed to assess the suitability of the prepared dye and the final product. They clearly evaluated all intermediate products ‘tacitly’ during the entire weaving process. Such an evaluation may have included among other aspects also the dye shade, perhaps time or temperature of its preparation, and perhaps other aspects that are difficult to describe by explicit, physical quantities. Since the practitioners’ knowledge from the ‘tacit dimension’ (Cook and Brown 1999) cannot be readily externalised, the account of a design process is typically vastly oversimplified by the sole focus on the explicit facets!

3.1.1.2 Rules are not enough

The next example is taken from slightly different domain of enquiry – formulation of a diagnosis in medicine. We describe diagnosing one particular disorder called Hashimoto’s lymphocytic thyroiditis (HLT) from sonographic images of the patient’s gland (Smutek, Tjahjadi *et al.* 2001). Just to give some basic background, we begin by stating a few facts about this disorder and techniques for its diagnosis. HLT is a diffusive inflammation of the thyroid tissue that eventually results in insufficient production of thyroid hormones. Since thyroid hormones regulate the body metabolism and have impact almost on all other organs, their low production may adversely affect patient’s health.

The diagnosis of this potentially dangerous disorder is usually based on the visual assessment of the images of thyroid area acquired by a sonograph (sound wave echoing). Medical students when learning to diagnose this complaint are told that HLT is manifested by a texture character captured on the sonographic image. They are also directed so as to focus especially on the ‘echogenicity of the tissue’ (ability to reflect the waves) and ‘textural structure of the parenchyma’ (including its regularity, roughness, etc.) The guidelines are rather simple and unambiguous; unfortunately, the reality is different. The scans of the thyroid area of the patients with a positive HLT exhibit a large variability and ambiguity. For example, there is not a single, uniquely and explicitly definable region in the picture to look at. The structural parameters to be assessed are very vague and do not have crisp values that would clearly prove or reject suspected HLT.

Just to illustrate what we mean by a vaguely defined problem, four snapshots are presented in Figure 3–1 (Sara, Svec *et al.* 2001). The dotted line is added to each image to simplify the identification of the ‘significant region’ that needs to be investigated. We can clearly see irregularity of the shape and size of the same region for different patients. In a real-world scenario, the shape and position of this region would have to be established by a medical practitioner from an original sonographic snapshot (i.e. without any graphical enhancements). Nevertheless, let us focus on the explicit advice from the books: '*Look at echogenicity and textural structure of the parenchyma in a significant region.*' Unless we have the necessary knowledge of what to look at, we can hardly use the explicit guideline to obtain a meaningful diagnosis. Clearly, the explicit account of the diagnosing process lacks an important piece of the entire puzzle – a chunk of knowledge, which is hardly formalisable. There is surely a thyroid context-dependent 'feeling' the medics develop during their numerous exposures to the similar tasks that enables them to recognise highly variable forms of HLT.

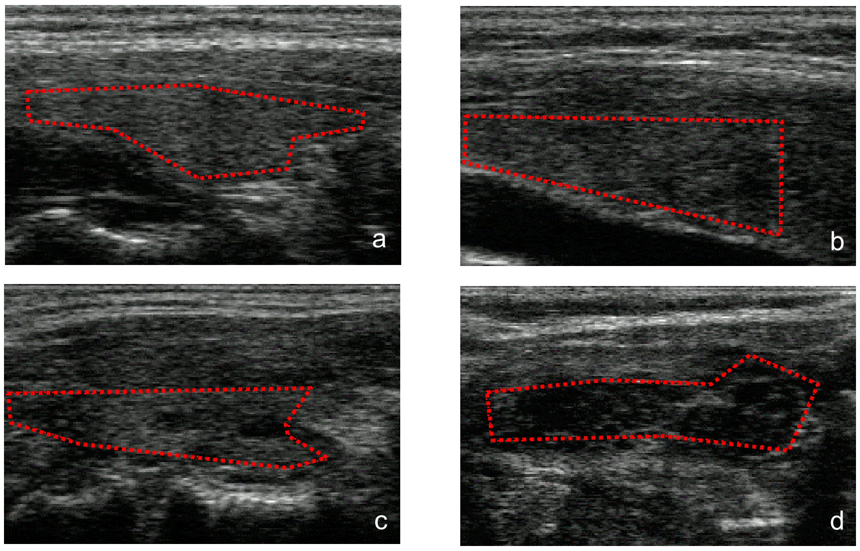


Figure 3–1. Images of healthy thyroid (a) and HLT-positive (b, c, d)

The visual assessment of the sonographic scans begins with the identification of a region of interest. This is depicted in the figures as an irregular shape with a dotted border. Once the significant region is determined, the properties of the texture are looked at. The practitioner needs to take into account the region itself, then texture granularity, smoothness, randomness, and many other features the parenchyma texture may exhibit. An answer to the question what is a 'significantly rough' or 'irregularly delineated' texture, is not simple. The practitioners usually find it very difficult to express this kind of knowledge but they admit that such knowledge is vital for the correct diagnosis.

Rather than having learned this knowledge in the explicit terms, the practitioners often talk about developing a feel for the situation or acquiring some 'intangible' but very practical ability of *exercising their knowledge* through their past experience. And it is this complementary knowledge and its relationship to the explicit one that is discussed more in-depth in the further parts of this document.

3.2 Context of the research

At the beginning of the thesis and chapter 3, we stated that the subject of our enquiry is design (in a broader sense) and engineering design (in a specific sense). Our intention is to investigate this subject as a complex reasoning task. We are particularly concerned with modelling the reasoning processes of a knowledgeable agent involved in designing non-trivial artefacts. This intention is thus defining the level, in which the subject will be investigated, and paradigm, in which the findings will be formulated.

The research discussed in the subsequent chapters weaves around three main themes addressing them from multiple perspectives:

– seeing **design** as a subject of enquiry that can be studied on different levels, including the '**knowledge** level', and
– investigating **design** as a reasoning process exhibiting certain specific features and patterns that can be **modelled** on the level of agent's **knowledge**

In the following sections, the themes and concepts roughly sketched above will be discussed in more depth. The purpose of the individual sections is to define the context for the described research. Another objective of these sections is to express our commitment to explicit interpretations of the three main concepts: *design*, *knowledge* and *modelling*. The importance of this set-up is reflecting the fact that these basic concepts are commonly used in the everyday speech, and as such, they may have different meanings for different people. Below are listed our interpretations of these concepts supported by the relevant literature. The stage set-up begins with discussing the terms '*knowledge*' and '*knowledge-level*' in section 3.2.1. Section 3.2.2 introduces term '*design*' and mentions characteristics that typically occur in connection with designing. Finally, section 3.2.3 provides a brief introduction to the paradigm of modelling.

3.2.1 Knowledge – a variable medium

The term 'knowledge' and various references to 'knowing' occur in statements we make every day. These terms are usually very informal and are interpreted very freely. Such an informal stance towards knowledge was inevitably a thankful subject for investigation in many fields, including artificial intelligence (AI). Nowadays, the AI research community subscribes to the terminology of 'knowledge' coined by Allen Newell (1982). He defined this term as a part of one specific level on which information processing of an agent may be described. The new level was defined as an extension of the traditional hierarchy of information processing as it was well known in computer science (Newell and Simon 1976). Newell positioned this new, so-called knowledge level above the level at which physical symbols are manipulated to emphasise its more abstract nature.

Each level in the information processing hierarchy is described in terms of a medium being processed, basic components that can be assembled into systems, and a set of laws governing the behaviour of the components and systems. Newell defines an agent solving a problem as the system for this particular level, and its main components are goals, actions, and bodies. The medium is knowledge, and finally Newell assigns the principle of rationality as the main law on this level – the agent's actions are determined by its goals; knowledge informs the agent's actions via its goals. Knowledge level may be defined independently of the level immediately below it (i.e. symbolic level). Newell describes several interesting properties illustrating the nature of the knowledge level:

– knowledge is seen as a kind of agent's *competence* able to generate actions;
– knowledge is linked with rationality (i.e. *goal-oriented* reasoning);
– knowledge serves as a specification of *what* a symbol structure should be able to do;
– a body of knowledge on the knowledge level is *realisable by different* systems (as well as representations) on the symbol level

Newell also formulates an important property of the knowledge level, when he says that knowledge can be defined only functionally (i.e. what it enables) not structurally (i.e. in terms of unique physical objects). Thus, different symbolic structures (representations) may be assumed to fulfil the 'function' of knowledge – knowledge is not restricted to any passive medium, language or representation! It is important to distinguish '*knowledge*' as a vanilla term coupling the agent's actions together from the '*represented knowledge*' as an embodiment in some explicit representational framework. In order to illustrate the difference, a few examples below will present knowledge from several different aspects – we may even say 'different types of knowledge'. The aim is to show what a high degree of variability exists at the knowledge level, on which we base the entire investigation.

Assume that we want to describe the motivating example of diagnosing HLT disorder from section 3.1.1.2 on the knowledge level. Our task is to ascribe a knowledgeable agent (e.g. a medical practitioner or a medic) some knowledge that would account for his actions when solving this particular, non-trivial problem. First, we can ascribe to the agent the available knowledge of medicine, which includes but is not limited to knowledge of human anatomy, clinical disorders, etc. Among other things, this knowledge may contain the fact that 'thyroid' is one of the human 'glands', and it produces a 'thyroid hormone'. Perhaps, the agent may have deeper knowledge and may recall the procedure how this hormone 'affects' the metabolism of a 'human body'. Undoubtedly, such knowledge could be found in the extensive medical literature, and as such, is typically taught in the first two years at the medical school. A vast portion of this knowledge could be also expressed very formally, as for instance in the statements given in the following list:

- type-of ('thyroid, 'gland)
- has-property ('thyroid, property-value ('shape, 'butterfly-like))
- affects ('thyroid-hormone, 'metabolism, 'positively)
- IF greater (content ('thyroid-stimulating-hormone), 5.5) THEN activity ('thyroid, 'low)

In addition to this encyclopaedic knowledge that can be expressed in the majority of existing formalisms, there will surely be some knowledge that will resist most of the formal expressive means. This is not to say that one language or an expressive framework is stronger than other. It may only be easier to express certain knowledge in a certain 'user-friendly' manner. For instance, instead of attempting to locate the position of thyroid gland using formal expressions, it seems to be easier to use other means. An agent may recall something as shown in Figure 3–2. These images provide a very succinct account of the relative position of the gland in respect to other organs. Those other organs may be easier to locate, they may be more visible, or simply their positions/dimensions do not show such a degree of variability as thyroid. It seems to be reasonable to ascribe our agents also such an informal (or less formal) knowledge to complement the earlier-mentioned formal facts.

Let us now turn to the problem in question – the diagnosis of one particular type of thyroid disorder, Hashimoto's lymphocytic thyroiditis (HLT). As already mentioned in section 3.1.1.2, the agent was taught that HLT could be recognised using sonographic scans. The advice says that HLT is exhibited through changes in the texture of the parenchyma. The practitioner should focus mainly on the 'rough and mottled texture' and changes in the 'echogenicity of the tissue'. With a smaller or greater effort we could even express this rule in an explicit and formal manner (for instance, as 'IF … THEN …' clauses). As it is visible from Figure 3–1 in section 3.1.1.2, the scans show a high degree of variability mainly in respect to the 'roughness' and 'granularity' of the tissue. The generally valid differences in the roughness would be hardly expressible.

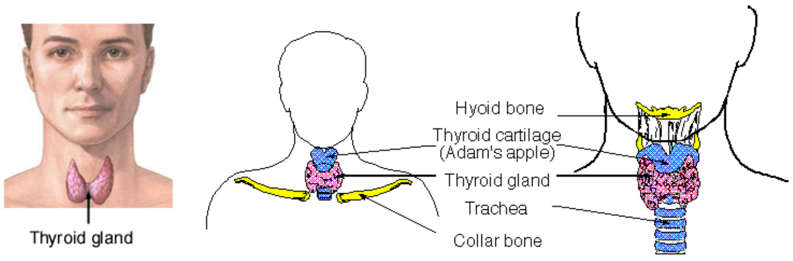


Figure 3–2. Informally represented knowledge of the thyroid position

Therefore, we shall ascribe to our knowledgeable agent also some knowledge that allows him to interpret such a generic term as 'roughness' in accordance with the thyroid-specific context. This knowledge will be restricted to this particular method of investigation (sonograph), body part (thyroid), and the disorder in focus (HLT). Similar images may be used for diagnosing other organs (e.g. X-ray scans of lungs) but in the context of a different organ, different properties may come in focus. E.g., in case of the lung scans and tuberculosis diagnosis practitioners typically look for 'shadowy regions' rather than 'rough texture'. Nevertheless, the correct interpretation of terms 'roughness' and 'granularity' is the essential part of the agent's knowledge. It may be hard or even impossible to express in some specific, measurable characteristics or formal reasoning chains how the disorder is recognised. We may say that the agent should have a 'feeling' for it – we may also say a tacit, intangible, inarticulate, experience-based knowledge.

Having an agent (a medical practitioner or a group of practitioners) solving this problem, we may start to validate the completeness of our knowledge-level model of their reasoning. As Newell (1982) says, however, the existence of a model does not mean that the agent actually 'possesses' all that vast amount of knowledge. He just behaves as if he had such knowledge as we ascribed to him in the previous paragraphs. The model would be sufficiently complete if it accounted for and could explain how the observed agents approach the given problem. Simultaneously, such a model should have the power to arrive at similar results as the agent.

Assume now that during the tests we observe some agents that perform extremely well; they reach their decisions with an almost constant precision and confidence. According to our model, they may have an extremely efficient 'inference engine' that can work very fast. Alternatively, they may have additional, 'experience-based' knowledge that gives them the answers very quickly. However, due to the large variability of the scans and the steady level of precise diagnosis these two suggestions lose their power. If our knowledge-level model of the agent cannot give the answer to the exceptional behaviour, perhaps the model itself lacks an important bit that was overlooked before. As if we forgot to ascribe some knowledge to the agent that was used…

An example of a 'hidden' or 'forgotten' knowledge' may be, for instance, the fact that HLT is an inflammatory disease, and as such can be proven by a presence or absence of certain antibodies in the patient's blood. Thus, in addition to the non-invasive diagnosis using sonographic scans, we learn that invasive needle biopsy is indeed an alternative way how to ascertain HLT very precisely and rather quickly. This is entirely straightforward knowledge that can be described by a formal procedure of taking a blood sample and conducting precise chemical analysis of its composition. Only in case of our initial knowledge-level model, it was not taken into account. In other words, although being 'clear, formal and straightforward', as if it has not been 'expressed' in our initial perspective.

Let us now summarise what 'types' of knowledge were identified in the simple motivating example. First, we distinguished between knowledge that is valid in a specific, *narrow context* and knowledge that is more *general*, applicable in many contexts. For instance, knowledge that 'thyroid' is 'a gland' has clearly a broader scope of validity than the fact that chronically inflamed thyroid exhibits local disorders in the parenchyma that appear as black mottled areas. Another category for differentiating types of knowledge mentioned in the paragraphs above is whether we are able to express the knowledge in some existing representation. For instance, we stated that generic facts and rules could be *expressed easily* in any formalism, including predicate calculus. On the other hand, knowledge for appreciating 'roughness' of the thyroid texture might be *more intangible*, even skilled practitioners may have problem to communicate this form of knowledge in the same representation as the rules, for instance. We mentioned that this feature is often ascribed to experience-based knowledge, and in certain case also practitioner's skills.

Another category that was distinguished above was the level of formalism chosen for representing a particular chunk of knowledge. Knowledge expressed in form of production rules or chemical formulae is far *more formal* than the sketch showing the position of the gland. The sketch is rather *informal* taking into account the extensive encyclopaedic knowledge of modern medicine. Nevertheless, it is very informative and has a high expressive power. Sketches are also a widely used mechanism for representing knowledge, particularly popular among engineers and designers (Goel 1995).

The next interesting category looks at the fact whether particular knowledge is actually *expressed explicitly* in the current problem-solving task. In some situations, it may happen that a designer or another problem solving agent 'possesses' a particular chunk of knowledge but from various reasons this chunk remains 'inactive' – *not expressed* and represented. In other words, the agent behaves as if he was not aware of potentially useful knowledge, whether it is represented formally or informally.

Table 3–1. Classification of knowledge sources

Knowledge category	Applicable adjectives/characteristics
Contextual dependency	context-free, contextual
Explicit representability	explicitly representable, tacit
Explicit presence	explicitly represented, implicit, 'hidden', 'forgotten'
Level of formalism	formal, informal

Let us therefore conclude the section on 'knowledge' by summarising the characteristics that may occur in connection to describing knowledge. The breakdown depicted in Table 3–1 shows the dimensions or categories for knowledge classification. The concepts in the first column correspond to different dimensions that can be distinguished when talking about knowledge. Terms that are listed in the second column are the extreme ends of spectrum rather then crisp categories. In addition to the multiple dimensions of knowledge as a medium, there are certain correlations among the adjectives taken from the different dimensions. For instance, tacit knowledge has been defined as something that cannot be expressed in any formalism, something that is not explicitly representable and expressible. From such a definition, it follows that it does not make sense to talk about 'formal tacit knowledge' and 'informal tacit knowledge' – the level of formality is a meaningless concept when talking about tacit knowledge (i.e. feelings, experiences, etc.).

3.2.2 Design – a subject of enquiry

Before reviewing the existing research on design issues in chapters 4 and 5, let us adopt a suitable definition of design. There are plenty of different definitions of what design actually is – almost as many as there are design researchers and publications about design. Nevertheless, it is possible to formulate different views in a broad and a narrow definition. For the purposes of a broad definition of design the opinion of Herbert Simon (1969) extended by Smithers, Conkie *et al.* (1990) is very expressive:

> "*Design is the concern of all occupations and professions that are engaged in converting the actual into preferred situations.*"
> "*Design task generally occurs when an external agent determines to change the status of its surrounding world. The main purpose of design as an activity is to deliver some outline how the desired changes can be accomplished.*"

According to the broad definition, design is a goal-driven, problem solving activity that covers also various specialised activities such as re-engineering or configuration. However, it is possible to narrow down the basic definition to express only the features that were long associated with the traditional design occupations such as architecture, engineering, or more recently computer programming. For its formulation, we use the words of Donald Schön (1983):

> "*A designer makes things. Sometimes he makes the final product, more often a representation – a plan, program or image of an artefact to be constructed by others. He works in particular situations, uses particular materials, and employs a distinctive medium and language. Typically, his making process is complex.*"

These definitions of design as a task are sufficiently domain-independent, and apply equally well whether town plans, technological processes or devices are designed. Chandrasekaran (1990) breaks the definition given above into smaller pieces, and expresses the essential aspects of design that suit particularly well the majority of engineering problems. He specifies a design task by:

a) a set of *functions* to be delivered by an artefact (incl. those explicitly mentioned and those implied by the problem domain);
b) a set of *constraints* that need to be satisfied by the designed artefact; and
c) a *technology* – i.e. a repertoire of components available and relations among them

According to Chandrasekaran's definition, the constraints may address design parameters, process of designing or process of manufacturing. A design solution is formulated in a form of used components, modules, and relationships among them that ensure the satisfaction of the desired functionality. A set of design components is considered a solution if it satisfies also the explicit and domain-specific (implicit) constraints. However, as Goel (1994) notes, design is in most non-trivial situations under-specified; i.e. the desired functions or applicable constraints are often incomplete. In some cases, in addition to the incomplete goals, the repertoire of available means is open-ended too. A new artefact may be designed not only from the existing elements, but new elements may be recursively designed and developed to deliver certain sub-functions of a larger artefact.

The open-ended nature of some design tasks prompted a few researchers to distinguish different 'classes' of design problems. For instance, Goel (1994) and Simon (1973) distinguish well-structured (trivial) designs from the ill-structured (open-ended) ones. Other researchers promote the breakdown that became almost 'a cliché' in the 'AI in Design' community (Gero 1990; Goel 1997). According to their division, design may be categorised as one of the following:

a) *routine* … sets of design functions, components, variables and their values are fixed,
b) *innovative* … essential functions and components are fixed but there is space for introducing something new into a given framework, or
c) *creative* … newly introduced design components, variables, or values go beyond the existing framework, and significantly extend it

In order to address the variability of design, we would like to propose an extension to the above-mentioned definitions, and define design as a goal-oriented process leading from the *initial problem specification* toward an *acceptable design solution*. The specification includes the above mentioned desired functions, properties, and constraints; whereas solution is seen in a broader sense. A design solution includes the *refined problem specification*, selection of available or designed components for achieving this amended specification, and process leading to the formulation of the two. The 'design process' is gluing together the evolving specification of design goals with the evolving specification of means (components, structures). This process represents 'a significant effort the designers make in order to give their problems a structure before they start looking for a solution' (Simon 1973).

3.2.3 Modelling – a paradigm for the enquiry

Word 'model' has many meanings in our everyday speech. According to the Merriam-Webster common dictionary, one meaning of the term 'model' is 'a copy' or 'an image'. A similar view can be found also in Klir's (1985) book on general systems theory. Klir argues that in systems theory the term 'modelling relation' is often used isomorphically with the term 'similarity relation'. Two systems are similar if they exhibit some commonalties and can be converted to each other by some suitable transformation. For instance, system ***X*** may be the original and system ***Y*** may be another system, which is similar to the original and may act as its substitute in respect to some essential features. The system ***Y*** is typically called '*modelling system*', and together with some relevant transformation makes up *a model* of the original system ***X***.

Klir also argues that modelling is essentially a relation of equivalence, and as such exhibits all properties of the equivalent relations (including symmetry, transitivity and reflexivity). Whether or not a particular system is 'a model' is usually a subjective, context-dependent decision of the creator and/or user of the model. If the system ***Y*** is able to substitute the original, and perhaps has some advantages over it (e.g. is cheaper, less dangerous, easier to understand, etc.) then it may well be a model of the original system. In other words, for declaring a system being a model requires having some original.

On the other hand, it is obvious that one original may have several different models that differ in one or more aspects (e.g. complexity or precision).

From the perspective of information processing, models may be developed on any level, including the earlier mentioned symbol and knowledge level (see section 3.2.1). The usefulness of modelling as an approach is shown in the following very simple example. Assume that we want to design a structure made from reinforced beams for a multi-storey building. This structure must be sufficiently robust and safe to support the walls and several floors, as well as withstand winds of given strength. Instead of constructing such a beam structure randomly and then measuring whether it satisfies given constraints, the practitioner creates a model of the structure. The structure characteristics are calculated using some mathematical model rather then measured on the produced structure. Various differential and algebraic equations for the calculation of the beam behaviour act as a model of the original beam construction. The equations give an answer whether the structure complies with the constraints without the need to implement and build the actual beam construction. In addition to giving a similar answer as the measurements on the real beam, they are much cheaper, safer, and faster to work with.

The other well-known category of models includes 'scale models'. In this particular case, a real physical object (e.g. a body of a car or plane) is replaced by its scaled counterpart. Instead of performing the aerodynamic tests on the original object, the engineers may obtain some initial picture from the scaled model. The scaled model of a car exhibits similar characteristics as the original in respect to the aerodynamics but it may completely fail to account for other features (such as driver's safety during a crash). Thus, for different purposes, it may be necessary to construct another model that may be used in the crash and safety tests.

Nevertheless, this short section was presented here to emphasise that according to general systems theory, models are ***not*** the *simplified versions* of the original systems. Models, including knowledge-level models, must be *equivalent* to the original system, albeit only in a selected sub-set of aspects. It may be easier or faster to work with a model than with the original, but this does not mean that a model must be necessarily a 'simplified' version of the original. For example, the mathematical apparatus for the calculation of beam behaviour is as complex (or probably even more complicated) as the original, physical beam structure. The inherent purpose of modelling is to develop systems that act similarly as an original rather than being 'simpler'! Thus, in our research we are developing a model that should account for certain knowledge-based actions similarly to a human designer.

3.2.4 Caveat on epistemologies

Our stance to the phenomena of 'knowledge', 'design', and 'models' is in accordance with an extensive study on different epistemologies of knowledge that is reported by Cook and Brown (1999). Both authors distinguish the epistemology of '*possession*' from the epistemology of '*practice*'. In other words, both aspects of knowledge are important – in order to solve problems, we need to *have* some knowledge, as well as *use* it in action, and *perform* a 'knowledge-based action'. The following quote from Cook and Brown's paper highlights the essence of this epistemological stance:

"*An accomplished engineer may possess a great deal of (...) knowledge; but there are plenty of people who possess such knowledge yet do not excel as engineers. (...)*

This means that if we want to understand the essentials of what accomplished engineers <u>know</u>, we need to look at what they <u>do</u> as well as what they <u>possess</u>."

This short extract from the Cook and Brown's paper informs a significant part of the research reported in this thesis. From this source originates also our perception of design, as defined earlier in section 3.2.2, which treats it is an evolving interplay of problem specification and solution. Design is not only about the initial specifications or about the final solutions; it is also about the ways of reaching the suitable and sufficient problem specifications as well as acceptable solutions. According to this perspective, design is an enquiry that draws on the extensive knowledge acquired by the design practitioners. At the same time, it is an outcome of that knowledge *applied in action* (i.e. evolving process of design-*ing*) rather than an outcome 'implied' by the possessed knowledge that is proportionate to the size of the repository.

These are also the reasons why we do not attend to another important phenomenon that often coincides with the term 'knowledge' – a *representation* of knowledge. Knowledge representation has been a popular subject for many research works (Mylopoulos and Levesque 1984; Gero 1990; Gruber and Russell 1991; Chandrasekaran 1993). However, we want to argue that the utility of knowledge is in its presence in a particular (problem-solving) action, and not in its possession expressed by any known representation. Although this view on the phenomenon of 'knowledge' is slightly opportunistic, it is fully supported by Newell's (1982) observation that sees knowledge as '*a competence-like notion being a potential for generating actions*'. In this context, 'competence' is a term that needs to be understood as an ability '<u>to have</u>' and simultaneously '<u>to exercise</u>' (see Chomsky's thought in (Newell 1982)).

Omitting further details of Cook and Brown's work on characterising 'possessed' knowledge in terms of individual vs. group and explicit vs. tacit, we conclude this discussion with a reference to an expressive 'representation' of the above-mentioned interplay. In Figure 3–3 on page 29, knowledge 'possessed' by an agent is categorised in a two-by-two grid, in line with Cook and Brown's claims. Each quadrant may be used in an action as a tool for the interaction with the world. Action gives the possessed knowledge its utility, and vice-versa, the possessed knowledge informs and orders the agent's actions. The influence is reciprocal not only between 'possessed' and 'exercised' knowledge but also among the different types of 'possessed' knowledge. However, the diagram of interplay seems to be a more realistic 'model' of the actual relationships between the different types. The scheme on the right hand side of Figure 3–3 models a reciprocal influence of the different types of knowledge, as well as generation of one type *using* another one. According to the figure, such generation occurs *in an action* backed and justified by the agent's knowledge. The reciprocal influence is depicted by curved arrows printed in boldface; the generation through an action by outward pointing straight arrows.

The implications of the duality of knowledge possession and knowledge exercise are reflected in our proposed framework in the very essence of the approach. We argue that a design problem cannot be articulated 'once and forever' in any given conceptual frame unless the designer applies the particular frame, and starts interacting with the artefacts that may be constructed in the current frame. The need for changing the current frame is not caused by the lack of possessed knowledge. On the contrary, this need is triggered by the knowledge applied in action, and draws on the 'possessed' knowledge to articulate a way out from the current deadlock.

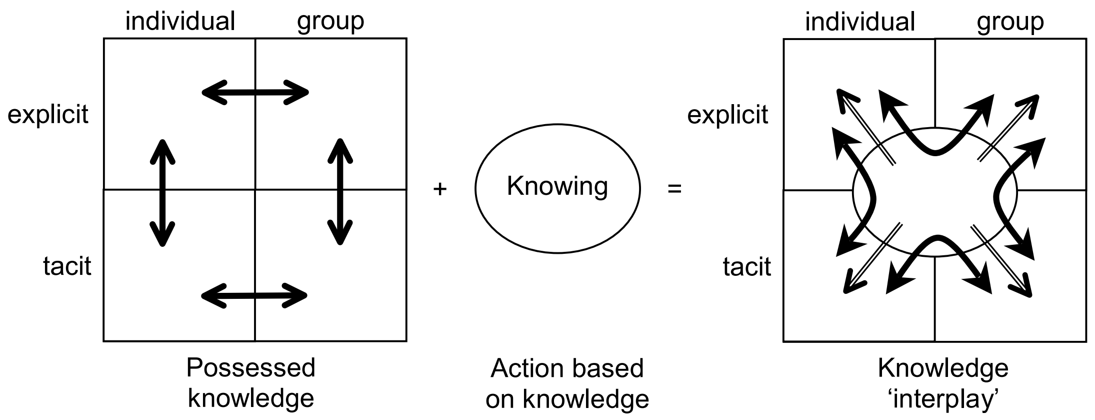


Figure 3–3. Knowledge and action interplay (adapted from (Cook and Brown 1999))

The paradigmatic shift from the 'transformation' of knowledge to the 'generation' of knowledge is the most significant factor that differentiates this approach from others. Using the terminology from the motivating examples described in section 3.1.1, the explicit knowledge of diagnostic rules, artificial dyes or keeping balance on a bike does not replace the tacit knowledge ('feeling') of a practitioner. Similarly, the tacit experience and feeling of the acceptability that a practitioner had developed for 'roughness of texture' or 'dye composition', cannot be simply 'explicated' by a rule or another form of representation. What can be done however, is to apply the tacit in an action, and acquire (i.e. generate) some explicit knowledge consequently. Alternatively, the explicit when tested through an action may help in the 'acquisition' (i.e. surfacing and criticising) of the tacit, inarticulate feel.

3.3 Contributions of the research

Having defined the basic terms and set up the stage for the remainder of this document, we may now formulate the goals of the research as well as the main contributions. This research shows a potential to contribute in several aspects to the cross-disciplinary foundations formulated in section 3.2. We perceive the contributions from two distinct angles rather than separated contributions to the isolated disciplines. First, we articulate the overall contributions, and in sub-sections 3.3.1 and 3.3.2 highlight and discuss the basic contributions from two perspectives.

- **Contribution 1**:
  Formalisation of the conceptual apparatus for representing and modelling design tasks and the articulation of a knowledge-level model based on the defined concepts. This contribution includes definitions of such terms as for instance, '*conceptual design frame*', '*problem specification*', '*problem solving theory*', '*solution admissibility*', or '*solution acceptability*'.

- **Contribution 2**:
  Empirical study and analysis of design tasks in terms of the formal concepts and operations introduced/defined within contribution #1. This category also features annotations of the experimental studies that preserve the contextual sequence of the designer's decisions.

- **Contribution 3**:
  Definition and empirically supported extension of the recursive model of conceptual framing in design featuring several patterns or *schemas* of reasoning strategies observed in the

experimental sessions. This contribution aims to model problem re-interpretation and re-framing in terms of interactions between a triplet of simple predicates (knowledge actions) – *specifies*, *satisfies* and *acceptable*.

Let us now look at the manifestations of these contributions in two perspectives. First, a narrower perspective of design theory and problem solving strategies specific for the design tasks is presented in section 3.3.1. A broader perspective contributing towards the research into knowledge-level modelling of the inarticulate reasoning processes follows in section 3.3.2. Since the recursive model resulting from the partial contributions is an essential outcome of the thesis, a separate section (3.3.3) is devoted to the re-iteration of the main characteristics of the model.

3.3.1 Contributions to design theory

The essential contribution to the research in design problem solving is in the attempt to articulate and justify many of the reasoning steps that are in other theories presented as obscure and intractable for any explicit representation. We shall show in the subsequent chapters that although some knowledge used in the complex reasoning steps of a design process may not be expressible, there are formal means for modelling the application of this tacit knowledge. In the list below, some of the partial contributions to the research on the theory of design are summarised:

- **Recursive model of framing in design** … An iterative model of design and designing is proposed as a recursive interaction of decisions and actions with the knowledge that is possessed by the designer. In addition to the explicit knowledge that may be encoded using some explicit representation (e.g. predicate calculus), the model of framing tries to account also for complementary knowledge sources (such as past cases or tacit expectations) often featuring 'inarticulate' knowledge. Whilst tacit knowledge is hard-to-encode in any available formal representation it may be 'acquired' from these complementary knowledge sources 'in action' – i.e. by applying it. One of less common forms of knowledge representation that has been used in our experiments has had a form of structured 'debate' where individual records represented the designer's (often informal) justifications of the design decisions.
- **Formal definition of typically 'vague' operations in design** … Some of the operations are identified that typically obscure the design as a knowledge-intensive task, and cause difficulties with the application of common searching techniques. The analysed operations play an important role in transcripts of our design experiments, and share one feature. Namely, they involve the designer's tacit knowledge to a certain extent. Therefore, a more in-depth account of such operations is useful for understanding of how tacit knowledge is deployed in real-world design, what constitutes it, and what is its context. Among others, we articulate a concept of knowledge-level, *conceptual design frames*, and their amendments.
- **Enhancement of design task models to account for 'realistic' designs** … The current design theories and existing generic models of design task are extended in order to take into account the incompleteness and open-ended character of the design problems in practice. We justify the need for such an extension by arguing that design is more about *sufficiency* and *acceptability* rather than optimality and completeness. This argument supports the theory that the problem

specifications and solutions *co-evolve* so that they are eventually acceptable for the designer. This perspective may be contrasted with a view when only the evolution of a design solution is taken seriously, and the solution *admissibility* can be easily assessed against the explict problem specification. We attempt to formulate the underlying principles of an operation that many cognitive scientists call '*solution talkback*' (Nakakoji, Yamamoto *et al.* 1998).

- **Approach to 'design' as an integral part of 'designing'** … It shall be argued that design is an integral part of the actual exercise of designing, i.e. applying the design knowledge for and in the action of design-ing. In other words, the perspective of the investigation shifts and pays attention to all aspects of a 'design case' – problem specification, solution/designed artefact, and the actual process of designing. A product of a successful design exercise is typically referred to as the *design solution* (designed artefact). However, the equally important by-product of a design exercise is the refined and extended specification of the problem, as well as the sequence of justified decisions that led to the successful accomplishment of design.

3.3.2 Contributions to reasoning and knowledge-level modelling

The fundamental contribution to the research from the perspective of knowledge-level modelling is the attempt to develop a model of reasoning that accounts for the different types of knowledge and their interaction. Since design is a prime example of a complex, poorly-structured form of reasoning, the findings may to certain extent be transferred also to other ill-structured and/or incomplete problems. This contribution can be further sub-divided into following statements that capture the main tenets of the research in reasoning and knowledge-level modelling of complex problem-solving tasks:

- **Interplay between different types of knowledge** … This investigation shall show how the tacit awareness of certain common features may be raised during an application of explicit, familiar knowledge of previous designs, and a suitable form of knowledge transfer. Rather then making the tacit knowledge explicit, this part of the research demonstrates how one type of knowledge (such as explicit design cases) may be successfully used for the acquisition and development of complementary types (such as tacit 'feelings'). The claims are validated by a set of experiments, in which designers articulated and recorded their decisions both formally and informally. Many informal threads show the references to various complementary knowledge sources, including tacit 'feelings' and expectations.
- **Extended perception of case-based approach to design** … As already mentioned earlier, previous design cases are a rich source of knowledge for any subsequent designs. However, the re-use and adaptation of a previous design solution for the current problem is only one form of knowledge re-use. In certain situations when the design problem is under-specified, it may even be impossible to re-use the solution from any previous case. Instead, it may be more beneficial to use a known specification of the past problem as an 'inspiration' for better understanding of the current task. In other words, a problem specification may also be re-used, not only the actual solutions. Re-use of the concepts from the past problem specifications helps to interpret and *frame* the current, ill-structured design problem.

- **Introduction of a term 'tacit inference'** … This inference is presented as an important form of reasoning suitable for the (re-)interpretation of the non-trivial, possibly incomplete and inconsistent problems. A typical feature of this reasoning is that in addition to the explicit knowledge, a bigger or smaller role is played by tacit, intangible knowledge, designer's experience, 'skill' or informal expertise. Since tacit knowledge cannot be subjected to the usual inferential mechanisms, it is necessary to find a way around. Often, a different view (*frame*) is needed to perform this specific form of reasoning. In other words, the designer may understand and criticise his or her tacit knowledge by referring to less familiar or less common frames. We use the expression '*understand and criticise*' deliberately to distinguish it from 'acquisition' or 'explication' of tacit knowledge (see also section 3.2.4).
- **Conceptual schemas of problem re-interpretation** … This contribution constitutes important progress in the area of modelling the ill-structured reasoning processes. It defines three distinct forms for the problem re-interpretation that foster the earlier mentioned interaction and recursion. The schemas range from the 'well-defined' problems that require little frame changing, to a frame refinement and the extension of explicit problem specification via reflection and re-framing. The most difficult form of problem re-interpretation features an amendment of the conceptual vocabularies for the particular design frame.

3.3.3 Implications for the recursive model of design

An essential outcome of this thesis is the theory of recursive framing in design featuring the conceptual design frames that were already mentioned in the preceding sections. Let us therefore briefly summarise the main aspects of this theoretical model that underpins the content of the subsequent chapters. The term 'recursive' demonstrates the fact that there are certain elementary knowledge-level 'operations' being repeated throughout the design task. According to our observations, these operations are mutually related to conform to certain usual 'patterns'. The model is not entirely deterministic; it does not prescribe any fixed path that guarantees to arrive at a solution. Depending on what operations take part in the design and the order, in which the individual operations are carried out, different behaviours may be modelled. Nevertheless, the behaviours are the result of a holistic effect of the recursive sequences rather than the individual decisions.

The model may be roughly broken into two parts. The first one and the one that is ***not*** elaborated in this thesis deals with a design problem on the explicit level. This topic has been well researched by numerous researchers working in the field (Bhatta, Goel *et al.* 1994; Wielinga, Akkermans *et al.* 1995; Brazier, van Langen *et al.* 1996b; Motta 1997). This usual perspective on modelling design process is then complemented by an extension featuring reflection and re-framing, and this is the main focus of this thesis. It accounts for selected features of design problems mentioned especially in the cognitive literature (Reddy 1988; Fischer 1992; Candy and Edmonds 1996; Nakakoji, Yamamoto *et al.* 1998).

Nonetheless, we show briefly that the explicit level may be modelled by interplay of three logical operations: abduction, deduction, and maintenance of logical consistency. Since design is an iterative task, these operations may be repeated several times; for example in order to 'fix' a conflict or inconsistency in the logical theory. Thus, there may be an 'informed abduction' that rectifies the ambiguity of a common abduction by taking into account the expected and desired functions. Similarly, there may exist an operation of 'restricted deduction' rectifying the main disadvantage of deduction – a vast amount of possible consequences that may be inferred from any given premises. Finally, we talk briefly of an assumption-based extension to the truth maintenance, which rectifies the inconsistencies in the large logical theories by temporarily splitting the inconsistent parts into separate *contexts*. Nevertheless, we do not pay any 'deep' attention to these explicit reasoning techniques. The main reason is the fact that such techniques are widely used and well investigated. These logical operations are described in many texts on the theory of predicate calculus (Levin 1974; Mendelson 1979; de Kleer 1986a; Genesereth and Nilsson 1987).

A substantial part of the thesis is devoted to the theory of recursive framing in design, which incorporates the earlier-mentioned ill-structured extension to the existing approaches to modelling design. In the recursive model of framing, we look mainly at how the explicit reasoning strategies may be related to the tacit ones. Especially, we focus our attention on the issues of re-structuring or *re-framing* of the current design problem. We agree that the earlier-mentioned triplet of logical operations may be used to solve a design problem, but we define this problem solving as an 'operation' *satisfying* only the explicit specification associated with the ill-defined problem.

This level of explicit satisfaction is complemented by another knowledge level that looks at how to *specify*, understand, and interpret the vague design problem. We articulate several important assumptions that enable us to see design as an interaction of these two predicates (knowledge-level actions). Perhaps the most important one is that any *satisfaction* or *specification* makes sense only within a particular conceptual frame. Thus, instead of 'solving a design problem' we talk about 'satisfying the explicit specification of the problem that was made within a certain frame, perspective'. To reflect the uniqueness of the 'frame-bound' specification and satisfaction, we introduce a concept of solution *acceptability*. We argue that this is a tacit assessment that compares and criticises the accomplished (partial) results against the original design problem. As if the designer asked a question: "*Is this solution really the one I wanted, expected for this particular problem?*"

If an answer to such a question is negative, we argue it points to an incomplete explicit specification of the design problem. The problem must be re-interpreted, and we propose several techniques modelling this iterative refinement of the designer's understanding of the problem. In the extended RFD model, we show that one form features *conceptual re-framing*, when the axiomatic and conceptual apparatus of a logical theory may be amended. The modification is typically performed by the introduction of new concepts and/or relations to the problem solving theory. These new concepts are then used as building blocks in finding a new candidate solution satisfying the re-framed problem, which is again subjected to the 'acceptability check'.

We will show that it is possible to allow modifications of both the explicit problem specification and solution. Both amendments may occur within a single design frame, or often in a sequence of reasoning steps involving design re-framing. At this stage of the thesis, *re-framing* may be viewed as stepping above the design problem and looking at it from a different angle. A different angle may bring forward some unattended and neglected features of the problem that may eventually lead to an interesting and acceptable solution. In the literature, this operation is roughly equivalent to an ‘insightful’ decision. Such ‘insights’ and ‘creative leaps’ are some of the characteristics that are typically associated in the literature with non-trivial design problems in the different domains. Schön (1983), for instance, bases the whole ‘art of designing’ on the designer’s ability *to reflect*, i.e. to move between the different perspectives and articulate new, potentially useful concepts.

The characteristics and the essential assumptions in respect to design are elicited from the literature reviewed in chapters 4 and 5. The initial definition of the above-mentioned concepts and the recursive model of framing are given in chapter 6. Next, the validation and evaluation of the claims is presented in the annotated and analysed empirical study that can be found in chapters 7, 8, and 9. Finally, chapter 10 shows takes into account the experimental observations, and presents an extended version of the recursive model of framing, including the above-mentioned *reasoning schemas.*

3.4 Position of the research

In this section, we briefly mention the research in other related fields that complement the work reported in this thesis. The full literature review is contained in chapters 4 and 5. One of the main objectives of this section is to note what the research described in the further chapters is *not* addressing in details. At the same time, the existence of some parallels between the different approaches must be acknowledged. We look at some of the techniques that were proposed in order to tackle under-specified problems, such as circumscription or default logic in section 3.4.1. In addition, in section 3.4.2 some of the related disciplines are mentioned, including ethnographical research in design, participatory design, and human-centred approach. It is not the intention to argue against or for these techniques, or evaluate them according to any specific criteria. The objective is to position the presented work among the other efforts of the establishing community of research in design. Note that the full literature review appears in chapters 4 and 5.

3.4.1 Notes on handling incompleteness

Since our initial ambition was to avoid the acts of magic and various ‘force majeure’ as the means for solving ill-structured and incomplete design problems, we first looked at various existing measures for tackling the incompleteness, vagueness, and typicalities. What particularly caught our attention were various non-monotonic reasoning mechanisms, such as default logic (Reiter 1980) or circumscription (McCarthy 1980). These mechanisms attempt to augment the common predicate calculus so that it can handle the real-world scenarios with defaults, incompleteness, or uncertainty. As McDermott and Doyle (1980) define in their paper, a reasoning system may be perceived as *non-monotonic* if a newly introduced axiom may invalidate the existing theorems. In other words, imagine the following scenario – a new observation is made in a particular world, and we use a non-monotonic logical system to reason about the properties of such a world. Due to non-monotonic nature of the system, we are bound to re-visit and re-validate all the inferences that were constructed in an original, incomplete theory that did not account for the new observation. Some of the previous inferences may remain valid in the extended theory, others may lose their validity – i.e. a non-monotonic extension of a logical theory may change our beliefs in certain logical consequences.

However, we find that these augmentations are powerful as long as only the explicit knowledge about the current task is incomplete, and there exists a ‘complete’ knowledge base, from which we can take the missing information (e.g. a default value). Similarly, the circumscription is working fine as long as we want to restrict our attention to the existing objects, and nothing else is needed for the resolution of a problem. The most important feature that is not present in these approaches is the ability of the logical systems to generate a new perspective by adding new knowledge and extending the current logical theory. Note that by ‘knowledge generation’ we do not mean only a retrieval of a fact from a ‘complete’ knowledge base! On the contrary, this ‘generation’ may affect the conceptual foundations and the conceptual vocabulary of the problem solving theory. More information on the research techniques mentioned above is given later in section 4.2.1.

In order to pay credit to circumscription, we acknowledge that it might be able to conjecture the conclusion that would go along the following lines. “If we had had this particular information available then we would have been able to deduce such and such (conjectured) conclusion.” This ability to conjecture something hypothetical is very powerful; however, since the circumscription is dealing only with the substitutions, how can we bring the particular information that would trigger the conjecture to the attention of the circumscribing operator? Consider the following situations that may constitute a possible framework for the further reasoning:

1. our logical theory is complete, and the only thing to do is to go through all predicates that exist in the theory and find the ‘right’ one(s);
2. we give up the logic, and refer to the human being as an unquestioned ‘force majeure’ that performs such a conjecture and delivers the result;
3. since the problem is not soluble in the current logical theory, we step outside the logic and use additional knowledge sources that may be available – they may not necessarily be expressible in the language of the current logical theory

Alternative 1 corresponds to an ideal state. As Tomiyama (1994) puts it in his paper, this would assume the existence of a ‘super-human’ who knows everything. Clearly, if this were the case, there would not be any need to worry about incompleteness. Any given task would be only ‘locally’ incomplete with regard to the explicit knowledge the designer has at any given moment. The design task in this context would be indeed a task of searching the knowledge base and explicating the relevant bits so that they extend the currently incomplete (explicit) design problem specification. We are sceptical about the super-humans or ‘complete’ knowledge bases, and do not consider this scenario for addressing incompleteness as plausible.

The second scenario (see point 2 above) goes to the opposite extreme. It gives up all the explicit theory and refers to a knowledgeable but often unquestionable authority. By referring to the authority, it is possible to make decisions, but not to explain or justify them. Between the lines of the description of this scenario, we see a reference to the artistry or talent someone is born with. In other words, the decision of a higher authority is often intractable to an external observer, and this scenario does not spend any effort to understand it. The decision is to be accepted as a given axiom, not discovered, proven, or in general, modelled and justified on the knowledge level. Note that this scenario is hypothetical, it does not relate to any research on user-centred design or other subtle HCI issues.

Similarly to the first scenario, the second scenario does not really give an answer to the question why and how the current perspective may be shifted. Therefore, the last remaining scenario (see point 3 above) might be the needed compromise. It says that there may exist some knowledge that cannot be encoded in the particular (e.g. logical) theory because it may be talking about the theory itself. Such ‘additional’ knowledge corresponds to various skills and hard-to-articulate experiences. This scenario is a popular subject of investigation in many research disciplines, including HCI or participatory design (see also section 3.4.2). These disciplines use various methods to acquire and/or describe these skills, as well as the associated experiential knowledge.

The third scenario assumes an existence of ‘*meta-knowledge*’ that could be deployed in order to manipulate the explicit (problem solving) ‘levels’ of knowledge, and thus shift the conceptual base of the design problems. This ‘meta-knowledge’ can also exhibit realistic capabilities for generating new information upon which a conjecture can be constructed at a later stage. However, there are several issues to be raised about it. For example, what is its form and how can it be modelled on the knowledge level? Let us consider the following situations as examples of the possible refinements of the scenario number 3 from the list above:

3a) Higher-order knowledge will be also in the form of a logical theory. In addition to expressing statements about the objects and relations in the current world, it may express some facts about when and how a particular axiom (conceptual object) can be amended.

3b) Higher-order knowledge will be formed by some generalised statements that are developed for a particular class of problems. These may represent various typical strategies, heuristics or ‘best practices’ for the navigation in an incomplete design space.

3c) Finally, this knowledge may be surfaced and critically applied using a reference to a large (not necessarily complete) base of previous, familiar problems; e.g. through reasoning by analogy or similarity, case adaptation or generalisation.

The approaches briefly outlined in the above list differ with regard to their dynamics. Knowledge described in the scenario 3a seems to be most static from all three forms, and too prescriptive to account for a flexibility and creativity in design. For example, it may have the form of various rules for choosing an appropriate alternative if a particular violation occurs. This form is not rare in design. However, it is typically associated with routine, straightforward design tasks that are well defined, and it is possible to foresee the violations and fixes. An example of such a design problem is discussed later in this document – see section 6.3.1 and Appendix B describing design of an elevator in the Sisyphus-2 domain. In the elevator example, the higher-level knowledge is referred to as *design fixes*, and in general most of the constraints articulated for the design problem, have an associated fix. The fixes are applied deterministically when a particular constraint is violated by the partial solution.

Scenario 3b with the default theories and common strategies seems to be a more realistic approximation of a design process. Heuristics and best practices may be more powerful than design fixes from the scenario 3a. In the existing research this scenario is represented, for instance, by the research work of John Gero (1990) on design prototypes. A prototype represents a class of products, contains its most significant features, relationships etc. However, a prototype is not a fully-fledged, off-the-shelf solution. Typically, a design prototype is a kind of skeleton for the further development and refinement. It is a point in the design space, from which further exploration may begin. Prototypes and various heuristics are very important in design practice, and they have potential to introduce new variables, parameters or concepts into the problem solving theory. Choosing a particular heuristic or prototype may focus the designer's attention on a specific constraint, which in turn, may serve for the purposes of a circumscriptive conjecture.

Thus, design prototypes, design heuristics or best practices are one 'class' of the strategies for conjecturing a further step. The prototypes may carry inside the information about default values or typical solving methods for a particular class of problems. Another knowledge source that has potential for discovering new, unusual conjectures for a particular design problem are specific design cases tackled in the past. The research conducted in line with scenario 3c covers such areas as case-based reasoning (Riesbeck and Schank 1989), case-based design (Watson and Perera 1997), or reasoning by analogy (Falkenhainer, Forbus *et al.* 1989). Often, these approaches exhibit the greatest degree of flexibility and dynamics as well as learning capabilities. These features are highly desirable and closely connected because any new design case that is stored in the repository can influence the results of the reasoning process in the future. In this sense, we may understand the process of extending the case repository as a specific form of learning from a direct experience.

A repository typically contains design cases, as the designers understood them in the past, when they were tackling them. Therefore, it is possible to make a claim that in addition to the individual designer's explicit knowledge such a repository may also contain group knowledge of a particular organisation, and it may reflect the designer's tacit, 'hard-to-articulate' knowledge. The paradigm of reasoning by analogy enables the designers to re-use the previous cases in many different contexts. Moreover, designers may take different perspectives to understand and explain a previous case and its relationship to the current problem, and eventually they may discover novel or unusual extensions. The previous design cases may be instrumental in interpreting (*framing*) the current problem in a usual or unusual way, using different conceptual frames. Nevertheless, further details of the design frames and framing are coming in chapter 6, later in this document.

3.4.2 Research perspective of HCI and related fields

The emergence of human-centred approach to design and deployment of the ethnographical methods came in the late 1980s as a reaction to the reductionist investigative techniques that prevailed until then. A typical reductionist's approach – the waterfall model of design is addressed in section 4.1.2. This new approach is more 'human-oriented' rather than 'process-oriented'. It does not attempt to break the activity of designing into an algorithmic sequence of simple operations; neither does it try to impose various task models on the task. The justification of this shift in the focus rests on the observation that many artefacts that involve the participation of humans are not open to a simplistic reductionism of the task analysis. Such a simplified stance had long been a primary technique in the analysis and design of technical and technological systems (Jagodzinski, Reid *et al.* 2000).

The main objection the new approach directs towards the more widespread analytical techniques is that any reduction of design into simple operations tends to be mechanistic. As such, it typically fails to recognise the tacit and shared aspects of the designer's work. Also, the operation- and task-based analysis often forgets to account for various organisational preferences and practices that play a crucial role in the real-world design, and that are not always explicitly accountable for (Groenbak, Kyng *et al.* 1993). In order to avoid the mistakes of the usual analytical approach, the new techniques demand a more holistic picture, and investigate the design as a social activity in a natural setting. Jagodzinski, Reid *et al.* (2000) put this point very clearly in the following quote:

"*It was realised that many of the crucial issues in the progress of projects arose from social activity, such as design review meetings and ad-hoc technical meetings…*"

In this thesis, we agree with the importance of tacit and group knowledge on design, but we do not investigate engineering design from the position of social psychology or ethnography. Perspective of human-computer interaction (HCI) research would perhaps be more focused on the actual interaction between the designers and computational tools, their intuitiveness or just-in-time action (Ezekiel 2000; Muller and Carey 2002). Obviously, many methods and methodologies used by HCI researchers or ethnographers can be used as input to this work, thus broadening our understanding of a particular field. For instance, in order to validate the theory of iterative framing in design we have been inspired by the existing field studies performed by the ethnographers (Rose, Shneiderman *et al.* 1995). We conducted our study in accordance with the principles of the participatory design, as described e.g. in (ACM 1993; Miller 1993; Muller 2002). This methodological inspiration helped to verify our theory of reflective and iterative design featuring (re-)framing. For instance, we looked at the following points from the participatory design when designing and conducting the validating experiments:

- Users are best qualified to determine how to improve/amend their work;
- Users' perception of a tool is crucial to its acceptance;
- Opinions from users with different levels of expertise (expert/novice) are equally valid.

Nevertheless, the main emphasis of the work performed during the author's PhD remains on the reasoning strategies designers typically use when tackling a vaguely defined problem, and the dependencies among them. The ultimate objective of this work is to achieve a better understanding of the nature of the conceptual phase of non-trivial engineering design projects in a narrow sense, or perhaps of design in a broader one. This objective informed also the preparation and conduct of the field study. The main aim of the initial experimental work was to confirm or refute the assumption based upon the review of the literature that the designers frame their problems and reflect on their understanding of the problem. In addition, we wanted to see how the designers work on two simultaneous fronts – generation of a plausible design solution and a further development of problem specification thus featuring a co-evolving understanding of a design problem.

We expected a reflection, i.e. oscillation between the design solution and design specification, preference given to the amendment of the existing solutions in contrast to a common backtracking approach. We also expected that designers would apply different reasoning strategies to suit the particular needs they may have at any given moment of designing. In order to achieve this objective, a tailored design toolkit prototype was developed and deployed in the experiments. The use of a specialised toolkit typically influences the results of the observation; therefore we were also interested to learn more about the users' perception of such a tool, and its potential with regard to the knowledge-intensive design support. In particular this part of our research is approaching the focus of the complementary research done in the domain of design ethnography, participatory design or HCI. However, we only touch it superficially, use it as a methodological foundation, but do not contribute to those areas. In other words, we deployed the existing methodologies and summarised the empirical findings and results as recommended for example, in (Rose, Shneiderman *et al.* 1995).

We emphasise that there is no intention to present this thesis as a contribution to any aspect of the theory of participatory or ethnographic design. Nonetheless, in some future projects, it may be interesting and beneficial to analyse the empirical data we gathered from the perspectives of the related research areas. The research fields highlighted in this section are complementary to this thesis.

3.4.3 A perspective of a Software Engineer

In this chapter, we touched the complexity of design, especially its conceptual phase, in which a problem is formulated. We saw accounts of how the problem conceptualisation affects the overall design process and its product (Candy and Edmonds 1996; Jackson 2001; Schön 1983; Ullman 1997). One domain where this issue is well elaborated is software engineering. Software engineers unlike architects or car designers really seem to work in two distinct worlds. According to Jackson (2001), in software design the problems are in the real, physical world but their solutions are inside computers. Software is much more abstract and lexical domain than any car parts or building structures. The discrepancy between the world of problems and solutions is better exposed in software engineering.

Jackson (2001) and Fischer (1992) note that it is very easy to tackle a software development problem as a problem of various interfaces and data structures that reside within a computer. However, the essence of a software design problem is in the real world; whatever resides in a computer tends to be a solution to this problem of a broader world. A problem must be understood and clarified before embarking on its solution; it must be put into a context. To that extent, Jackson (2001) emphasises the importance of context diagrams for choosing the relevant domains involved in the problem, how do they relate to the machine, and what interfaces would be shared by the domains and the machines.

As a formal representation of a context diagram, Jackson introduces a notion of *problem frame*[3]. He defines a problem frame as a familiar sub-problem class that can be used for the decomposition of a larger and vaguely specified problem. He also notes that the decomposition must not necessarily be hierarchical; it may be parallel and concurrent. Problem frames are somewhat similar to the design patterns or prototypes of Gero (1990), see also section 3.4.1. However, unlike design prototypes that focus on the solutions, Jackson's problem frames are applied to the problems and specifications.

A problem frame acts as a kind of pattern that enables a designer to define his or her software design problem in terms of intuitively identifiable problem class. In other words, a designer may define various roles, characteristics, interfaces and *typical* requirements based on applying a perspective of a particular problem frame on his or her design problem. Each such problem frame is a projection of a real-world problem onto a potentially applicable 'software problem'. Such projection suggests a typical concern of a given class of problems and also suggests a suitable solution to the concern.

[3] Interestingly, in 'problem frames' Jackson uses a similar concept as Schön (1983) to describe an operation that precedes and complements a process of problem solving.

An example of a problem frame is given in Figure 3–4, and shows a scheme for commanded behaviour problem. In this case, software resides in a machine called 'sluice controller', which interacts with two physical domains – the actual 'gate' and 'human operator'. The desired commanded behaviour is also explicitly represented in the frame in terms of 'raising and lowering the gate'. The relationships between different roles in the system correspond to different kinds of data that can be shared between different nodes. For instance, the operator may issue a command 'Raise gate' to the controller and it should be understood. Thus, this command is a shared piece of data between an operator and a gate controller. However, this command would make little sense if it was shared between the motor of the gate and the controller. What needs to be shared between the controller and the motor is a different command – for instance, impulses 'Motor ON' and 'Motor Anti-clockwise', which would be understood by the motor.

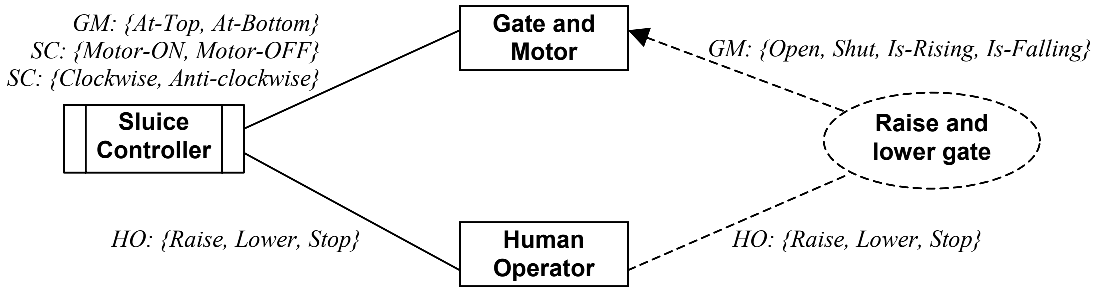


Figure 3–4. Design problem translated into a specific problem frame (Jackson 2001)

This simple problem frame shows the complexity of trying to make sense of a real-world design problem. Already in this primitive form, it brings a designer's attention to the different types of data to be used, different information channels to be designed, but also viability and/or sensibility of certain combinations of different data types. Later in the design, the 'Motor ON' would result into a machine-friendly electronic impulse of given amplitude. On the other hand, 'Raise gate' command may be implemented as a two-position lever on a control panel. Although this early scheme says nothing about the implementation, it makes explicit the difference between these two pieces of data.

Problem frames are an interesting and rather rare attempt in the design community to tackle the problems that are also a topic of this thesis formally. However, because of the vagueness of the real world, problem frames can come in many flavours (Jackson 2001). We therefore agree with Jackson's remark that there cannot be a universally valid single description or specification of any problem. All specifications are made with a particular purpose and in a particular context. They are thus very fluid and subject to minor or more radical amendment in response to the actual solution and the 'real problem.

After setting up the stage and briefly discussed the essential terms we are focusing on, we move to more specific parts of the thesis. Let us begin with the review of the related literature with regard to the foci and objectives of our research. The next chapter (number 4) reviews several theoretical positions on design. This is followed by chapter 5 that focuses on the knowledge-level aspects and various knowledge-level models of design.

4 PERSPECTIVES ON REASONING IN DESIGN

The purpose of this and the following chapters is to review the existing perspectives and research on design. First, we take a broad view and present different perspectives and angles for the investigation of design. The broad review in this chapter is complemented by a more focused perspective on knowledge-level modelling and application of knowledge-based tools in design support, presented in chapter 5.

In this chapter, we first introduce design from a number of perspectives. This introduction helps to articulate and ground the basic assumptions of the recursive frame-based model of design later in this document. In the subsequent sections, we show that design is ill structured and ambiguous, from various perspectives. The structure of design problems is typically very loose in respect to the reasoning strategies that may be deployed in the construction of a problem-solving space and efficient navigation within it. Different authors pay attention to different types of reasoning that may be observed in a design process. A concise summary of the reasoning methods applicable to design is presented, for instance in (Chandrasekaran 1990). Section 4.1 is a review of the research on design as various scientific disciplines and researchers see it. It covers artificial intelligence and knowledge modelling perspectives, technological issues as seen by design practitioners, and ends with cognitive aspects and a debate on features such as innovation and creativity.

A separate section 4.2 is devoted to the introduction of two paradigms underlying this thesis. We review the existing research on the co-evolving specifications and solutions, and introduce the concepts of 'reflection' and 'shaping the design situation'. The research presented in this thesis directly extends the research on reflection in design. It is shown in section 4.2 that there is a intimate relation between the research on handling the incompleteness and vagueness of the real world and the reflective (re-) shaping of the designer's understanding of design situations.

The review of existing approaches and perspectives is concluded by a summary that relates these different points of view to the one chosen as a basis for this thesis. Section 4.3 proposes basic features that are typically associated with all design problems regardless of their complexity. The list of 'applicable' features is proposed as the author's interpretation of the research reviewed in section 4.1. This list brings to the reader's attention the gaps in the existing perception of the science of design, which are likely to be an interesting subject for future development. It is argued at the end of this overview that most of the existing accounts reviewed address and model design on an explicit level. At the same time, we believe that there is a serious lack of research on modelling inarticulate reasoning processes, which are closely investigated in the subsequent chapters.

4.1 Design in multiple perspectives

Chandrasekaran (1990), but also Simon (1969), Coyne, Rosenman *et al.* (1990), and others, often approach design as a problem of searching through large spaces of potential building blocks (typically structural elements). On the other hand, Oxman (1990), Smithers, Conkie *et al.* (1990), Gomez de Silva Garza and Maher (1996), and others, emphasise the exploratory nature of design. These researchers claim that design is more an exploration of a problem solving space rather than search in such a space.

This belief is supported by the observation that in addition to searching through a complex space, a significant aspect of design is also the construction of a precise structure of the space from initial, vague desires. These two basic approaches look at the design tasks from a *problem solving perspective*, and are discussed in section 4.1.1.

When we look at the methods or classes of methods, which can potentially be distinguished in the different phases of a design process, we can also come up with multiple definitions of what actually constitutes design. By putting emphasis on the operational processes, design could be roughly divided into three consecutive 'operations' (Coyne, Rosenman *et al.* 1990) that can often be decomposed further into finer-grained activities (Ullman 1997). According to this school of thought, design usually begins with a task analysis, and continues with the synthesis of an artefact. Generally, it ends with evaluation of a proposed artefact in respect to its acceptability. This coarse-grained sequence of operations can be formed into rather complex chains, loops and hierarchies. More about this *operational view* on design can be found in section 4.1.2.

Many researchers look at the design as a challenging *cognitive activity* (Gero 1990; Candy and Edmonds 1996; Cross 1997). They perceive design as a sequence of certain 'primitive' cognitive operations, whose main purpose is to strengthen a designer's insight into the problem at hand. The repetitive and possibly iterative sequence of these 'primitive operations' then constructs the complex activity of designing. In section 4.1.3, we look at some cognitive terms, such as initial divergence, followed by various result transformations and a convergence toward a final design solution. Moreover, we mention a few of the approaches drawing upon cognitive science; e.g. design patterns and prototypes (Gero 1990), or memory-based re-use (Gomez de Silva Garza and Maher 1996; Watson and Perera 1997) – both popular especially in architectural design.

An important and favourite cognitive phenomenon studied by numerous researchers is creativity and innovation in the design (Altshuller 1984; Gero 1990; Goel 1997; Thompson and Lordan 1999). Since creativity is often cited as one of the typical features of non-trivial, real-world designs, we devote a separate section to this phenomenon. We look at two contradictory *views on creativity*, and mention the implications they have on modelling design formulation, solution discovery, and design support. In section 4.1.4, we give space to one of the attempts to support human creativity using an algorithm that is governed by objective rules similar to those of physics or mathematics; TRIZ – a theory of Russian researcher and design theorist Altshuller (1984).

4.1.1 Search vs. exploration in design

Chandrasekaran (1990) defines design as a search problem in a large space for objects that satisfy multiple constraints (see the definitions in section 3.2.2). Only a small number of objects in such a large space constitute satisfactory solutions, and an even smaller number the optimal solutions. He argues that what is actually needed in design is some knowledge that radically shrinks the search space, or significantly accelerates the navigation through it. Such knowledge could often be expressed as a mapping from a set of available primitive components (structures) onto the states that the designer wishes to achieve or avoid. Bylander and Chandrasekaran (1985) were among the first researchers who proposed that the desirable design states could be expressed as *desired functions* of the designed artefact. Chandrasekaran (1993) later enhanced this view, and suggested a clear and objective ontology of relationships between the (available) structures and (desired) functions. The idea of well-defined links between certain structural elements and appropriate functions occurs also in the research of Qian and Gero (1996) and Iwasaki, Fikes *et al.* (1993). Even the RFD theory we present later is built with this view in mind – see the basic assumptions introduced in section 1.1.

In general, all these researchers construct causal links between the primitive structures and the deliverable functions. The researchers assume that the desired functions are a part of the problem specifications, and since the chains between the functions and primitive structures are well defined in a particular domain, they may be used to accelerate the search. Formally, they are talking about two strategies of search. First, *backward chaining* or abduction when they are looking for a structure satisfying a function. Second, *forward chaining* or deduction when they are interested in the functionality of a particular structure. They typically reserve the term 'design' for problems, in which the former searching strategy takes place; i.e. how to find a structure that would deliver the desired functionality. The latter strategy is often referred to as qualitative modelling or prediction of behaviours from structure. This distinction is clearly observable in the papers from two different groups of Stanford researchers. Gruber and Iwasaki (1991) deploy the causal chains between structures and functions to analyse 'how things work and <u>what they do</u>'; i.e. they look at what functionality is implied by the given structure. Whereas Iwasaki, Fikes *et al.* (1993) emphasise synthetic features, and look at '<u>how</u> things <u>are intended</u> to work'; i.e. how to design things that would satisfy the designer's intentions regarding functionality. Iwasaki *et al.* understand design as an activity covering the synthesis of an artefact. Analysis in their approach typically plays an important role in the validation and justification of the designed artefacts and devices.

On the other hand, there is a large group of researchers who argue that analysis and synthesis are mutually very closely related, and design is seen as a result of their interplay. The key argument of this school of thought is that design cannot be treated as pure deployment of the existing links between various elements. For instance, Oxman (1990) emphasises the active role that designers typically play in the design. They not only use the information and data they have at their disposal, but also '*actively manipulate them utilising a higher level knowledge and advanced control mechanisms*'. Logan and Smithers (1992) make a more radical point arguing that the distinction between analysis and synthesis is fundamentally meaningless. They claim that the formulation of a design problem at any stage is not final. As the design goes further, new knowledge about applicable primitive structures may become apparent. New knowledge may help to reveal the inconsistencies in the existing specification of the problem, and eventually lead to a new understanding of the problem at hand.

Smithers, Conkie *et al.* (1990) compare the situation, in which designers work, to that of '*adventurers or explorers who just landed at a new island and have very limited knowledge of its geography*'. Initially, they may assume some features of the island but in order to make a precise and valid geographical report, they eventually need to go out, and explore the environment of the island. Every exploratory journey may focus on a slightly different feature of the island, and as such may bring new data and new knowledge to shape their original impressions and/or expectations in respect to the geography of the island.

A general argument advanced by the work of the ‘design-as-exploration’ school is that search is *passive* – i.e. only the existing knowledge is deployed, nothing new is ever uncovered (or discovered). On the contrary, exploration emphasises the *activity* and the development of existing knowledge. In search, there may be some heuristics available to move through vast design spaces quickly and efficiently (Chandrasekaran 1990), but most of the heuristics assume some ‘a priori’ knowledge. When such knowledge is not available, the searching mechanism is typically not able to generate or otherwise acquire it on the fly – in this sense it is passive. Exploration on the other hand, enhances the search with some active aspects. There may be available some ‘a priori’ knowledge to ‘seed’ the exploration, but this knowledge may be constantly re-shaped and re-validated as the design progresses and new solutions or relations become apparent.

Understanding design as a problem of exploration seems to be more realistic from the epistemological perspective, too (see introduction to this issue in section 3.2.4). Cook and Brown (1999) but also Schön (1983; 1995) report the importance of (human) activity in designing artefacts and learning. They even distinguish formally between ‘knowledge’ as something that is possessed by a subject and ‘knowing’ as something that is actively practised. They argue that knowledge is inherently passive, and may acquire its full potential only when applied in an action. Only in such a manner something new can be obtained, produced, or generated out of the ‘possessed resources’. Using their vocabulary, the search involves only the possession of knowledge, and navigation through it. Whereas the exploration involves an action, in which particular knowledge can be validated and refined.

4.1.2 Operationalisation of design process

In the previous section, we briefly introduced the epistemological discrepancy between two different approaches to design. One addressed design as a problem of pure search for the right structure, and it was contrasted to an approach seeing design as interplay between searching for structures and the exploration of the consequences. Researchers from an engineering background and many design practitioners also attempt to split the complex design process into smaller chunks – often called primitive operations or processes. Asimow (1962) outlined three major operations or tasks that are typically performed during the design of the technological artefacts as: *Analysis*, *Synthesis* and *Evaluation*.

In the first task – analysis, the designer attempts to understand the problem, and come up with an explicit specification of design goals. The identified goals are used in the subsequent stage – synthesis, where solutions are sought that would be plausible in respect to the analysed goals. Finally, in the last operation – evaluation, the validity of synthesised solutions is assessed, the optimal alternatives are selected and/or some implications are drawn about how the goals can be further refined. A cycle may be achieved when the evaluation yields a need to re-examine the goals, and triggers again the analysis of the design problem. Repeated analysis has consequently an impact on the current solutions that may need to be re-built – or *re-engineered*. This model of a design process is often associated with the adjective ‘waterfall’. It assumes that the activity at any stage may begin only when the previous stage has been successfully accomplished. However, when we start looking at the details of the individual phases we find that the assumption of clearly divided stages is rather idealistic. One of the representations of a waterfall model of a design process is depicted in Figure 4–1 below.

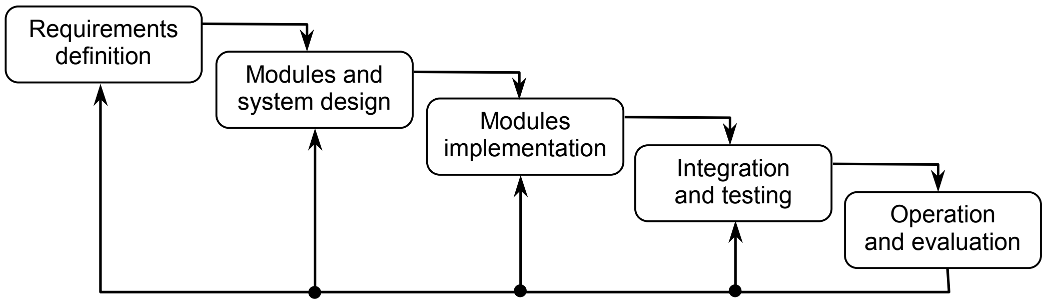


Figure 4–1. Sample representation of waterfall model

Because of the too-strict and rigid distinction of the individual operations in the waterfall models, many researchers came up with more sophisticated breakdowns of a design process. For instance, Ullman (1997) but also Nidamarthi, Chakrabarti *et al.* (1997) distinguish roughly the following sequence of operations usually observable in the designs of technological artefacts:

Planning →

→ Development of engineering specifications →

→ Development of conceptual solutions →

→ Implementation of solutions and product refinement →

→ Evaluation of the implemented product(s)

Unlike the waterfall model that featured a single-direction loop, there are multiple 'feedbacks' in these more detailed breakdowns. A designer may interrupt the current flow of the design and return, for example, to the planning phase once the initial problem specification was explicated. Alternatively, after proposing a conceptual solution, the original specification may be amended to extend or shrink the number of conceptual solutions. Obviously, evaluation may return the entire design to any of the previous phases, not necessarily to the beginning, as it was the case with the waterfall model.

The major problem with such an operational approach to design is very similar to the distinction between the analysis and synthesis, as mentioned in the previous section. While the searching paradigm assumes a clear borderline between the two sub-tasks of the design, the exploratory paradigm is more modest about such clear separations. The same issue arises also in the attempt to divide the design phases into distinctive operations that are clearly related and ordered with regard to their causality or temporality. In general, there is only a small group of design problems, to which such simplified operational models fully apply. The vast majority of problems are less transparent, and rather than a sequence of some primitive operations, their mutual interplay is idiomatic.

Obviously, the operational school of thought brings in several benefits. If such a methodology is put into practice in a real-life company, then the process of documentation and support for the evaluation of the designed products is typically more straightforward and more transparent. However, we have to note that the emphasis on clearer documentation by a strict distinction between the different phases of the design may significantly compromise innovation and the appearance of novel ideas in general. Ullman (1997) formulates an important warning when he argues that a lot of current design practice has '*over-the-wall*' character. He illustrates the situation with the fact that different teams of professionals involved in a product design may work mutually independently, and communicate with each other only through a limited channel. As if they were working inside a walled yard, and all their communication with the external world consisted of occasionally throwing small scraps of information over the wall and catching the response from outside in a similar manner.

Although such isolation may be useful in certain specific cases, it is not a typical environment for humans to live and work in. These issues were also investigated from a psychological perspective, and are reported, for instance, in (Reddy 1988). Reddy's observations are basically in accordance with those made by Fischer (1992) who argues that whilst some level of operationalisation might be a useful first approach to a complex issue, there remains a danger of '*re-inventing the wheel*'. When a strict and unnatural division of activities is imposed on designers, as most of the waterfall operational models do, it can be observed that these models are oversimplified skeletons of a real design process. Such oversimplification may lead the designers to committing the same failures repeatedly. Often, it may produce products that comply with the forced explicit analysis but have very little to do with the real needs and expectations of the customers.

We already mentioned that there were numerous attempts to improve the performance of the waterfall models. One such improvement was presented as a spiral model of an entire product life-cycle. However, as e.g. Ullman (1997) notes, the spiralling models exhibit serious gaps in respect to dividing 'operations' that are typically very closely related. To rectify this gap, Nuseibeh (2001) presented a 'Twin Peaks' model, which spiralled between two dual 'worlds'. He developed the model in the domain of requirements engineering and software design. As he argues, the Twin Peaks model intertwines the requirements with the architectures in order to achieve incremental delivery of a product. This model is one of few formal attempts to ascribe equal importance to the development of requirements ('problem analysis') and the development of solutions to these requirements (i.e. 'product synthesis'). Since this 'inevitable intertwining of problem specifications and solutions' (Swartout and Balzer 1982) is at the heart of this thesis, the Twin Peak model is depicted in Figure 4–2 to illustrate the concept of mutual interplay and interaction.

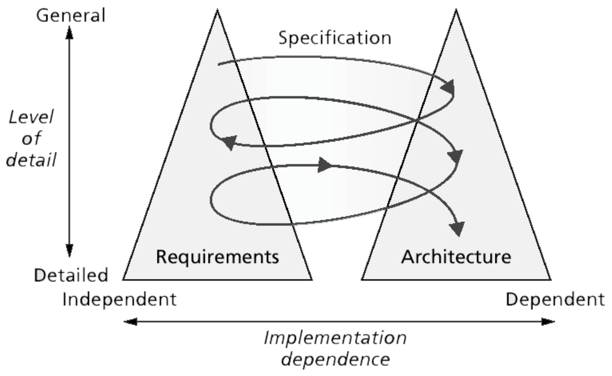


Figure 4–2. Twin Peak model featuring concurrent operations – development in the world of problem requirements and in the world of products/solutions (Nuseibeh 2001)

4.1.3 Cognitive aspects of design

Researchers working with the cognitive aspects of design also attempt to break down the process of designing into smaller cognitive units. Numerous studies have been conducted in order to bring some light from this perspective, and in general, they share the opinion that the following cognitive processes are observable in the different stages of a design process. Designers typically approach their problem through a *divergent strategy*, whose purpose is to prepare a certain number of alternatives. Then they go through a *transformation phase* that puts all the bits together in a tried and tested or possibly a novel fashion. Finally, they *converge*, and focus on the selected issues, work out their details, and usually implement them in an appropriate way.

Addressing divergent thinking, Candy and Edmonds (1996) emphasise the need to keep several possible design channels open in parallel. This need is particularly strong when the aim is to produce innovative or creative designs. The authors also argue that there is no single pattern of generating design ideas. On the contrary, multiple perspectives often enrich and broaden the knowledge a designer may have about a particular domain. The initial divergence may seem harmful and distracting to many designers but novel ideas do not usually appear as a bolt from the blue sky. Even if the insight is sudden, a designer or an inventor must be prepared for it. The divergent understanding of a problem at hand from several complementary perspectives, enables designers to understand the existing approaches in similar and different domains. This divergence however, is not self-purposeful. It helps the designers to discover interesting analogies with designs in other areas, and ultimately transform their own design space as well as their approach to a given design problem.

Cross (1997; 1999) reports a similar causality between divergent thinking at the beginning, the ability to transform the existing artefacts into novel ideas, and the convergence that elaborates a transformed conceptual solution into a final product. Like Candy and Edmonds, Cross presents the observations that include the exploration of the problem space and the generation of more common or familiar solutions as key phases of design. Often, a novel idea comes to life during the convergence and transformation phases. The introduction of such innovation may seem to be unexpected and abrupt. However, after a careful analysis, Cross argues that often, even a very innovative concept can be traced back to the early phases of design through several variations and transformations. Consequently, such novelty has its roots in less usual transformations of the existing, trivial or more familiar approaches.

He calls this sudden appearance of an idea a *creative leap* rather than referring to some ‘mystical’ enlightenment. The essence of a creative leap is to construct a bridge between familiar, more common solutions and unusual innovations. An important assumption for a successful creative leap or bridge is that a designer has to be firmly standing among the familiar approaches. Just as jumpers do not jump without a previous warm-up, the designers cannot wait for a novel idea to fall from the sky. According to Cross, their ‘warm-up’ may consist of several transformational techniques that make the designers aware of different analogies and similarities. A typical example of a transformational technique that is often applied in the everyday life, is a *combination* of existing approaches. Cross illustrates this with an example involving a bicycle rack, in which the designers first investigate a concept of ‘panel’ to serve as a rack. An alternative, completely unrelated approach looked at a rack as a kind of ‘bag’. Unfortunately, both suggested alternatives had certain strengths and certain weaknesses. However, the designers ‘combined’ them and came up with an idea of something that was as sturdy as a panel but also had side walls like a bag. See sketch in Figure 4–3 for a schematic ‘combination’ of the familiar solutions. An idea that sprung from a rather strange and physically unfeasible combination of a 'panelled bag' led step-by-step to a new concept of a 'tray'. This novel concept exhibited the desirable features of the more common alternatives, but also it was a compromise between their negative features. Nevertheless, it had a potential to serve as a basis for the detailed development of a bicycle rack.

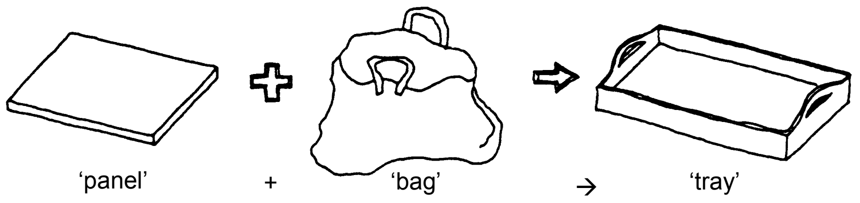


Figure 4–3. Combination of early design ideas into a novel concept (Cross 1997)

Another frequently mentioned cognitive activity that is observable in the design is the *emergence* of new design aspects or features. Usually, a property is understood as emergent when it 'comes' with the rest of the explicit (design) solution; it is implicitly observable but such an observation typically requires a higher-level perception and imagination (Gero 1996). A familiar example illustrating the emergence taken from non-design cognitive studies is shown in Figure 4–4. Depending on the way we look at the picture, we may see a white goblet in the middle or two black faces in profile turned towards each other. Emergence may usually manifest itself as unexpected features exhibited by a designed product. Although emergence may play an important role in triggering the appearance of novel ideas, the emergent features may be also harmful.

An example of emergent property from a design domain involves a fast-activating airbag. Without any doubt, the activation speed is a very desirable feature designed in order to protect a driver from the crash injuries. However, in certain cases, the same fast-expanding airbag may inflict worse injuries to the driver than the impact itself. It may break his or her neck or cause a concussion of the brain. These are not features that would have been required or desired by any customer. These (negative) features simply 'emerged' with a designer's decision to follow a particular solution, strategy, implementation procedure or a range of design parameters.

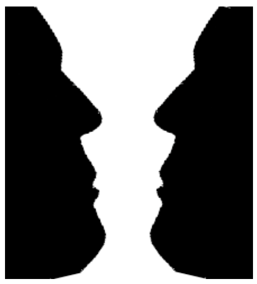


Figure 4–4. Example of an emergent feature – shape of the object(s)

In the previous paragraphs, we referred several times to 'familiar approaches' or 'more usual solutions' that can be transformed into innovative ones. What is the source of these familiar or usual solutions? One source investigated by John Gero (1990) in the context of architectural design consists of various *design prototypes*. As Gero claims, a typical feature of architectural design is the existence of certain *patterns* that describe the essential attributes of the buildings but leave a lot of freedom for innovation and/or extension. According to Gomez de Silva Garza and Maher (1996), design patterns and prototypes belong among memory-based design techniques. Another representative of such cognitively inspired approaches to design is *case-based design*, *CBD* (Watson and Perera 1997).

A typical application of the memory-based techniques begins with articulation of a desirable pattern that addresses the design problem – e.g. 'we want to design an office building' (Gomez de Silva Garza and Maher 1996). The next step typically involves retrieval of known prototypes or previous design cases that comply with the articulated pattern. The retrieved – familiar approaches may exhibit different granularity, applicability or similarity. Nevertheless, one or more of these familiar concepts are taken and the designer attempts to adapt them to his or her problem. The adaptation may involve the refinement of an applicable prototype (e.g. an office building with a rectangular geometry, open-space layout and space area larger than 2000m^2). Alternatively, it may involve a combination of several familiar solutions (as in Cross's example involving a bicycle rack).

The different memory-based approaches to design differ mainly in the computational details of how the individual operations or cognitive activities are represented and modelled. Another point distinguishing them depends on the source of familiar models or solutions. Designers may use a repository of specific design cases tackled in the past – this is a paradigm underlying CBD. They may refer to a repository of generalised prototypes or models as in prototype-based design. Gero even distinguishes several types of models – prototypes being one such type. First, he defines a *stereotype* as a model that can be copied without change – as e.g. in the off-the-shelf or mass production. Then, *archetypes* are first or singular instances of a particular type/model – e.g. Rolls-Royce is an archetypal car of royalty. Finally, *prototypes* are generalised representations of the groupings of certain elements from similar design cases (see the above-mentioned 'prototypic office building').

The familiar sources and design methods can be applied rigidly or flexibly. Gero (1990) and partly Chandrasekaran (1990) characterise several classes of design problems according to the ratio between rigidity and flexibility of knowledge re-use. Many design problems are *routine* or well defined. They exhibit a complete and strictly structured design space, all necessary parameters, values or means of deployment are well known in advance. Typically, design is concerned with the instantiation of the parameters and the production of a final product. Another category involves *innovative problems*. Innovation may take place when a known and familiar structure acquires a novel mode of deployment; i.e., the design moves out of the well-defined tracks and the results are usually not guaranteed, as was the case with routine problems. Typically, innovative designs use the existing parameters and assign them new or less common values.

The final category consists of *creative problems*. These problems exhibit minor or more significant extensions to the familiar, well-defined design spaces. Typically, such extensions cannot be achieved only by a novel deployment of known structures and parameters. They require also the articulation of new structures, relationships or paradigms. Products resulting from such creative extensions are highly appreciated by society and are often referred to as *inventions* (Altshuller 1984).

4.1.4 Is creativity predictable?

The previous section concluded with the introduction of two cognitive terms often heard in relation to design – *creativity* and invention. These are terms used widely in our everyday lives, and as such are charged with many misconceptions. There are two schools of thought when it comes to explaining the aspect of creativity. Watson and Perera (1997) but also Steinberg (1994) point out that creativity in design can be much more a social issue when the ultimate judge of the creativity of a product is the society that is using it. Another community considers creativity to be a result of a certain processes and problem solving techniques (Altshuller 1984; Thompson and Lordan 1999; Liu 2000).

There are many questions left open by both communities. Can a creative design process be repeated and still produce creative results? If a creative process is re-usable, are its modifications also considered creative? The counter-argument to these questions claims that it is the deployment of a product in a particular domain that may be judged innovative or creative. The particular solution may be known in other domains, but it is its unusual deployment that gives it the adjective 'creative'.

Thus, creativity may be associated with both the product of any design method deployed in certain environment, as well as the specific design procedure. Both views have many supporters and opponents. This viewpoint, seeing the product and its deployment as a measure of creativity, has support, for instance, in the community of prototype-based (Gero 1990) or case-based design (Gomez de Silva Garza and Maher 1996; Prabhakar and Goel 1998). These researchers argue that a sufficiently large repository of previous design cases (products) may trigger the awareness of interesting and unexpected analogies or innovative explanations of puzzling features.

Watson and Perera (1997) compiled a list of selected aspects of the case-based paradigm to demonstrate its suitability for creative designs and a close relationship between creativity and the use of previous design cases. They mention several features of various case-based design systems as relevant to creativity. To illustrate their understanding, some of their observations are highlighted in this section. According to the review, a case-based approach to design provides the designers with useful insights as to how components were combined in a previous, familiar case in order to deliver the desired performance of a product. Moreover, reminders of the previous instances of a particular type of design, may improve the current designs by enhancing the creativity of the designers.

However, as Dominowski and Dallob (1995) note, the existence of a familiar representation, whilst solving a problem, may have also an adverse effect on the designer's creativity. The familiarity may help to clarify the outstanding issues but it may also cause the designer to become fixated on the familiar interpretation. Such a situation typically occurs when a designer tries to re-use the existing approach literally; he tries to conform to the direction set out by the familiar case. Thus, the reference to a familiar design case experienced in the past may inhibit and suppress creative insight, as well as enhance it. From this observation, we may conclude that the presence of the previous design cases is, in general, a helpful but not a sufficient condition for creativity. To this point, Watson and Perera (1997) add that rather than the actual design cases, the process of their re-use is more important. The way, in which the designer deploys the previous cases, influences whether a memory-inspired approach would enhance or inhibit creative element.

There are many models of a case-based approach to designing products. However, there are only a few generic algorithms acceptable as proven methods for designing creative and innovative products. This may suggest that the view of creativity focusing on the design process (algorithm) that yields a creative outcome is less popular in the design community. However, this is not true. For instance, Altshuller (1984) developed a theory of inventive problem solving by generalising over a large number of cases that were accepted as innovative patents or inventions worldwide. One of the outcomes of his theory (called TRIZ or TIPS) is an algorithm for improving the chance of coming up with an inventive solution to a given problem.

TRIZ is a very interesting theory because it focuses an inventive design task into a small core of basic procedures. First, an explicit contradiction is identified in the specification of the desired product behaviour. Next, the core of invention is in the removal of such a contradiction. Altshuller understands the contradictions as mutually connected features that are typically observable in a designed artefact, and the improvement of any one of them worsens the other one. So far, TRIZ reminds us of the decisions as described by numerous other researchers (Candy and Edmonds 1996; Cross 1997). However, the main contribution of TRIZ is that Altshuller generalised many possible contradictions to a matrix of typically contradictory classes of physical properties. Thus, the designer may associate a specific identified contradiction from his problem with one or more patterns in such a matrix.

In addition to assisting with the identification of conflicts, Altshuller also proposed a table of methods and techniques for the removal of the contradictions. He called these methods *inventive principles*, and most of them were generalised from the previous inventive solutions. Thus, the main principle of TRIZ is that it assigns one or more applicable fixes (i.e. inventive principles) to the feasible and typical contradictions between the physical quantities and behaviours. Nonetheless, it must be noted that the theory has been developed for a particular domain of designing technological products. Thus, when talking about 'typical contradictions' TRIZ refers to contradictions that are typically observable in the domain of engineering design – e.g. if a particular component should exhibit properties of both conductors and insulators or be both present and absent.

Formally, Altshuller's theory follows the same steps as described by Candy and Edmonds (1996) or Cross (1997). It starts with the identification of the problem (physical contradiction). After finding an explicit contradiction, a set of applicable inventive principles is retrieved from the inventive matrix. The whole procedure ends with an application of the selected inventive principle and its adaptation for the current problem. TRIZ supports both the retrieval and the application of the inventive principle. Indirectly, by providing usual sources of contradictions, it also supports the recognition.

The 'operation' of retrieving applicable inventive principles is supported by strong empirical evidence gathered from thousands of inventive cases submitted throughout the world. The evidence is presented in a succinct form as a matrix or a two-dimensional table. In the terminology of case-based design, we may say that the inventive matrix provides a referential index for the recognition of certain patterns in the problems tackled. For example, if there is a conflict between improving the temperature and precision of manufacturing, TRIZ suggests 'creating or stabilising a required structure' (see below for an example).

The latter 'operation' featuring adaptation of the inventive principle for the current problem is supported by examples. A designer may be shown how other applications and implementations of the same inventive principle were designed, what structures or relations they contained, or what they looked like. However, there is very limited help provided with the adaptation or transformation of rather abstract inventive principles for the particular problem or domain. Consider the following scenario describing a solution of a 'typical' inventive problem from (Altshuller 1984):

Problem: "*A lens is to be polished to a high precision and accuracy. For this purpose, a coolant needs to be delivered between the polished surface and polisher (made of resin). An attempt was made to incorporate slits into the polishing assembly where water could be squirted occasionally. However, the perforated surface of a polisher performed poorly in respect to the precision…*"

Contradiction (verbally): "*The cooling capability of a perforated polisher conflicts with its ability to polish glass. A 'whole' polisher cannot cool sufficiently, a perforated one does not deliver good results. Thus, a polisher must be both 'perforated' and 'solid' at the same time…*"

Contradiction pattern: "*This is a conflict between a structure exhibiting desired behaviour in respect to temperature and the accuracy of manufacturing.*"

Inventive principle: "*Create a structure with a desired property (temperature) from the existing one.*"

Examples of the principle: "*Use interference waves, standing waves, magnetic waves. Alternatively, apply phase transitions, mechanical or acoustic oscillations. Finally, consider cavitation.*"

Inventive solution 1: "*Make the polisher from a micro-porous material; thus it contains microscopic holes for pouring a coolant and at the same time is 'solid' on a macroscopic level. The 'conflict' is no more – a porous polisher both is and is not 'solid'…*" → implements cavitation

Inventive solution 2: "*Make the polisher from abrasive particles frozen into ice. As the lens is polished, the ice starts melting, thus delivering water (a coolant) onto the surface of lens. The polisher is 'solid' but simultaneously, water can pass 'through' it…*" → implements phase transition (ice ⇔ water)

Nevertheless, as Altshuller emphasises in his book, it is up to the designer to find a suitable implementation of a particular inventive principle for his problem. Thus, the process of recognising and 'exaggerating' the contradiction is clearly structured and supported empirically. The next step is left entirely with the designer; he or she must recognise what can be re-used from the different examples and how it can be adapted and implemented. However, the search for the 'right' implementation is not blind and uninformed anymore. Already, there is a pattern (model) that needs to be instantiated.

Thus, an exaggeration of a problem specification into a contradictory pair of parameters is instrumental in avoiding the accidental exploration of all possible analogies between the previous cases and the current problem. Instead, a consultation to a ready-made, empirically supported matrix offers a conceptual solution or a model that removes the contradiction and may bring in novelty as well.

As we mentioned above, the TRIZ community (Altshuller 1984; Sushkov, Mars *et al.* 1995) developed a few techniques for recognising conflicts, and TRIZ itself helps to construct a conceptual solution. However, we believe that the process of recognising such contradictions is not elaborated sufficiently by this community. The recognition process may involve several iterative steps refining the designer's perception of the problem, and addressing the issue from different perspectives. Another gap that is worthwhile addressing features the fact that unlike in the example above, the contradictions may be sometimes very hard to articulate in such clear and explicit terms. Thus, a designer or inventor may spend a significant time trying to elicit the essence of a tacitly perceived conflict in various explicit categories before arriving at the 'right' one that yields the 'right' (i.e. *acceptable*) result.

To conclude this section, TRIZ (and related research; see also section 5.1.4) has many interesting features that should be contained in a broader framework of design problem solving. As we show later in the document, Altshuller's theory can be incorporated in the proposed recursive theory of design framing. It may be perceived as one of several mechanisms that implement reflection on design actions. Nonetheless, we re-iterate that TRIZ was developed for technological systems and explicit physical contradictions. Thus, it is not directly applicable outside engineering design.

4.2 Incompleteness, evolution and other relevant research

In this section, we present a few interesting and closely related views on problem solving in design that deserve a separate section. This is mainly because they form the essential foundations of the claims proposed in further chapters. First, we review a few approaches to handling the incompleteness and vagueness of the design problems in section 4.2.1. Next in section 4.2.2, we introduce the terminology of reflection and framing on a relatively abstract level.

4.2.1 Handling ill-structured problems

A lot of research addressed the poor structure, vagueness and ambiguity of real-world problems in general – design being no exception. This research was accelerated in the last century by the arrival of computers and the desire to employ the computers as problem solvers. The interesting category and not entirely beaten challenge involves the problems of everyday life with all the vagueness and ambiguities of the environment, in which they are typically tackled by humans (Newell and Simon 1976). Obviously, the vagueness and the need to work with incomplete information posed a major obstacle to such efforts. Common sense was another major hurdle (McCarthy 1980).

For instance, John McCarthy (1980) was one of the first researchers who proposed a method for handling the incompleteness. In his theory of circumscription, he defined a schematic rule that could conjecture that only the objects that can be shown to have certain property by explicit reference to given facts, are *all* the objects that have that property. For example, if there is a boat available for transporting the missionaries and cannibals across river and nothing else, the boat can be circumscribed as a sufficient and the only means for crossing the river. See also the analysis of this example in McCarthy's paper cited earlier.

When a common sense rules that a boat can be used for crossing the river '*unless there is something wrong with it or something prevents its use*', McCarthy argues that circumscription can restrict the 'world' to comply with it. In other words, if there are no explicit facts showing and requiring that *something* prevents using a boat, the circumscription assumes (conjectures) that there is nothing to prevent using the boat. Thus, there are no "somethings" that are unaccounted for – no new objects can be introduced into the circumscribed world unless they were there before circumscription.

The assertion that a certain (finite) set of explicit statements sufficiently specifies a (design) problem is not a 20th century invention. It re-occurs in the literature literally throughout the centuries. The origins of the assertion are in the famous rule by William of Ockham, articulated six centuries ago.

Ockham's rule (also known as Ockham's razor) asserts that *it is futile to use more* (objects, conditions) *when we suffice with fewer*[4]. In other words, we are indeed functioning in potentially infinite and open world; however, we attend to such a world using a minimal set of statements that are needed in a particular situation, and no more. This enables us to cope with the ambient ambiguity, incompleteness, and a potential infinity. Thus, McCarthy's circumscription is a newer version of an old wisdom.

Another similar attempt to reduce the conceptual world into absolutely necessary 'bare bones' led in the beginnings of AI as a science, to the introduction of a '*closed world assumption*'. This assumption is saying nothing else than the original Ockham's rule – i.e. only the objects explicitly mentioned are objects that are needed (~ that exist in the conceptual world). An alternative articulation of a closed-world assumption (CWA) can be found in (Cohen and Feigenbaum 1982)*: "CWA is that all relationships not explicitly stated to hold do not hold*." The same is implied by the theory of circumscription mentioned earlier, as proven by Davis (1980).

Davis expresses this construction of boundaries in potentially infinite conceptual worlds by terms of minimal models and minimal completion. Using these two terms, he articulates the same sentence as McCarthy in form: "*if a sentence is inferable from a minimal extension of a certain axiom, then it holds in all minimal models*." The essence of his theorem is in the minimal extension of an axiom that is defined in terms of relativising the quantifiers in the axiom over all formulas of the given theory. The formulas allowed to get involved in the relativisation are in Davis's paper expressed using the same symbols as McCarthy; i.e. represented by a predicate schema $\Phi(x)$ where 'x' is a free variable. Thus, eventually, Ockham's rule, circumscription, or minimal model completions do nothing other than reducing a potentially complex conceptual world into manageable, unambiguous, and above all *explicit* problem solving spaces.

Sometimes, this attempt to circumscribe goes so far that it removes the boundary between the worlds and their models. In the author's opinion, and example of such well-meant but rather strong statement is the definition of ontology by Thomas Gruber (1993). According to the definition, '*ontology is an explicit specification of the conceptual representation of a particular domain. The term is borrowed from philosophy where it is a systematic account of Existence. For knowledge-based systems, what "exists" is exactly what can be represented.*' From a knowledge-level perspective, this is a controversy promoting the explicit representation of knowledge to knowledge itself! As we mentioned in sections 3.2 and 4.1, there may exist tacit or inarticulate knowledge that cannot be explicitly represented, nonetheless still being 'knowledge'. However, the definition fits seamlessly into the framework of research activities that try to deal with the incompleteness of the 'real' world by 'closing' its conceptual boundaries.

In order to conclude this short excursion into handling incompleteness, we say that circumscription or a conceptual closure of the world seems to be an important operation for tackling the problems in a real world. On the other hand, we argue that design cannot be confined within a single conceptual world that is closed at the beginning *once and forever*! One may indeed reason about the problems inside a circumscribed world, but at the same time, there needs to be a chance to *re-circumscribe* the

[4] William of Ockham (1280-1349) – a philosopher, logician, and one of the first 'modern' representatives of a conceptual stance towards knowledge. The quote is a translation of one form of his famous maxim: "*Frustra fit per*

world in a different manner, containing different concepts and objects. In other words, we believe that design exhibits a process of circumscribing and re-circumscribing the world into different conceptual models. It is therefore an interesting question to extend the scope of research on handling incompleteness and to investigate how a 're-circumscription' can be accomplished.

4.2.2 Knowledge and reflection 'in action'

Earlier, in chapter 3 we emphasised a fundamental and epistemological difference between a 'possessed knowledge' and knowledge applied in action. This concept is closely related to the research on handling incompleteness, as we show below. It also provides a direct link to one of the fundamental concepts of the RFD theory presented later in the thesis (chapter 6). Consider the following extracts from Donald Schön (1983) describing the design activity to start with:

> "*Because of the complexity (of design), (...) the designer may take account of the unintended changes he has made in the situation by forming new understandings and making new moves.*
>
> *He shapes the situation, in accordance with his initial appreciation of it, the situation 'talks back', and he responds to the situation's back-talk. Design is thus a conversation with the situation.*"

Thus, the important lesson that can be learned from this quote is a presence of shaping in design. According to Schön, during such a 'shaping operation' designer forms a perspective, a point of view for further investigation of the problem (design situation). Thus, he or she tries to impose a particular view on the problem to both solve it and learn more about it. Compare this imprinting of an initial appreciation to the terminology from a previous section. This process of shaping is very close to the circumscription or enclosure of the world as introduced in section 4.2.1. However, Schön goes one step further when claiming that a situation can be *re-shaped* (i.e. re-circumscribed) if the need arises. Such a need may be triggered by unexpected results of the approach based on the initial shaping.

Furthermore, Schön associates another important characteristic with the process of shaping and re-shaping the design situation. He argues that it must be a *reflective* process. Thus, in order to answer the situation's back-talk, the designer has to reflect in-action on his or her current understanding of the problem. This includes the reflection and tacit appreciation of the strategies that may have been used to support a particular decision or that emerge from the design action. S/he may address the phenomena or features that are implicitly present in the performed action, which became apparent only within a retrospective reflection. In other words, through reflection it is possible to see the effects of the designer's tacit or implicit understanding on the shaping of the problem.

Not only that, through reflection it is possible to criticise such tacit understandings formed around a repetitive experience and *practice* of designing. Thus, in Schön's words, "*practitioners reflect on their knowing-in-practice*" (and their knowledge originating in practice). The 'conversation with a situation' from the quote above is therefore an interplay of shaping the situation, reflecting on it, and re-shaping. This conversation can be *repeated*, until its results conform to the designer's tacit expectations supported by his or her practical and empirical experience!

plura quod potest fieri per pauciora."

From a perspective of problem solving in the incomplete worlds (section 4.2.1), the role and purpose of a concept of 'design shaping' becomes even clearer and more definite. We argue that design 'shaping' (or 'framing') may be seen as a specific kind of circumscription that restricts two different, but mutually intertwined conceptual worlds. The designer's understanding of a problem circumscribes his commitment to certain objects, structures and models used for a solution construction. Vice-versa, the particular solution objects circumscribe the changes of the commitment to certain understanding and interpretation of the problem. The conversation of problem solutions and problem understandings only emphasises the fact that the two commitments cannot be made *independently*! If a designer attends only to one of the spaces (usually development of solutions), the entire process is crudely simplified. Such a simplified process addresses only a small part of design knowledge; see also the example involving shawl weavers using hand-made and artificial dyes in section 3.1.1.1.

Schön is not the only scientist addressing the conversation between the designer and the design situation. A similar notion oscillates also in the works of Fischer (1992), Fischer, Nakakoji *et al.* (1993), Nakakoji, Sumner *et al.* (1994), and Nakakoji, Yamamoto *et al.* (1998). We return to the achievements of this research group later in section 5.1.5. Another research community investigating the role of reflective co-development was already mentioned in section 4.1.2 – the 'Twin Peak' model of designing software systems (Nuseibeh 2001). The advances of requirements engineering (RE) reported by Nuseibeh seem to be close to the perspective of shaping and re-shaping the understanding of the problem, that was presented in this section. For instance, d'Avila Garcez, Russo *et al.* (1999), Russo, Nuseibeh *et al.* (1999), and Nuseibeh and Easterbrook (2000) emphasise the evolving nature of the design specifications as an essential part of modern software development. Thus, the 'Twin Peak' model developed by the RE community to explain the mutual interaction between architectures (solutions) and requirements may be used to illustrate also the essence of Schön's theory of reflection-in-action. For a scheme of 'Twin Peak' model, see Figure 4–2 in section 4.1.2.

The concept of repeated shaping, reflecting on, and re-shaping the situation is a foundation of the work reported in this thesis. In chapter 6, we return to this terminology and define formally a concept of 'design frame' based on the uncovered relation between McCarthy's circumscription and Schön's reflective re-shaping. We also address the issue of how is it possible that an apparently sound and logically admissible candidate solution may be tacitly perceived as unacceptable, thus triggering the need for re-shaping the situation. The re-shaping will be defined as an articulation, prioritisation or a conceptual evolution of the existing problem specification and/or conceptual closure of the vague and incomplete world of a design problem.

Another research undertaking addressing and investigating the issues of interactions between the different conceptual categories is Function-Behaviour-Structure (F-B-S) framework (Gero 1998b; 1998c). Figure 4–5 shows a typical data flow between the individual categories. Design begins with a description of product functionality F, accordingly an expected behaviour of the product B_e is articulated, and appropriate structure S is chosen. Gero refers to the sequence $F \rightarrow B_e \rightarrow S$ as product synthesis, and complements it by analysis and several types of re-formulations. The important facet of the model is that the re-formulation runs 'backwards' from structure S through the actual behaviour B_s and eventually aims to amend the required functionality ($F \rightarrow F'$) and/or the expected behaviours of the product ($B_e \rightarrow B_e'$). The last category of the model below is labelled as *D*, and stands for a design documentation, justification, and explanation of a particular choice.

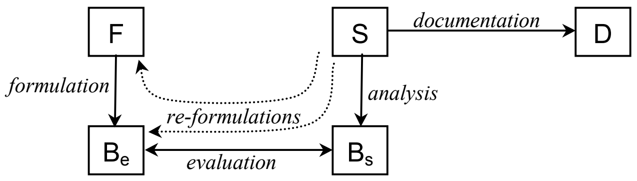


Figure 4–5. F-B-S model of designing (Gero 1998b)

Gero emphasises the importance of reflection and cognitive emergence in design (see also example in section 4.1.3). He brings in another important requirement or condition for a successful reflection. Namely, he argues that reflection is very closely bound with the situation; it matters when, where and how a particular reasoning chain occurs. This concept of 'situatedness' is manifested in the design by a selection of an appropriate behaviour and structure. This selection is not random nor is it given a-priori to the designer; it is produced in response to the design situation, and depends on the particular context, perspective, or focus.

When talking about Schön's shaping the 'problem space' or McCarthy's circumscription, we should mention again the work of Michael Jackson (2001) on software and requirement engineering. As it was already introduced in section 3.4.3, he is an author of a methodology for software design deploying his theory of problem frames. He sees the problem frames as representations of familiar classes of sub-problems in design that can be decomposed and related in a particular way. The actual decomposition and some relationships or interfaces is to the certain extent dictated by a particular problem frame.

In the context of this section, his problem frames may be seen as clearly corresponding to the notion of circumscribing a vague world. While not providing all the answers and solutions, the problem frames aim to provide a mechanism to restrict a designer's attention toward the real world on a limited set of perspectives or *concerns*. Commitment to a particular concern circumscribes the availability of certain roles, relationships, events, etc. in the specification and solution of a particular problem. The choice of a suitable problem frame is crucial, as Jackson emphasises, because it affects the whole development of a software system.

The interpretation of a particular design situation depends on how the situation is represented, and vice-versa, its representation is influenced by the interpretation. Thus, Jackson's research fits well into the context of Gero's research into design patterns, F-B-S theory and concept emergence. These two works present attempts to express Schön's theory of reflection-in-action and dialogue with the design situation in more formal terms. We return to the F-B-S framework in the next chapter, and then present our perspective for the formalisation of the 'situated reflection' and 'problem framing' in chapter 6.

4.3 Summary or what is the design about?

Section 4.1 presented design mostly in the engineering disciplines from several different perspectives. All perspectives focus on some specific aspects of design, and have their specific weaknesses and strengths. Some of the existing discrepancies between the different schools of thought were already mentioned in sections 4.1.1 to 4.1.4. Also, some similarities between the different approaches were hinted at. In this section, we try to put the findings from the different approaches together, and suggest possible associations between the works of different research groups and researchers. The key outcome of this section is a list of typical features that in our opinion characterise engineering design. This list is used in the subsequent chapters for the development and validation of our recursive model of framing in design (RFD). Alternatively, we may see this section as a critical overview of the selected gaps and issues outstanding in the current 'science of design'.

First, let us focus on the discrepancy between the searching and exploratory paradigms. As already mentioned in section 4.1.1, the key difference between the two paradigms is the active or passive role played by the agent, who solves the design problem. In case of 'design as a search', the agent has all the necessary knowledge available in a form of design spaces – i.e. spaces containing all possible design elements, structures, modules and relationships among them. This space can be according to Chandrasekaran (1990) vast, and only a small part of it constitutes satisfactory and/or optimal design solutions. If this were a correct perspective then it would be sufficient to develop fast techniques for navigation in such a vast design space, and all design tasks would become routine.

Unfortunately, this is not the case in the real world with its under-specified problems. Brute-force techniques for navigation may work in the context of relatively well-defined problems such as deciding the next moves in the games of chess or checkers, but design problems seem to resist the application of the accelerated search techniques. As a workaround, the researchers propose some 'a priori' or 'higher level' knowledge that can inform the search algorithm. This heuristic knowledge enables the agent to distinguish between a dead end and potentially fruitful path without having to reach the end of every single path. The introduction of a 'higher level control' seems to be a natural extension of the original paradigm of search. However, the exploratory paradigm as a realistic enhancement of the search does not actually live up to the promise it makes.

A typical source of heuristic extensions to the searching techniques is designer's experience with the class of similar design situations. This is an important remark; nevertheless, many theories leave untold the process, by which the heuristic 'higher-level control' is exerted. In most cases, the process of heuristic reasoning is ascribed to the designer, who acts as a kind of 'black box'. The 'black box' usually complies with some rules; however, in case of design we often refer to personal experience and empirical knowledge without immersing into any details. This observation may suggest that in addition to the explicit design space there is also another source of knowledge that cannot be positioned in the same design space and remains around to complement and perhaps accelerate the explicit search strategies. Using terminology from the caveat on epistemologies (section 3.2.4), this corresponds to a kind of 'tacit' knowledge that cannot be articulated in the terms of those well-defined, conceptual design spaces.

A similar fate seems to affect also the proposed operationalisation. It is possible to construct simpler or more elaborate operational models of design, e.g. such as reviewed in section 4.1.2. As long as the criteria for testing proposed design solutions are known in advance, and the proposed artefacts satisfy these criteria (albeit to different degrees), the various breakdowns seem to perform satisfactorily. If an explicit criterion is not satisfied to the desired degree, there are usually available various fixes, which enable designers to modify the current solutions. Some of the existing models (Ullman 1997) also acknowledge the fact that new criteria may emerge during design, but they address the issue of emergence very superficially. In most cases, the workaround refers to a 'third party' (for example, a customer) that evaluates the current state of the design and formulates what additional criteria or requirements would be desirable to include to the design problem specification. Implicitly (and quietly), it is assumed that the customer would be able to clearly articulate the under-performing portions of the current design solution, thus assisting the designer.

However, it may happen that the customer rejects a design solution, which is perfectly optimal according to the given explicit criteria, and complies with all given requirements, constraints and/or norms. Nevertheless, the same may happen also without any explicit customer acting as the ultimate judging authority. The designer may reject a perfectly sound solution, because 'it just does not seem right'. And such reasoning turnarounds are already beyond the scope of any operational model of design. As Schön (1983) comments in his book, such turnarounds in the designer's reasoning may be very common and may be due to the complexity and incompleteness of the design situations:

> "*Because of the complexity, the designer's moves tend to produce also consequences other than those <u>intended</u>. When this happens, the designer may take account of the <u>unintended</u> changes he has made by forming new understandings and making new moves. Design is thus a conversation with the situation…*"

This quote mentions one reason for a rejection of a sound solution; namely, observing unexpected or unintended behaviour. Schön described several situations from his experiments that complied with the quoted pattern. Designers he observed typically expressed their dissatisfaction in terms of the proposed solution 'not seeming to be right'. This assessment against their tacit intentions and expectations was then followed by a reflective attempt to articulate the problem in explicit terms. Although the tacit knowledge informing the decision about unacceptability remained unexpressed, it was applied for the solution evaluation and re-shaping of the design situation, which is vital in order to support the above-mentioned 'conversation between the designer and a design situation'. Another issue arising from the operational models is a crisp border between the design phases. Although Ullman (1997) and other researchers speak against the 'over-the-wall' design, many models still attempt to distinguish and separate the indistinguishable.

However, this would contradict the observed idea of a 'dialogue with design situation' mentioned earlier. Practitioners are able to understand (i.e. analyse) the design problems not 'per se', but mainly through the attempts to change (i.e. synthesise) their solutions. Vice-versa, the proposed solutions are changed and refined following a sequence of attempts to understand the design situations (Schön 1983). This observation seems to support the idea of a 'tangled' co-evolution, interplay of problem specifications and solutions rather than a crisp, stepwise development. Moreover, designers often work with the theories that are partly incomplete and/or inconsistent. Instead of trying to avoid the inconsistency thus severely restricting their creative freedom, they cope with it using iterative probing, and concurrent articulation of their intentions and the solutions satisfying them. Thus, a 'conversation with the design situation' (Schön 1983; Nakakoji, Sumner *et al.* 1994) or 'intertwined development of specifications and solutions' (Swartout and Balzer 1982; Nuseibeh 2001) are indeed the essential features of non-trivial design problems that must be taken rigorously.

Next, let us have a critical look at the cognitive theories of design and designing as they were reviewed in section 4.1.3. As we could see in section 4.1.3, cognitive scientists consciously acknowledge the existence of some non-explicit knowledge possessed by the design practitioners. They use multiple terms to refer to an obscure procedure of discovering new aspects inside the current design situations. For instance, Dominowski and Dallob (1995) emphasise a designer's 'insight' into a situation, whereas Candy and Edmonds (1996) talk about a 'deep immersion' into the design problem and the underlying domain. Cross (1997) is referring to the same situation using the terminology of 'creative leaps' and 'firm standing among familiar ideas'. Other views on this phenomenon appear e.g. in (Davidson 1995; Jagodzinski, Reid *et al.* 2000). Unlike other researchers, Cross not only states the facts but attempts to construct potential models of this 'insightful', non-explicit and non-monotonic forms of reasoning.

He proposes various methods of how a new solution may appear as a result of manipulations with the known, familiar solutions or simpler solution models. Only very little mention is given to the fact that problem specifications also need to be manipulated, and this manipulation has often a bigger impact on the designer's 'insight' than any solution modifications. Cross and other cognitive scientists support the stance as mentioned in the previous paragraphs – a modified solution serves as a trigger that makes it possible to formulate a new requirement. The designers reason about the modifications drawing on their creativity and experience from the previous exposures to similar tasks. However, at the bottom of many creative and insightful designs there seems to be knowledge complementing the explicit reasoning processes. On the other hand, reference to such concepts as creativity, insight, talent, deep domain knowledge and experience, seems to add more complexity to an already tangled problem. Are these concepts 'the reasons' for the designer's expertise in practising design, or are they 'the consequences'? If they are consequences, then there seems to be something left unattended by the various explicit models that are mentioned in this chapter and chapter 5.

The last perspective reviewed in section 4.1.4 focused on the creativity and considered TRIZ, a theory for solving of inventive problem (Altshuller 1984). This theory adopts a procedural stance towards the reasoning processes in design, and formulates an algorithm for the resolution of non-trivial problems. TRIZ is a nice example of a theory that attempts to merge the explicit rules and principles of designing with experiential knowledge. Altshuller proposed a method for an informed discovery of the unacceptable cores in the design problems. The core consisted of an exaggerated contradiction between two or more physical requirements on the technological systems. A specific contradiction characterising the problem was matched against an empirically obtained reference index, and applicable remedies were proposed. Thus, the inventive contributions of the generations of designers that are often hard to understand are generalised into a reasonably complete referential matrix.

TRIZ also proposed the illustration of the abstract principles by the specific previous cases that belong to the same contradiction class. Thus, the techniques proposed by TRIZ to address the rectification of an explicit contradiction by construction of an abstract solution model are very useful for non-trivial designs. TRIZ, according to its authors, is a theory for the innovation of the artefacts and invention of potential improvements. The previous cases act as an inspiration for the resolution of identified contradictions. Nevertheless, the process of articulating the exaggerated contradiction is usually not sudden and may involve several iterations and trials. Designer or inventor must often refer to the deep knowledge of the familiar solutions to recognise the contradiction.

Similarly, TRIZ leaves the whole reasoning in respect to how a particular fix can be implemented with the designer and his domain expertise. Although this theory is not generally applicable outside engineering design of physical systems, it represents an important milestone. The important contribution is in its very intimate relation with the specific solutions. The aspect of 'shaping' the understanding of the design problem by trying to construct potentially applicable solution models is deeply embedded in this theory. Only, the process of 'shaping' is herewith referred to as articulation of an explicit and exaggerated contradiction and retrieval of an applicable remedy.

Designers' capacity to *see-as* and *do-as* allows them to understand the vague design situations and develop often unique design solutions. From the earlier paragraphs of this section, we emphasise the interplay of both actions – seeing-as *whilst* doing-as. In other words, design requires the activity in form of an action – experiment, solution construction to verify the perceived similarity. The fundamental question of a designer when 'making things' is not a simple 'How to do it?' More often than not it is a composition of 'How to do it?' and 'What to do?' These two questions are usually very closely followed by a more speculative 'What if?' featuring an experimental probe into a design situation. Such probe validates the answers to the two previous questions, as well as confirms or rejects the conjecture. Often, the interplay of these question may lead to a discovery of something new that was previously overlooked.

Whilst a 'What if?' question stands typically for an explicit action it is not applied blindly and randomly to a design situation. Usually, it is accompanied by a validating 'Do I like what I got?' question. This validation is often left to a subjective and tacit judgement of a designer or a 'third party'. Thus, 'tacit' validation is an interesting form of 'meta-control' of the design process. It helps to cope with incomplete and under-specified situations, and at the same time prevents the designers from generating too many conjectures. Design is unlike scientific experimentation that is concerned with the rigour of an action (experiment). Design takes into account the designer's subjective expectations, personal attitudes, assumptions, intentions, and overall impressions from the proposed solutions. In other words, the assessment of the logical validity of designed products is unquestionably needful, but it is not sufficient for designing! There are also extra-logical means playing an important role in design, and in our opinion these means deserve a proper place in the models of the reasoning in design.

Let us conclude this brief summary of interesting aspects of design with a list of tenets that in our opinion, characterise the design activity:

- Design develops new artefacts that deliver certain desired goals.
- Design is inherently an iterative process.
- The understanding of the 'desired goals' changes and develops as well.
- Design relies on the exploration of consequences through conjectured experiments.
- Design uses a variety of knowledge sources including a designer's previous experience.
- Analogy and similarity are important vehicles for 'seeing-as' and 'doing-as'.
- Multiple parallel lines of enquiry are open during design.

- In addition to rigorous assessments, a subjective feel for the situation is also a driving force in non-trivial designs.

These characteristics of design are applied in the construction of the recursive model of framing (RFD) in chapter 6, as well as in the evaluation of the experimental work (see chapters 7 and 8). Nonetheless, before proposing the RFD model and reporting on the conducted experimental work, a few of the existing implementations of the knowledge-based design support systems and models are reviewed in the subsequent chapter. The following chapter will be then be concluded with an articulation of the gaps between our perception of design and that of the reviewed approaches.

5 KNOWLEDGE-INTENSIVE DESIGN SUPPORT

So far, this document has discussed engineering design on an abstract level from several different perspectives, and highlighted the relationships between these perspectives. The review of the different views on design in chapter 4 concluded with a list of the essential attributes of non-trivial design problems. It was argued that inarticulate, reflective knowledge applicable to the reasoning processes in design is a natural extension of the explicit knowledge of design elements, methods and practices. The tacit knowledge was typically manifested and criticised in reflective re-shaping of design situations. The close relationship between these complementary types of design knowledge was identified in the work of several researchers, and certain invariant references to such relationship were summarised in section 4.3.

The objective of this chapter is to review selected efforts aiming to articulate knowledge-level models of design accounting for various empirical observations reviewed in chapter 4. A parallel goal is to discuss several implementations of various theoretical frameworks addressing the same findings. In the review presented in this chapter, we focus on the design models and applications that address the gap between the initial, vague expectations, and the formal specification of a design task.

First, let us define the contextual boundaries in which the subsequent discussion is grounded. As it was the case with the definition of engineering design, there are several definitions and views of what is actually meant by 'knowledge-intensive' or 'knowledge-based' design models and/or tools. For instance, Gerhard Fischer (1992) formulates the point, in which the different definitions seem to diverge, as those attempts trying to 'get the human out of the loop' versus attempts trying to 'get a computer into the loop'. Similar categorisation can be found in (Garrett 1998). From this point of view, it is possible to divide the existing research on design support models/tools roughly into three categories:

1. *automated design systems* (often referred to as intelligent CAD or I-CAD), whose aim is an automation of design processes; once the expert knowledge is acquired and suitably encoded in the I-CAD system, the role of human designers is that of a source of additional requirements or new evaluation criteria (MacCallum 1990);
2. *knowledgeable design assistants* that are able to assist the human designer in one or more phases of a design process; these tools typically have an extensive problem-specific (sometimes also domain-specific) knowledge to recall and/or adapt similar design cases – CBR (Watson and Perera 1997). Often, they maintain consistency of multiple design contexts, and are able to do routine reasoning (Smithers, Conkie *et al.* 1990; Tang 1997);
3. *knowledgeable design critics* usually follow formal development of a product or the informal justification of designer's decisions, they tend to draw on a rich domain-specific knowledge, and 'criticise' or 'comment on' the explicit patterns of design (Fischer 1992; Nakakoji, Sumner *et al.* 1994)

From the three categories, we focus on the last two. This choice is supported by the stance toward tacit, inarticulate knowledge (and knowing) as presented earlier in the thesis. In the words of Cook and Brown (1999) or Schön (1983), in order to take the full advantage of tacit knowing one needs to be involved in an action. The explicit formulation of a designer's understanding of a problem is, generally, insufficient to guarantee a proficiency in a particular action. Thus, the idea of placing a human being into a role of 'Big Brother' commanding a computational system through an explicit formulation of requirements and criteria as argued in point 1) above, is flawed. It inhibits the ability of humans to increase their expertise in the field. It also prevents them acquire deeper understanding of the domain or the design problem. Therefore, we subscribe to Fischer's (1992) view that automatic programming is not achievable when the goals are articulated outside the solver. Fischer continues that it is also not desirable because humans enjoy '*doing*' – in addition to observing others doing the job.

Rather than automating engineering design using computational tools, we see the computers as <u>*enabling tools*</u>. These tools should manifest the ability to enhance the power of human designers, as well as shed some light on the reasoning processes underlying design. Therefore, and in order to underline such a position, we use the definition of a knowledge-intensive design tool originating in AI research group at the Edinburgh University (Tang 1997):

> (Knowledge-intensive design tool) *is a computational system that supports decision making during the design process, and enables designers to explore the structure of design problems and their solutions. It achieves this objective by combining human designer's (tacit) experience and (explicit) expertise with the domain and design knowledge that may be stored in a computational tool.*

In line with the perspective seeing design support systems as enabling technologies for the purposes of exploration of the problem structure, it seems reasonable to demand some additional properties for the support systems. Through an informed exploration, designers typically extend their current knowledge about the design problem, its structure, particular domain, etc. In order to make a support system useful; one important property that needs to be targeted is its ability *to adapt* – to acquire new knowledge. Adaptability is according to Simon (1969) one of the most significant human characteristics – perhaps the most human one. Humans adapt to their external environment by acquiring new knowledge or skills. They adapt to the given problems by articulating new conceptual frames thus interpreting those problems in a suitable fashion. Any supportive tools that aim at enhancing human powers should also exhibit these (or similar) adaptive capabilities. In this particular sense, it might be even allowed to use the adjective 'intelligent' in connection with design support to replace or enhance more 'down-to-earth' term 'knowledgeable'.

The subsequent section gives a brief account of the selected research efforts in the domains of design theory and design support. As we mentioned earlier, we focus on the categories of design assistant and critics (points 2 and 3 in the list above). The approaches reviewed are chosen as typical representatives of these categories. Further details of the briefly reviewed tools, methods and approaches are available from the respective sources referenced in section 5.1. In line with the theoretical proposal of using a knowledge level as a base of our investigation (see chapter 3), we look at the roles the models typically play in design. Section 5.2 therefore focuses on one specific approach to modelling reasoning in engineering design – knowledge-level models (see also introduction in section 3.2). The discussion about knowledge-level models first highlights the significant outcomes of selected research projects contributing to this field. The purpose is to identify and comment on the desirable features of knowledge-level models, which are vital for the formulation of the theory of recursive framing in design (RFD) later in this thesis.

5.1 Overview of design support philosophies

In this section, different approaches towards design support are reviewed, including their underlying architectures and philosophies. The aim is to address alternative research directions within the field of knowledge-intensive support for engineering design, and give the reader a short account of several usually deployed approaches. It does not mean that the absence of one approach from this overview makes it less important. Many different approaches were looked at, and instead of exhaustively reviewing all of them, we opted for the selection of few that are typical representatives of their classes – one may even say 'archetypes' (Gero 1990):

- *Edinburgh Designer* as an example of the use of a blackboard architecture and truth maintenance within an exploratory design;
- *KRITIK and family* to illustrate the case-based approach to design support as well as the opportunities provided by the deployment of domain models;
- *IMPROVISER* as an implementation of a cognitive model of the design process using a case-based and constraint satisfaction approach;
- *Invention Machine Lab* as the implementation of TRIZ – a theory for the solution of the inventive problems by articulating and removing the physical contradictions;
- *DODE-s* featuring several domain-oriented design environments implementing the design rationale, critiquing and commenting

Each of the tools is reviewed with a focus given to the tenets that were articulated as a conclusion to the review of different perspectives on design; see section 4.3. These conclusions summarised the features we believe can be ascribed to the majority of non-trivial design problems. Therefore, we may consider the review of the tools as validating our conclusions about the properties of a design process. To refresh the reader's memory, the most important features included:

- Design is inherently an iterative process.
- A designer's understanding of the design problem is also developing.
- Design relies on the exploration of consequences through conjectured experiments.
- Design uses a variety of knowledge sources including a designer's previous experience.
- Analogy and similarity are important vehicles for 'seeing-as' and 'doing-as'.
- Multiple parallel lines of enquiry are open during design.
- In addition to rigorous assessments, a subjective feel for the situation is also a driving force in the non-trivial designs.

5.1.1 Edinburgh Designer

As its name suggests, the Edinburgh Designer System (EDS) is a knowledge-based design support system developed at the University of Edinburgh within their 'AI in Design' research programme (Smithers, Conkie *et al.* 1990; Logan, Corne *et al.* 1992). It would be more correct to say that it is a series of developments in the field of knowledge-based support for design, which took place in course of several years. EDS is an embodiment of a knowledge-level view of engineering design that treats the design process as an exploratory task. In EDS, a design task typically starts with some initial requirements. These are refined during the design into ‘design description documents’ (DDD). The construction of these design descriptions is not a one-step activity; it is highly interactive in terms of knowledge that is available to the designers and that is generated in the process. Such knowledge is stored as another basic entity of EDS – ‘domain knowledge base’ (DKB). Typically, DKB serves as a ‘bootstrap’ to start up the design process by identifying the regions of interest in the vast space of possible designs that can be feasibly accomplished in a particular domain.

DKB contains knowledge in form of so-called *clichés* that can be identified with the concept of generic knowledge-level models of the elements, modules and relationships in a particular domain. However, as the authors emphasise, the clichés are closer to the frames (Barr and Feigenbaum 1982) rather than prototypes (Gero 1990). This remark is made in the context that the clichés may contain also some knowledge about how to design. They may explain how to use a particular module or calculate a particular variable. Clichés are also organised in hierarchical structures, thus creating a link with the research on common ontologies (Gruber 1993) and knowledge-level models (Newell 1982).

One of the conceptual methodologies of EDS is the *exploration* of the space of potential designs. This is achieved by regarding all the designers’ statements and decisions about design as *assumptions*. Term ‘assumption’ is borrowed from de Kleer’s (1986a) research on ATMS – assumption-based truth maintenance systems, and denotes currently believed statements about the problem at hand, external world, etc. Assumptions are to be distinguished from the axioms in a sense that the belief of axioms is ‘unchanging’, whereas, the belief of assumptions is subject to changes and re-validations. ATMS plays an important role in maintaining several parallel lines of enquiry that may represent alternative decisions or solutions, and may not be mutually compatible. The alternative directions are introduced as assumptions and ATMS takes care of dividing the clashing assumptions into independent *contexts*. Contexts can be manipulated through the modifications of the underlying assumptions.

Clichés are an important concept from the knowledge representation and presentation perspective. They are stored in DKB, and may contain definitions of various classes of devices and components that belong to a particular domain, various applicable parameters, constraints and descriptive constants. In addition to the declarative and parameterised relations, they may contain knowledge of how to design a particular (sub) component. Here is a simplified example, how such a knowledge representation may look like (extracted from the detailed description in (Smithers, Conkie *et al.* 1990)):

Class name:	water turbine	
Parameters:	*Initial/given*:	speed, diameter, material, efficiency, design output
	Calculated:	discharge rate, maximum/minimum head, calculated output
Variables:	efficiency of generator, rated output of turbine, optimum speed, optimum discharge	
Constraints:	optimum speed of model is 70.0 rpm, discharge rate of model is 1.2 m^3/s	
‘How to’:	see functionTable ‘discharge_characteristic’ interpolating the variables guide vane opening, discharge rate and optimum speed	
	see functionTable ‘efficiency_matrix’ for non-interpolating relation between . . .	

EDS as a system has a blackboard architecture that controls the calls to different knowledge sources depending on the sets of underlying assumptions and relevant operations. A new datum can be inferred using any of the available knowledge sources, such as module class definitions, clichés, geometric or algebraic engines. An interesting feature is that a designer is also modelled as one of the available knowledge sources, and as such has to 'compete' with the other sources to have a chance to contribute to the process of inference. The designers are in the context of EDS and similar systems, typical sources of the design assumptions. If a designer's contribution is activated as a knowledge source, EDS accepts the decisions and then propagates the consequences of such external decisions throughout the knowledge base, other activated knowledge sources and active design descriptions (DDD). The same propagation takes place with a contribution coming from another source (e.g. algebraic calculation).

Generally speaking, EDS is a representative of a large class of similar approaches to engineering design, and as such, it is reviewed in this document. Multiple knowledge sources, opportunistic control strategy, tight co-operation between computational tool and human designer, consistency maintenance and parallel design contexts are all aspects complying with the characteristics of a design process listed earlier. Further details can be found, among many others, e.g. in (Tang 1997).

However, in spite of many interesting ideas that appeared in course of the development of the prototypic systems, the authors agree that there are still many areas not tackled by the EDS approach. Among others, the role of meta-knowledge and designer's tacit knowledge in designing artefacts is certainly underestimated. As it was already mentioned, designer is treated as one knowledge source. However, unlike with the other knowledge sources in EDS, the designer's comments are never rejected as possibly irrelevant. If they violate the existing assumptions or contexts, new contexts are created and the remaining knowledge is respectively re-organised.

Such behaviour is good (as we discussed above), but it is also a source of many sudden decisions that are not necessarily grounded in the currently active explicit context. To the certain extent, they may reflect designer's tacit knowledge; however, the designer is an 'unquestioned source' of design assumptions. EDS is not focused on any form of help or suggestion about further exploration of the DKB, alternative directions or potential dangers of a chosen path. Similarly, EDS does not provide space for conjecturing unusual approaches to the problem at hand, apart from propagating the designer's assumptions. Thus, EDS is able to refine the view as 'shaped' by the designer's assumption and perform routine reasoning within available DKB. However, the system lacks a method or perhaps a specialised knowledge source that would be able to explain or justify such shaping.

To summarise the points reviewed above EDS and the related class of blackboard-based systems with an opportunistic control flow is a significant development in the field of artificial intelligence deployed for design support. Many of the ideas formulated in EDS resound throughout this document, albeit in a modified, extended, and more formal shapes. We emphasise the feature of multiple contexts and the opportunistic control strategy featuring exploration as the most valuable contributions of EDS.

5.1.2 KRITIK, IDeAL and family

As in the previous section, when a family of several design support tools has been reviewed under one name, this section also looks at several systems. This time the review is concerned with systems that were developed by Ashok Goel and the research community based at the Georgia Tech, over the course of several years. The review highlights the essential concepts from their approaches, and additional details can be found e.g. in (Bhatta and Goel 1994; Bhatta, Goel *et al.* 1994; Prabhakar and Goel 1998). Although KRITIK is the original 'founder' of the family, the review is at large based on the advanced and re-worked design support tools, such as IDeAL and E-KRITIK that evolved and inherited many features from their ancestor.

From the perspective of tool architecture, all the above are integrated case-based and model-based design systems. The essential concept is in the analogy-based retrieval of similar cases solved in the past from a memory of cases. The retrieval engine uses richly indexed base of numerous designs accomplished in the past, and as a key index it uses the functionality that was delivered by the particular design. In addition to knowledge of delivered functionality of the stored design solutions, each design case also contains a description of an underlying structure and a description of causal behaviour(s). Causal behaviour is a link between a particular functionality and a combination of the structural elements used in a particular design case to implement the functionality. Once a case analogous to the current problem is retrieved from a memory, the system focuses on its behavioural model. The behavioural model of an analogous case is revised to incorporate any particularities of a new problem and forms a skeleton of the new design solution.

New design solutions are typically verified through a qualitative simulation; i.e. the system tries to prove that it is possible to infer the desired functionality from the particular structure and behavioural links. The IDeAL tool extends the validation and is able of simple ‘learning’. The new designs (including selected structures and adapted behaviours) are cross-indexed with the existing cases. If a sufficiently large number of similar problems are tackled, there is a chance of learning interesting commonalties among the stored records. The stored records are represented as triplets called S-B-F paths – Structure-Behaviour-Function. Alternatively, as Qian and Gero (1996) argue that these should be F-B-S paths because design typically starts with a known or desired functionality, and structures are designed (e.g. abduced) accordingly. An example of the S-B-F representation of a sulphuric acid cooler (SAC) is borrowed from (Bhatta and Goel 1994) and is presented in Figure 5–1.

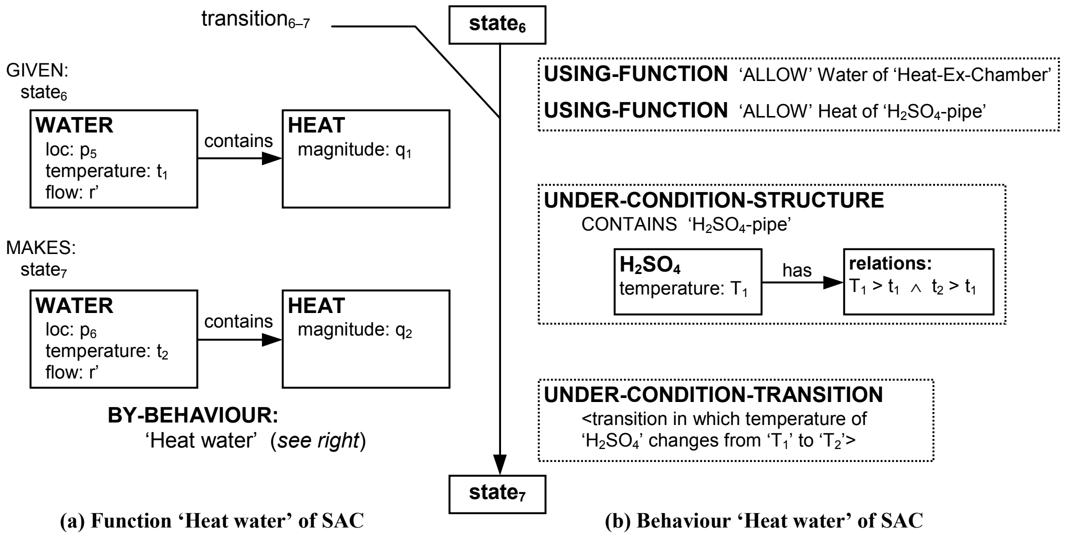

transition$_{6–7}$
state$_6$
GIVEN: state$_6$
WATER loc: p_5 temperature: t_1 flow: r’
contains
HEAT magnitude: q_1
MAKES: state$_7$
WATER loc: p_6 temperature: t_2 flow: r’
contains
HEAT magnitude: q_2
BY-BEHAVIOUR: ‘Heat water’ (see right)
USING-FUNCTION ‘ALLOW’ Water of ‘Heat-Ex-Chamber’
USING-FUNCTION ‘ALLOW’ Heat of ‘H_2SO_4-pipe’
UNDER-CONDITION-STRUCTURE CONTAINS ‘H_2SO_4-pipe’
H_2SO_4 temperature: T_1
has
relations: $T_1 > t_1 \wedge t_2 > t_1$
UNDER-CONDITION-TRANSITION <transition in which temperature of ‘H_2SO_4’ changes from ‘T_1’ to ‘T_2’>
state$_7$
(a) Function ‘Heat water’ of SAC
(b) Behaviour ‘Heat water’ of SAC

Figure 5–1. An example of SBF representation (Bhatta and Goel 1994)

The whole family of support systems is based on the early ontological research of Bylander and Chandrasekaran (1985). Goel extended the representational schema of components and substances proposed by Bylander and Chandrasekaran into S-B-F (or Structure-Behaviour-Function) model. For example, terms ‘WATER’ or ‘HEAT’ in the figure are substances, and ‘H_2SO_4-pipe’ is a component with certain substance. In general, components are usually constituent structures that may have various substances and relations among them. Functions are special schemas that relate the state (of a device or component) at the input with the output state through a causal chain of behavioural transitions. For instance, the function of SAC is described as a change in the temperature of water ($t_1 \rightarrow t_2$) by the absorbed 'HEAT' and flowing between two locations p_5 and p_6 (see left-hand part of Figure 5–1).

Since the functions are described as relations between input and output, they may be organised according to various types (e.g. transformation, movement, etc.), and such a typology then serves as an index. The types are also inherited from (Bylander and Chandrasekaran 1985), and in the figure are represented by term 'ALLOW', which is one of the fundamental actions that can be carried out with a substance or component. Thus, the statement 'USING-FUNCTION ALLOW…' on the right in the figure refers to a class of those devices or components that allow the presence of a particular substance ('Water') in a particular component ('Ex-Chamber'). The substance and component are always local to a tackled design problem.

Finally, behaviours are defined as state transitions between the states of a device or component. Mostly, causal relations are considered for the description of means how the transitions follow each other. An example of behaviour for 'heating water' by absorbing the heat from sulphuric acid is schematically depicted on the right-hand side of Figure 5–1.

Model-based component of the design tools is investigated in a very specific context. What the authors call 'model-based hypothesis formation' is essentially an application of well-structured domain knowledge of the functional typology or known structures. The background information is used in order to identify the suitable part of a design case that may be generalised, and what kind of generalisation may be safely performed. A result of any such generalisation is the creation of a hypothetical design case – a *model scenario* that subsumes the specific design cases. Such hypothesised models can be later re-used for developing new devices that exhibit certain functional similarity to the learned models. This approach to 'prototype' formation is also advocated by Gero (1998a).

One of the tools developed as a part of these efforts extended the notion of S-B-F paths to include the interactions with the environment of a designed artefact (SBF approach originally dealt solely with the internal behaviours). The motivation underlying this extension is that it is often the case in engineering design to re-engineer an existing device so that it fits a new environment. Such an extension introduces a new concept to the whole research – the issue of artefact *adaptability* to the changing external conditions. Thus the environmental interactions are modelled in parallel with the internal behavioural transitions, and often the generalised patterns discovered across the specific design cases may be used to justify a particular internal or external transition.

In other words, E-KRITIK as an environmentally aware spin-off from the original KRITIK, relies more on the model-based component of the tool. It uses it not only for the generalisation of design cases but also for reasoning about behavioural transitions. At the same time, this design tool contributes by distinguishing between the 'intrinsic' functions of a device and the 'environmentally-bounded' ones. The former class conforms to the view that is more common, and that sees functions as abstractions of the complex internal behaviour of the device components. The latter class is seen as an abstraction of the interaction of a device with its outer environment. This 'environmental' view on functionality is important because it upholds the ideas discussed at the beginning of this thesis that a design task is often specified as a set of required changes to the environment. These changes then imply what functionality the artefact should exhibit. Pursuing this point to its extreme, for example, Qian and Gero (1996) argue that it is the actual (often the sole) purpose of a designed artefact to deliver functionality as desired for the changes in the external world.

5.1.3 IMPROVISER

Another case-based tool for design support is IMPROVISER (Kolodner and Wills 1996). Unlike the ones reviewed in sections 5.1.1 and 5.1.2 that focused on the knowledge-level account of designing, this model and its implementation are more interested in the cognitive aspects of design. Methodologically, this system subscribes to similar models of design as those reviewed in the empirical work of Cross (1997) and Candy and Edmonds (1996) in section 4.1.3. Especially, the notion of 'sudden recognition' of an artefact feature or suitability for certain purposes is reflected in the above-mentioned work. As Kolodner and Wills argue, '*what is noticed as being relevant not only influences what new variables emerge but it may also determine how an object is used*'. The sudden and unexpected recognition of a new object often involves a 'discovery' that the ordinary and common objects have new functions or serve new purposes. Such a gradual emergence of design criteria, parameters, requirements and constraints is then an essential feature of creative and innovative design.

According the IMPROVISER model that was subsequently implemented into a design tool, the focus is given to the preparation phase of a product design. This is the divergent phase when the problem may be viewed from different angles (see also 4.1.3). New ideas may emerge during the preparatory phase, or the existing constraints may be relaxed and modified, and the problem may be re-formulated and re-interpreted several times. What is evolving in this phase is mainly the problem specification – however, it happens in parallel with recalling potential solutions and past cases. Preparation is followed by the assimilation of discovered features, or as Kolodner and Wills call it – '*making sense of proposals*'. During assimilation, conflicts may arise in the current perspective, and the adaptation of a proposed solution must be performed. Simultaneously, assimilation gives feedback to the preparation, and it may return the design process back to the closer investigation of the issues.

The implementation of the cognitive model that was briefly introduced above, contains several modules that are dealing with and providing support to the different cognitive actions. At the bottom is a 'long-term memory' that contains a library of previous design cases. Tightly bound to it is a 'working memory' that keeps the design options under consideration for fast access. Relevant designs are stored as problem contexts, and they serve as one of the inputs for the solution transformation module. This module is able to perform various cognitive operations upon design contexts; e.g. it can merge several contexts, augment them with the experimental data, etc. Design specification is evolving through manipulation of existing knowledge in a specific module called situation assessment. This module has the means to discover the opportunities for re-interpretation of the current findings and generalisation of the empirical data. From the point of controlling a design process, such an opportunistic control strategy fits best the blackboard architecture (similarly as for the tools reviewed in section 5.1.1).

Vital concepts of IMPROVISER are problem contexts. Each context represents a sub-problem of an overall design problem and it contains knowledge, ideas, and constraints specific for a particular sub-problem. The contexts are organised along two complementary dimensions. First, sub-problems are refinements of broader, higher-level problems. It means that a complex problem is decomposed into several simpler sub-problems that may be tackled separately. In addition to this hierarchical structure, sub-problems and their descriptive parameters may exhibit horizontal – i.e. 'peer-to-peer' relationships. The sub-problems may share parameters and influence them in a contradictory or co-operative manner. For instance, from an example described in (Kolodner and Wills 1996), the egg launching sub-problem may tend to increase the launch strength of a spring, and this would increase the capacity of the plant. However, the faster the eggs are moving the more likely they are to crack, therefore a cushioning sub-problem tends to restrict the velocity of the eggs. The two sub-problems contradict each other in respect to the velocity of eggs and plant capacity. This is an interesting and unexpected parallel with the research reported in section 4.1.4 and also 5.1.4.

Each context is described by a set of considered design options (cases or partial solutions). It also contains a set of conditions the solutions must satisfy, an index between the options and conditions regarding the degree of their satisfaction, and finally constraint priority schemas. Contexts are created dynamically as the proposed options are considered and the problem specification evolves. If the system unexpectedly finds a relevant feature or an existing case addressing an issue, it may add it to the current problem context, and the context may need to 'assimilate' a new chunk of information. Thus, the role of problem contexts of IMPROVISER may be associated with the operations of shaping and re-shaping of the problem (section 4.2.2), and our concept of 'problem framing' (chapter 6).

IMPROVISER assists the designers in several ways. For instance, from the given sub-problems it may generate a dependency graph of various variables, thus drawing designer's attention to potential conflicts. It may use the fixes associated with the constraints to search for a better assignment to a particular parameter. From this perspective, this tool behaves as a typical constraint checking and propagating engine (Kumar 1992). Thus, it represents another variant of a hybrid system, similarly as all tools mentioned in the previous sections. E.g. EDS from section 5.1.1 that was built on the deployment of abstract, knowledge-level models of domain elements and maintained the consistency in the inferred knowledge through chunking designer's assumptions using truth maintenance paradigm. KRITIK from section 5.1.2 used a library of previous cases and applied simple model-based techniques for reasoning about it.

Unlike those tools, IMPROVISER deploys a base of previous design problems, and attends to the current design task through formulations of additional constraints. Constraints from multiple sub-problems are propagated, and fixes hardwired into the problem contexts are used to resolve the conflicting influences. IMPROVISER tool can be considered one of the most advanced implementations of the theoretical research, which aimed to develop cognitive models for the design problem solving. In contrast to the majority of other approaches, IMPROVISER addresses also sometimes 'woolly' terms that are used by other cognitive scientists, such as 'enlightenment' or 'insight'. As we already mentioned, it also addresses the issue of problem shaping (framing), and acknowledges the existence of contradictions in the problem solving theory.

5.1.4 Invention Machine™ Lab

There are have been several attempts trying to apply the findings and suggestions of the theory for resolution of inventive problems (TRIZ) proposed by G. Altshuller (1984) in the 1950s. One of the most successful is a set of tools marketed by Invention Machine™ Corporation under name of IM Lab or the latest release as TechOptimizer. Before analysing this software in respect to the support given to design, the basic aspects of the original Altshuller's theory are repeated so that the reader can have a fuller picture. For further information about the theory, see section 4.1.4 and also (Sushkov, Mars *et al.* 1995; Shulyak 1999) and the web site of the community on http:// www.triz-journal.com.

TRIZ was built as a theory for the systematic description of the early phases of design problems although its primary concern was with highly innovative and inventive tasks. Unlike most of the other tools that are reviewed in this chapter, TRIZ and IM Lab pay more attention to the systematic articulation, representation, and utilisation of design knowledge. Particularly important is the knowledge of how to manipulate the elements, cases and models known in a particular domain. The essential knowledge source of the theory and the tool is a large repository of inventions and patents accepted by former Soviet and worldwide patent offices. Based on the study of these archives of design cases, a few conceptual pillars were formulated by Altshuller:

a) there are certain domain-independent regularities in solving technical problems;
b) technical systems evolve through the elimination of identified conflicts;
c) any inventive problem can be exaggerated and represented as a conflict between requirements and constraints of a technical system;
d) knowledge of various physical principles and physical effects can be used to identify and resolve the contradictions

Especially, the first concept of regularity and the second of conflict resolution are particularly strongly emphasised. These concepts led to the construction of an 'inventive matrix', in which the typical combinations of various physical effects are identified and represented as abstracted patterns of the contradictions. The term 'conflict' or 'contradiction' is the key to the entire theory that provides several means for its removal and resolution. Consulting a matrix of 'inventive principles', the inventor is presented with an abstract model of inventive solution that needs further elaboration for a particular domain and task.

Invention Machine Lab (or also IM Lab) focuses exactly on the above-described problem definition and conflict identification rather than specific design solution. Having narrowed the real problem down to a small set of fundamentally conflicting requirements, IM proposes possible solutions from a knowledge base of inventive principles. The base of principles summarises the analyses of millions of patents and thousands of physical, geometric and chemical effects. The primary module of the tool is IM Principles that aims to resolve the engineering conflicts and find an alternative strategy rather than compromising a design by retracting one the conflicting requirements. The interface allows a designer to define a problem in a TRIZ-friendly, structured form. From a list of known features (such as speed or energy consumption), s/he selects a pair consisting of a *desired* feature, which is to be improved, and a *contradictory* one to be avoided. Based on these input parameters, a number of principles are proposed to resolve the conflict, supported by examples of how they have been exploited previously. Each example has a rough sketch and respective behavioural description available.

Consider a typical example borrowed from the IM Principles demo. A red-hot bulky part is craned to the oil bath to harden, but being hot, the oil catches fire and its surface starts burning. This must be avoided. Therefore, designer looks for a usual solution – a lid with a hole in the middle. However, this is not acceptable because it is complex and makes it difficult to manipulate the bulky part. Ideally, the lid should not be there because it hinders the craning. Next, the designer expresses the specific contradiction in standard terms as follows. The conflict is that oil burns because of large surface area exposed to high temperature; when the large area is covered by a lid, this makes the manipulation difficult. The system presents three standard solution models: (i) segmentation, (ii) vibration or oscillation, and (iii) phase transition. The designer chooses 'segmentation' and inspects previous solutions applying this principle – e.g. a porous polisher as described in section 4.1.4. Finally, he adapts the abstract model as shown in Figure 5–2 (on the right) by replacing a rigid lid by a layer of non-combustible granules. These cover the surface and at the same time easily 'give way' when the bulky part is lowered to the bath. The same granules may also feature a principle of phase transition, if they were made from material that in contact with fire released extinguishing gas.

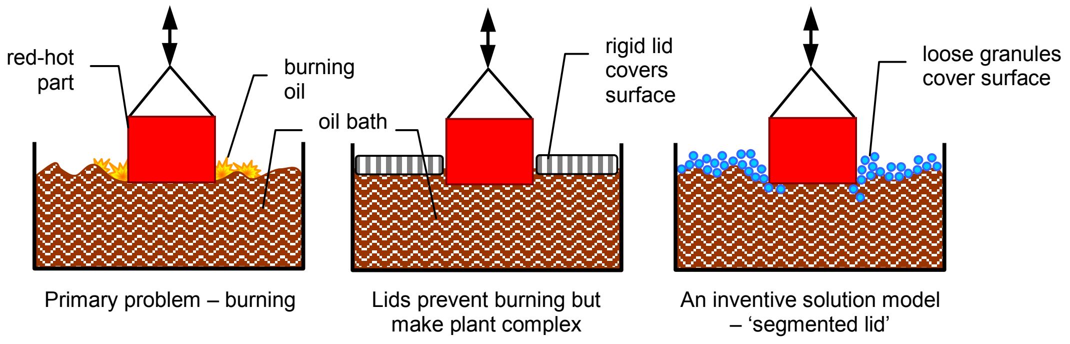

red-hot part
burning oil
oil bath
rigid lid covers surface
loose granules cover surface
Primary problem – burning
Lids prevent burning but make plant complex
An inventive solution model – 'segmented lid'

Figure 5–2. Illustration of sketches from IM Principles

In addition to the inventive principles there is also an 'IM Effects' module that associates various functions of the technical systems that can be potentially required with various physical, chemical and geometrical effects. Typically, these effects deliver the commonly desired functions in a non-standard manner. Similarly as with the principles, effects are also illustrated by existing examples of how the proposed effects have been exploited in the past designs/inventions. For example, a well-known capillary effect of liquids may be used to a great benefit for accelerated cooling or warming of the machined parts. An illustrative example is common medical thermometer.

The last module – 'IM Prediction', is basically comparing the current state of a technical solution with an 'ideal state'. System ideality is one of the main laws underlying the TRIZ; it claims that any technical system evolves so that the same functionality is being delivered by simpler and cheaper designs[5]. This module is able to 'predict' how a particular relation between different substances may influence the performance of the system – again, based on the past cases.

[5] 'Simpler and cheaper' is understood in respect to resources needed to deliver and maintain the functionality, incl. energy, material, information, etc.

The contribution of the development of various tools based on the TRIZ is in their focus on the correct definition of the design problem. Clearly, the IM suite of tools does not try to solve design problems automatically. By forcing designers or inventors to adopt a structured approach to their work, it does develop re-usable techniques and a vocabulary for a problem definition. The knowledge base of the IM suite itself is valuable, offering an extensive range of principles and especially specific examples that can be searched separately from the structured approach. Thus, designer has a chance of a 'hands-on' approach. These are also the main reasons why IM, TechOptimizer and the related tools were included to this brief review of interesting design support philosophies.

TRIZ and IM Lab can be very helpful in their structured and systematic approach. However, the actual problem definition is approached similarly as in EDS (section 5.1.1). It means it is up to the designer to reveal the conflicts in his or her design problem, and interpret them in particular standard terms. Similarly, the abstract solution models are exemplified, but the designer has to have sufficiently extensive knowledge of the domain to 'translate' a principle or effect into the vocabulary of the tackled problem. This was illustrated above, when a principle and concept of 'segmented lid' was understood abstractly – introducing totally different structures (granules) that behaved similarly.

Among the vital lessons from Invention Machine are surely the structured and systematic approach to design problems, and a rich reference to the previously solved design problems in the same and different domains. Another important attribute emphasised in the theory, is the acknowledgement of certain regularities in tackling the non-trivial (often inventive) design problems.

5.1.5 DODE-s or domain-oriented design environments

Gerhard Fischer together with other researchers from University of Colorado at Boulder developed a series of design tools that were broadly referred to as domain-oriented design environments or DODE. These environments were developed for a range of different domains. For instance, VDDE investigates the user interface design (Nakakoji, Sumner *et al.* 1994; Sumner, Bonnardel *et al.* 1997), JANUS and KiD look at the interior layout and floor planning (Fischer, Nakakoji *et al.* 1993), IAM-eMMa and ART support the image and document authoring (Nakakoji, Yamamoto *et al.* 1998). A summary and overall philosophy of the DODE approach is also discussed and assessed in (Fischer 1992). Rather than reviewing each of the modified and tailored environments separately, this section focuses on highlighting the important similarities that are invariant across such a diverse range of domains. Some tools are better to illustrate one aspect, whilst other environments perform better in another aspect of the basic philosophy.

Generally speaking, DODE-s belong among the class of systems known as knowledgeable *design critics*; i.e. the third category of the design support systems as it was introduced at the beginning of this chapter. The main purpose of all of the above-enumerated tools is to complement the designer's efforts, and assist to his or her uncovering of the initially implicit features in the design situations. Using different words, critics can be seen as systems that are able *to augment the ability of designers to evaluate their designs and make decisions concerning what to do next* (Nakakoji, Sumner *et al.* 1994). DODE-s were developed as a reaction to the approach that took the form of knowledge-based design assistants and representatives of which were reviewed in the previous four sections. Fischer (1992) argues that design assistants often lack the vital features that makes it difficult to re-use the principle of one tool across multiple domains. Before making any premature comments, it must be noted that Fischer uses the term 'design assistant' more in the sense of an automated design system that takes well-defined requirements, and produces a well-structured design solution. It means that his view is closer to our first category of design tools although it uses different terms of reference!

Bearing this note in mind, we can then agree with Fischer's disapproval of 'automatic designers'. Such tools typically have strongly developed task-specific knowledge – i.e. knowledge of design techniques and methods. Domain-specific knowledge is not common for all tools from the category of 'assistants'. Very often, the knowledge of a domain is associated with a repository of previous cases, and little attention is given to the domain or problem-specific design methods or principles. Often, the assistants try to interpret the domain-specific cases in terms of methods or principles that are more general and not domain-specific.

According to Fischer, the lack of deep domain-specific knowledge may indeed limit the amount of support given to designers, as well as hinder the communication between designers and customers. DODE-s aim exactly at the major gap in these automated and assisting design systems. Namely, the integration of problem setting and problem solving. Fischer clearly subscribes to and supports Schön's (1983) argument that *professional practice (of designing) has at least as much to do with defining and understanding a (design) problem as it does with solving a problem*. DODE-s approach this issue by employing various languages of doing (e.g. various sketches, mock-ups, etc.) for the representation of the initial intention. Such languages have a potential to simplify further reflection on the proposed solution. As Nakakoji, Yamamoto *et al.* (1998) say, the languages-of-doing *feature a representational talkback, i.e. representations are able to reveal otherwise implicit features of a design*.

Different design environments such as KID, JANUS, VDDE or ART mentioned at the beginning of this section, exhibit some differences in the implementation and realisation of the individual modules that constitute a particular tool. Nonetheless, a generic core of most of the developed DODE-s typically contains at least the following modules:

(i) a high-level *input interface* featuring appropriate 'language of doing' for formulating initial requirements (e.g. KiD has a visual editor of floor plans and lists of available components);

(ii) a *catalogue* of components permitted to be used in design with a tool for the construction of potential solutions often using iconic representation of components (e.g. KiD works with kitchen floor plans, its 'components' are various types of stoves, sinks, fridges, etc.);

(iii) a *simulator* and analyser of current solutions for checking their compliance with requirements, guidelines and past cases, and critiquing designer's actions in that respect (e.g. with kitchen design there are constraints for left- and right-handed users that are checked automatically);

(iv) a *case-based* mechanism for the retrieval of past designs (in KiD, the past cases are text-based representations of layouts that can be efficiently 'translated' into a visual 'language of doing' and displayed on a screen);

(v) a tool for recording and browsing argumentative *justifications* of design decisions (e.g. designer may decide to violate and override a constraint of 'optimality', such a decision can be explained or otherwise justified)

The case studies from using DODE-s show that users often disagree with a proposed criticism coming from the design tool, e.g. because of 'non-optimal' or unusual positioning of a fridge and sink in the kitchen. However, this disagreement is not harmful; mainly because the different (and rather strong) 'opinion' usually triggers a deeper analysis of the reasons that led to the unusual decision. In many cases the disagreement may lead to formulating counter-arguments and preferences. Eventually, designers may become aware of new constraints, requirements or a mutual compatibility of the deployed components. The introduction of new requirements, constraints and preferences, or the modification of the existing ones can be perceived as a shift in the designer's perspective. Procedurally, it may be associated with Schön's operation of design *re-shaping*. Thus, an informed critique may act as an unexpected feature that can lead a designer to articulating another perspective or frame for attending to the problem.

Another important feature of design problem solving that is observed in some studies of DODE-s reveals a tentative nature of the approach human designers typically have when designing vaguely defined artefacts. The mechanisms underlying DODE-s (mainly the visual construction tool and the associated log of justifications) fit well into the vagueness of the design exercise. Mainly, because designers can test their ideas or preferences, as soon as they express them in the suitable 'language of doing'; e.g. as a sketch (Goel 1995). Furthermore, each modification to the visualised solution sketch is assessed by the tool's analyser for its compliance with the known domain constraints. Thus, the interplay of problem specifications and hands-on approach to solution generation as mentioned in section 4.2.2, is emerging from such a co-operation between the tool and the designer. The result is an 'inevitable' intertwining of the specifications and implementations (Swartout and Balzer 1982).

To summarise, DODE-s represent a very important research undertaking in the direction of human-centred design support. They manage to address successfully many issues associated with the cognitive investigation of design and designing. Among their key contributions, we can emphasise the accent given to the problem formulation and understanding, the co-evolution of problem specification and implementation, and the frequent usage of informal design rationale. More information on the topics of DODE touched in this section can be found in the research papers of L3D group from the University of Colorado (Fischer 1992; Fischer, Nakakoji *et al.* 1993; Nakakoji, Sumner *et al.* 1994). Similar findings are reported in (Swartout and Balzer 1982; Schön 1983; Buckhingham Shum 1996; Nidamarthi, Chakrabarti *et al.* 1997).

5.2 Model-based design support

Theoretical framework for the theory of recursive framing in design is the main theme of this thesis. It is proposed further in this document. Nonetheless, as we already mentioned in chapter 3, the development paradigm of our RFD model is knowledge-level modelling. We define the model with a help of circumscribed knowledge spaces for both the problem specification and solution construction. Let us therefore, investigate the roles played by the knowledge-level models in design. In the following sub-sections, we define these models and review their properties that in our opinion are relevant to the reflective design featuring the articulations and amendments of conceptual frames.

The review of the design tools presented in section 5.1 contained several references to various forms of modelling the designer's knowledge. Several different roles of the models were implicitly mentioned, and they are re-iterated explicitly in this section. This section therefore, looks at the models and their role in a non-trivial design more in depth. The focus is on knowledge-level models for the purposes of design, their basic principles, and characteristics that are desirable for design. Since this is a review chapter, we include also a short account of related research work where models on various levels play significant roles. Before reviewing two typical representative of knowledge-level modelling in design in section 5.2.2, we introduce knowledge-level models and ontologies in section 5.2.1.

5.2.1 Introduction to modelling on the knowledge level

Term 'knowledge level' was coined by Alan Newell (1982) in order to distinguish a higher level of information processing in addition to the lower, 'symbol level'. Both levels refer to different processes a knowledgeable agent is performing whilst solving a problem or doing a (intelligent) task. Symbol level is concerned mainly with a description of syntactic operations of an agent, which involve processing and manipulation of certain available 'symbols'. For instance, the number theory in the mathematics uses 'numbers' as the basic symbols and a few simple rules to solve the problems. On the other hand, knowledge level is concerned with *knowledge* that is used and created by an intelligently behaving agent. One point for the investigation on the knowledge level is what knowledge concepts are present or lacking when solving a task. Another, perhaps even more interesting question, is how available knowledge couples the actions of an agent to the task being solved. How does it relate the individual actions to each other, and to the environment in which they take place?

A careful reader may have already noted that the statements above do not make any commitment regarding the form the knowledge-level models may have, nor the internal representation they may be implemented in. As was already stated in section 3.2.1, the main advantage of this 'higher tier' in problem solving is that it abstracts knowledge as a *competence*, which is independent of the physical representation of symbols and concepts it manipulates (Newell and Simon 1976). This attitude towards knowledge acted as an important impetus for artificial intelligence research in general and knowledge-based systems in particular. Moreover, as it is the case with modelling in a general systems theory (see also section 3.2.3), one can build models that are closer or more distant approximation of the real world (Klir 1985). It is fully understandable that many different implementations of general knowledge-level models can be found throughout the history of AI (Stefik 1995). Some of the proposals that have a direct connection with design are briefly discussed later, in section 5.2.2. In this section, we concentrate on the features that make this knowledge-level account interesting for design as an intelligent problem solving activity.

Let us discuss several properties of the knowledge-level models that we see as beneficial for design. First, we return to the statement from Newell's definition at the beginning of this section. Namely, that it is an agent's knowledge of a specific domain that determines its actions during a problem-solving episode in a particular environment. Different actions of an agent that seem otherwise random and unrelated, have a sound common ground in the knowledge the agent has about the domain and uses in it. Thus, a knowledge-level model may be perceived as a kind of *indexing* or *referential framework*, to which the agent's actions are related. However, such an index is usually very complex, far richer than the indexes that are known, for instance from libraries. Typically, a knowledge-level model is not a linear structure. It is a richly interconnected network of the records that in addition to determining the agent's actions, defines also the 'meaning' of those actions. In other words, it is the agent's knowledge that gives to any individual action its proper place in problem solving. Thus, the knowledge endows the agent's actions with some *semantics* and *context*, in addition to a strict syntax.

Another aspect of knowledge-level models that distinguishes them from symbol-level implementations is that they can be shared and easily transferred among multiple implementations. This aspect is typically illustrated using an analogy with computers[6]. Hardware (incl. a processor, storage units, memory units) is unique for every computer – it is a symbolic representation framework. However, software (operating system, applications and data) is more flexible, and to the certain extent, can be shared between different computers. When we replace term 'computer' with a more generic term 'agent', it is possible to perceive knowledge-level models as that layer in the information processing, on which the collaboration between the agents, and *knowledge sharing* may be observed. In order to share knowledge, the agents must subscribe to a common expressive mechanism, so that they can use the same knowledge in a similar manner. They must share the semantics for interpreting the concepts from a transferred knowledge-level model. An example from our everyday life of a sharable knowledge-level model of an external world, commonly applied in practice by almost everyone is a human language developed as a means to communicate, i.e. to share, publish and acquire knowledge.

Having discussed the most important features and advantages that are brought into the game by the knowledge-level systems informally, we may now turn to definitions that are more formal. One particular approach to modelling on the knowledge level is using an entity known as a *common ontology*. Ontology as a term (and a scientific theory) has been known in philosophy for several centuries. It denotes a theory *that is concerned with the nature and relations of being*. This philosophical term was brought to the computer science and consequently 'AI in design' community by researchers from Stanford University during their research into knowledge interchange. Gruber (1993) understands ontology in the world of computational systems as '*an explicit specification of conceptual representation*'. He justifies such a view with pointing to the fact that '*in knowledge-based systems what exists is only what can be represented*'. Chandrasekaran, Josephson *et al.* (1999) add to Gruber's view that ontology is typically '*a representation vocabulary specialised to some domain or problem matter*'. An agent can then use such a vocabulary to express and represent its knowledge related to the actions performed in connection with a particular problem or task.

An ontology is thus an account and representation of an explicit portion of what is often referred to as a 'domain theory' or 'background knowledge'. An ontology may certainly be developed for many different purposes. For instance, for a consistent naming of symbols that are manipulated in a particular domain or for a definition of important relations among certain objects. Although ontology defines the conceptual objects and symbols of a particular domain, this definition is clearly on the knowledge level. An agent does not define the implementations and the individual bits and pieces of the objects,

[6] This is definitely, not the most fortunate metaphor (as many computer users would agree), but it shall suffice in making a point we want to argue.

but rather it assigns certain meanings to such objects. As it is defined, the only commitment made by ontology is regarding the interpretation of the objects and relations in a domain. Ideally, no commitment at all is made in respect to the agent's underlying internal representations. On the contrary, it is often the case that the commitments, which an agent makes using a particular ontology, may have several different internal (or symbol-level) representations. An ontology is thus inherently a knowledge interchange framework that has a competence of referring to different knowledge sources and representations through the same channel or 'interface'.

If multiple agents (including mixture of human and artificial) subscribe to the same ontological commitments in respect to a particular domain, these commitments serve as a *common* or *shared ontology*. Such an ontology, when shared among different agents, provides a vocabulary (similarly as a human language does) for the exchange of information. At the same time, ontology defines a vocabulary not only for expressing the simple terms from a domain, but also allows assertions about the world that are more complex. For instance, a chunk of data acquired in an empirical study of the world performed by an agent may be rather poor in respect to yielding an 'immediate' knowledge. Measured temperature of '39.5°C' may have different implications in different domains. In forecasting the weather, it may imply a hot day; in medicine, it confirms a serious fever of a patient. Thus, the full potential of a particular chunk of data is revealed only when it is related to other data and knowledge of a particular domain. In other words, it must be interpreted in the domain ontology; only then, further complex assertions may be made. Generally, the 'profit' from the information acquisition is greater for the interpreted (or ontologically grounded) 'knowledge' than it was for the 'standalone data'.

For a single subject matter several ontologies may be developed. Van Heijst, Schreiber *et al.* (1997) organise ontologies into hierarchical structures, and talk about 'upper-level' and 'lower-level' models. As they argue, some knowledge has often more generic nature than the other does. In some cases, knowledge may be used in many different problems; whereas in other cases it is very narrowly bound to a particular problem or task. An example often quoted throughout the knowledge modelling literature considers a 'diagnostics'. Accordingly, the main principles of diagnostics are invariant whether one works as a general practitioner in medicine, a computer support specialist, or a shop-floor technician. Terms as 'cause', 'effect' or 'location' are sufficiently generic to be expressed by an upper-level ontology of generic diagnostics tasks. These terms however, may be fine-tuned for the individual domains. For instance, doctors usually talk about 'symptoms' as a specific type of 'effects'; they distinguish several specific 'locations', such as 'skeletal apparatus' or 'digestive system'. The same terms specified for the domain of machine diagnostics may include e.g. 'power distribution apparatus', 'mechanical apparatus' or 'input/output system' as specific 'locations'. Such specific terms may be grouped into a 'lower-level' ontologies to emphasise their association with particular specific domains. The relation between a more abstract ontology and the specific ontologies is typically hierarchical.

A significant part of the research community distinguishes between domain- and task-oriented ontologies that can be easier to re-use and share than highly specific, problem-oriented ontologies (Motta 1997; Chandrasekaran, Josephson *et al.* 1998; Motta and Zdrahal 1998). At this point, we can comment on one often-cited characteristic of ontologies that was not presented in the definitions above – the ontology *re-usability*. In the above definitions, it was only mentioned that agents could share an ontology as a common knowledge-level model, similarly as a human language is shared among the different people. It must be noted that such a sharing can be understood differently according to two basic paradigms. First, a common ontology may be shared 'spatially'; i.e. there are multiple, 'physical' agents that use the same ontology to mediate and communicate their mutual knowledge exchange.

Alternatively, an ontology may be shared 'temporally'; i.e. the agent can use knowledge expressed using a certain ontological commitment made at a different time, usually in the past. In the most generic case, sharing may occur along both paradigms – one agent may use knowledge acquired and published by a different agent in the past if both of them commit to the same, shared ontology. Some researchers – e.g. Motta (1997), refer to the sharing on a spatial paradigm as (proper) 'knowledge sharing', in order to distinguish it from the temporal paradigm that is often called 'knowledge re-use'. Nonetheless, both terms essentially argue the same – an ontology can be *shared by* the multiple domains, agents, and problems, or it may be *re-used in* the different domains, tasks, or problems. From this reason, the condition of the re-usability of knowledge-level model was not particularly emphasised in the definitions above and in section 3.2.1.

5.2.2 Some applications of modelling in design

The main objective of this section is to extend the skeleton defined in section 5.2.1. Two applications of knowledge-level modelling are presented to illustrate the challenges investigated in respect to knowledge models. Similarly as it was the case with the review of the current trends in the design support (see section 5.1), the research mentioned in this section is not exclusive. It is a flavour of what kind of problems is investigated at the crossroads of knowledge-level modelling and engineering design. Due to a vast amount of work done, it is virtually impossible to review the whole research because of the sheer number of different approaches and specific applications. Therefore, we opted to give an overview of a couple of large-scale efforts that attempt to model a generic design process.

Instead of repeating what was discussed in the review of the design support tools (section 5.1), we focus on the additional details of the models in design and their relevance to the knowledge-level perspective. Typically, each tool or methodology from section 5.1 corresponds on the knowledge-level to an appropriate model of the design process. However, it should be noted that these 'procedural models' of design; i.e. mostly concerned with design as a task, are not the only applications of modelling techniques in design. The work reviewed below includes the research on the Generic Task Model of Design as a representative of a class of generalised procedure- or process-oriented application of knowledge-level modelling (section 5.2.2.1). To complement it, the research that formulated the General Design Theory is reviewed in section 5.2.2.2; this research is a representative of a more 'declarative' application of knowledge-level models in design.

5.2.2.1 Generic Task Model of Design

The key idea behind the efforts developing generic task models is the acknowledgement that there are often some problem-independent features typical for a class of tasks. These higher-level knowledge models were already mentioned in the previous section with diagnostics serving as an example. Regardless of whether a diagnosis is concerned with the medical complaints, shop-floor machine maintenance or pupils' performance, certain actions are repeated across the domains, and can be observed in all applications. Therefore, it seems to be reasonable to formalise this domain-independent pattern on an abstract level. This level is often referred to as a task or task-description level, and knowledge models developed on this particular level are in the AI community well known as *generic task models* (Tansley and Hayball 1993). From a large base of generic task models (or GTM) developed for various specialised tasks, we focus in this section on the GTM for design (GTMD). In particular, the review draws upon research into GTMD conducted by Frances Brazier and the researchers from the Dutch Vrije Universiteit in the past decade (Brazier, van Langen *et al.* 1996a; Brazier, van Langen *et al.* 1996b; Brazier, van Langen *et al.* 1998).

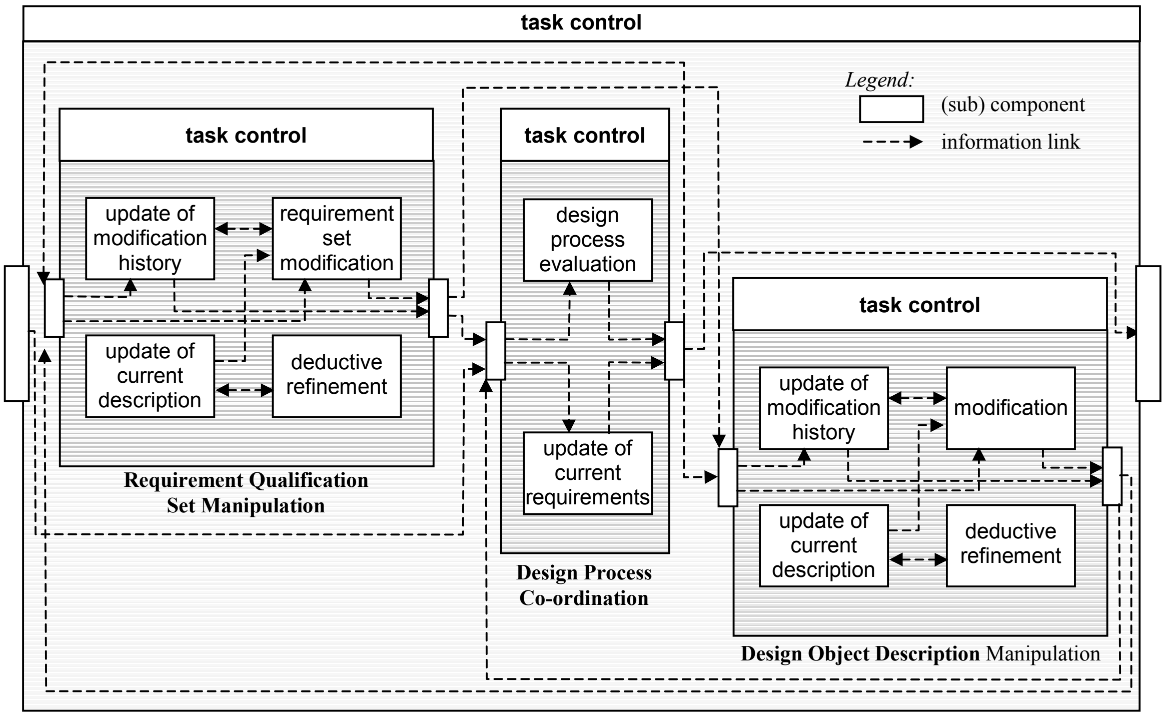

task control
Legend:
(sub) component
information link
task control
update of modification history
requirement set modification
update of current description
deductive refinement
Requirement Qualification Set Manipulation
task control
design process evaluation
update of current requirements
Design Process Co-ordination
task control
update of modification history
modification
update of current description
deductive refinement
Design Object Description Manipulation

Figure 5–3. Generic task model of design – structure (Brazier, van Langen *et al.* 1996b)

GTMD assumes that the knowledge of initial requirements is available at the beginning together with the knowledge of design objects and strategies. Requirements may be ordered using a preferential measure, some of them may be necessary, others only preferable. GTMD models the preferences in a module named '*requirement qualification set manipulation*' (see Figure 5–3). Design strategies known to the designer enable him or her to construct the designed artefact. The same artefact may be described from several different views – GTMD takes this variance in account in form of a specific module for these purposes. In Figure 5–3, this is named '*design object description manipulation*'. The description of a design object (artefact) is mostly incomplete, and it is stepwise extended during the design process, similarly as the set of requirement qualifications. The third module in the figure has the role to co-ordinate design process based on the evaluation of its current state. This module named '*design process co-ordination*' selects a suitable strategy from the knowledge base for further exploration of a design space. The selected strategy then has an influence on the initial requirement qualifications set, as well as descriptions of the design solutions.

As visible in Figure 5–3, each of the modules can be further decomposed into simpler operations or *tasks* that are performed during the design process. For instance, in case of requirement manipulation module, four simpler tasks are defined in the following order:

1. *(requirement set) modification* … this task determines which strategy is chosen for the selection of requirement qualifications, what preferences are applied to the selected relevant requirements, what level of strictness is applicable, and so on;
2. *deductive refinement* … this task infers what requirements are to be considered having decided on certain preferences earlier, and what possible modifications to the current set are admissible;
3. *update of current description* … in this step, the selected requirement modifications are enacted and incorporated into the current set of requirement descriptions;
4. *update of history* … this task keeps track of the development and stepwise refinement of design

A similar decomposition and the flow of information is defined for the design object description module; i.e. the module responsible for the development of design solutions. First, candidate objects for the modification are identified; then the actual changes and modifications are deduced. Next, the changes are incorporated into the existing description, and in order to keep a track of the changes, the history record is updated. The update processes for both modules are closely related; i.e. a change performed in one module may have an immediate or delayed impact on the other module.

Between these two modules that work with and manipulate the knowledge of design objects, there is the third module concerned with the knowledge of design actions (see also section 5.2.2.2). The main role of the co-ordinating module is to assess the current state of design, and drawing on this information suggest further amendments in the space of design requirements. In other words, it makes sense to continue with further manipulations of either requirements or solutions, if there is a benefit observable in the development history. For instance, the performance of the design solutions is improving in respect to a particular criterion (e.g. a price). Design process is progressing (not stuck in a deadlock), and there is still an unused potential in the currently active strategy.

Design task, as described and defined by the model in Figure 5–3, is sufficiently abstract to be considered ‘generic’. As such, it may be further adapted and refined for the purposes of specific applications or problems. For instance, Brazier, van Langen *et al.* (1996b) illustrate the deployment of GTMD for an elevator design in Sisyphus-2 domain, which is a well-known benchmarking case for the domain of knowledge modelling (Yost 1992). To account for the specifics of the elevator design, the generic sub-task of a design description (solution) modification is further refined into separate tasks for the analysis, determination of the modifications, and their implementation. The determination sub-task is further decomposed according to the prevailing method used for the problem solving – e.g. ‘extend-and-revise’ or ‘propose-and-revise’. Each of the phases is further detailed into actions that can be readily and directly executed by a problem solver.

The case study of an elevator design shows how a generic task model can be enhanced and refined, when we decide to use it for a specific problem and domain, and choose a suitable problem-solving method. The same case study and a similar breakdown of the knowledge models into task-oriented, domain-oriented, problem-oriented and method-oriented appears also in (Motta and Zdrahal 1998). According to these authors, the advantage of generic models is that they can be used independently of each other, and mutually combined almost without any restrictions. For instance, one may re-use a task model of 'propose-and-revise' problem-solving method without any commitment to the domain or problem-specific concepts. Of course, this statement has to be taken with a certain degree of scepticism because there are usually constraints as to the applicability of certain methods for particular classes of tasks or domains. However, the idea of such a wide and almost universal re-usability is very challenging, and it became a popular research topic in the past decade (Motta 1997).

GTM in general and GTMD in particular can be very useful in the design of artefacts of a certain type and class. If the various combinations of requirements, fixes, preferences, and parameters are well defined and known in advance, then GTM-s perform very well and very reliably. The fewer irregularities there are in a particular domain, the less work is required in order to tune a generic model into a working procedure. However, in the case when the fixes or preferences are ambiguous or incomplete, and the structure of a design process is changing case to case, it may require a lot of effort to adapt GTM to such an ill-structured problem. Nonetheless, it is still possible to use GTM as a kind of template for the rapid development of the core application, and extend the prototype instead of building the whole application from scratch.

Another useful feature of GTMD is its compositional and decomposable structure. This structure makes it even easier to re-use the modules and/or replace the generic tasks with finer-grained components, as it was the case with the elevator case study. Such modularity enhances the re-usability of the individual components, and any further refinements of GTMD can be seen as replacements of the generic templates/modules with more domain-, problem-, or method-specific 'plug-ins'.

5.2.2.2 General Design Theory

The origins of the General Design Theory (GDT) can be traced back to the late seventies and early eighties when it was presented first in Japan, later in the rest of the world. The primary source of information for this review is (Tomiyama 1994), where the promises made by the theory when it was launched are evaluated. This concise account of GDT defines it as a knowledge-level theory that is interested with the objects that are conceptually manipulated during the design process. GDT regards design as a mapping from the function space to an attribute space, and associates each object (entity) from the world to an abstract entity concept. The theory assumes that all entities in the world can be described by a finite set of known attributes, and that each 'real' entity corresponds exactly to one abstract entity concept. This type of theory is however, rather idealistic and assumes that everything that needs to be known about a design problem at hand is actually known.

As mentioned in section 4.2.1, the assumption of a closed world, in which all objects are well defined, is useful for tackling the incompleteness and variability of the world. Nonetheless, if this were the case and such 'complete' knowledge base could be constructed for design, it would deny some of the essential characteristics such as exploration and interpretation. The design would be about straightforward retrieval of the solutions and one-to-one mapping from a complete functional specification to a unique (and the best possible) solution.

In the real world, design is a stepwise refinement of a design solution and underlying constraints. In order to comply with reality, GDT refers to physical laws as a boundary that constrains the compatibility and applicability of various entities and entity concepts. The iterative development of design solutions to the incompletely described problems is tackled in GDT by the introduction of a concept of '*meta-model*'. This concept represents a more realistic development of design solutions rather than their simple retrieval. It is defined as a finite sub-set of those attributes and values that comply with the physical laws of the 'real world'. In other words, a meta-model as understood by GDT, is a physically achievable approximation of a design solution. The meta-model is constructed to consider only the objects not contradicting the given constraints. In this aspect, it is reminiscent of the constraint propagation techniques known in the constraint satisfaction (CSP) domain (Kumar 1992). As new attributes are added to the meta-model to satisfy the constraints, the model becomes increasingly more specific, and finally evolves into a final design solution.

GDT and its implementations distinguish between two kinds of knowledge in design. First, it is the knowledge of design objects, and second, it is the knowledge of design actions. At the object level there are several logical operations, such as deduction and abduction, complemented by an extension to the standard logic – circumscription (McCarthy 1980). The operation of circumscription is the interesting one in this triad because it is responsible for the reformation of knowledge in order to resolve a violation of a constraint. However, this seems to be at the same time also a major gap of the GDT model.

As argued in section 4.2.1, the reason is that the circumscription, similarly to the 'closed world assumption', only restricts the set of facts. The restriction follows the rule that *the objects mentioned as having certain property (e.g. compliance with a constraint) are the only objects that are actually needed to satisfy it (a law or a constraint)*. Thus, circumscription in GDT can restrict the choice of the attributes and their values to restore the consistency of the solution model with the constraint. Such an approach is more about knowledge refinement than any 're-formulation'. Consequently, GDT is unable to come up with a proposal of a solution extension that may violate some of the constraints of the physical laws. Thus, the concepts of Schön's surprising behaviour or Altshuller's contradiction are avoided by circumscribing the problem solving theory.

Circumscription narrows the scope of suitable designs by assuming that all the relevant facts were taken into account when performing this operation on the object level. If new knowledge or a fact becomes available, they may easily return the design to its early phases because they may 'cancel' the circumscribed inferences[7]. Circumscription is a powerful mechanism bringing common sense into pure logic; nevertheless, it does not have generative powers to convey any new knowledge to go beyond the problematic statement. It is a vehicle for conjecturing, simplifying and constraining the incompletely described discourse. Nonetheless, circumscription may play a significant role in shaping or framing the design problem, as it was mentioned in section 4.2.1.

Back to GDT and its other level of knowledge – knowledge of design actions. According to the theory, this is a purely deductive mechanism comprising a set of rules determining what to do next on the object level. However, apart from the heuristic prediction, GDT says very little about the fact that the 'action knowledge' may be occasionally tacit or inarticulate. Designers may approach the problems

[7] This is well visible in the 'interpretation' of circumscription into plain language that is saying that the objects mentioned are the only ones needed, '*unless there is something wrong with them*' (McCarthy 1980).

opportunistically and decide on their strategies and next steps on-the-fly. Their 'strategic' knowledge (Candy 1998) may involve rather complex reflective processes, and resist attempts to model it as 'pure deduction'. Yet GDT has many progressive features that comply with the recent cognitive studies of a design practice (see also section 4.1.3). It supports the work in multiple design contexts and maintains the consistency within a context through a truth maintenance system. It acknowledges that the real-world design is a process of solution refinement that rarely begins with a set of complete initial requirements. Finally, it agrees that design is neither synthesis nor analysis (abduction/deduction), but rather a combination of both operations.

On the other hand, what is missing in the theory from our point of view is some kind of 'talkback' from the space of solutions back to the space of functions and requirements. The only form of feedback GDT models, is the violation of constraints by a proposed solution model typically followed by the circumscription. The entire model, though being iterative, follows one direction. It is a kind of 'spiral' rather than 'oscillation' from the functions towards the objects, attributes and values, and vice-versa. The feedback is limited to the cases of explicit inconsistency of a proposal with the physical laws and constraints (i.e. physical inadmissibility).

GDT acknowledges that new knowledge may be articulated by a designer, but does not actually attend to this articulation. It does not propose any models associating the formulation of new constraints with any reflection or tacit assessment. The designer is a 'black box' doing the evaluation, and proposing all modifications to the task specification, and it is not the aim of theory to investigate the details of such reasoning. The role of the theory is to investigate the relevant space of the possible designs, make sure that they are physically admissible, and maintain the logically consistency in the development of the solution. Tomiyama (1994) describes both the original GDT as well as one of its extensions; other extensions can be found e.g. in (Takeda and Nishida 1994).

5.3 Summary or a role of models in design

As we already emphasised several times in this document, one of the major problems with design is its initial under-specification and incompleteness. In addition to the development of a design solution, it is also necessary to refine and enhance one's understanding of a particular design problem at hand. Chapter 4 identified the operations of shaping and re-shaping (or framing/re-framing) as the fundamental techniques for tackling the ill-structured design problems. The formulation of conceptual frames for a particular problem helps to see the current problem *as* the previous one (Schön 1983). Such a conceptual commitment necessarily includes the usage of the same or (ontologically) similar conceptual vocabulary for the problem interpretation.

In terms of what was briefly discussed in this chapter, we may say that the ability of 'seeing as' involves certain ontological commitments shared between the two design problems. Certain knowledge-level accounts can be transferred from the previous design case to the current one. In line with the definitions presented in section 3.2.3, it is possible to argue that in respect to those particular commitments, the previous case is actually, *a model* for the current design problem. The previous case with its (already well defined) conceptual vocabularies, design specifications as well as design solutions, is easier to work with than the incomplete current case. The previous problem might not be simpler or less difficult in terms of structural or behavioural complexity. Nonetheless, it is better defined, and provides 'means' for tackling the incomplete, current design situation. Using these means, we can approach the current problem in a more 'friendly' and familiar fashion.

Thus, we believe that the role of knowledge-level models in design is far greater for the problem specification (and its evolution) than it is presented in the reviewed literature. In order to use a full potential of the above-mentioned 'conceptual re-use', it must be ensured that the terminology used in the previous and current design cases is not ambiguous. In other words, it is desirable to use different, problem-specific languages for the interpretation of the individual design problems (Fischer 1992; Goel 1995). Nevertheless, to make sharing feasible, it is also desirable to *cross-reference* these interpretations to a common shared index. Such an index may also have a form of a knowledge-level model that may contain certain problem-independent commitments. An example of such an index is for instance, the ontology of components, substances and actions (Bylander and Chandrasekaran 1985), physical quantities and laws (Borst 1997; Wevers 1999), and similarly.

In order to summarise the important facets, the following features of the knowledge-level modelling approach will be detailed later. In chapter 10 the theory of iterative design using framing is elaborated into details. The foundations of the theory are a leading topic of the chapter 6.

- Previous designs can serve as knowledge-level models for the current problems.
- Conceptual commitments are re-used from the previous cases in order to see the current problem as a previous one.
- To avoid ambiguity, a more general, problem-independent knowledge-level model may be needed to serve as a cross-referencing index.

5.4 Gaps between the theory and practice

As we could see in chapter 4, different perspectives for the investigation of a design task have to tackle the issue of uncertainty, vagueness, incompleteness, and ambiguity. As is visible from the review, the positions on these issues are split, and there are often contradictory views on the subject. In our opinion, this controversy is mainly because most views do not attend to design holistically. This section summarises the essential pieces that are missing in the design process description, and it uses the characteristics from section 4.3 to make cross-referencing easier.

First, we ascribed an important role to the exploration in design. The aim of exploration is to not only come up with a solution but also improve and refine the understanding of the structure of the design space and the design situation. Exploration is a technique coping with the under-specified situation, and it may use the unstructured, intangible knowledge that takes form of various assumptions, focuses, or contexts. In the literature, we can encounter numerous references to a designer's experience, which forms a basis for an informed exploration and interpretation of potential design solutions. Nevertheless, despite the fact that many researchers accept the importance of the problem interpretation, there is surprisingly little work done on what is the place of the tacit, inarticulate, or 'hard-articulate' knowledge in modelling the problem interpretation.

Similar issues arise when we investigate design from the operational perspective. As far as we restrict our attention to the development of the design solutions, it is sufficient to assess the admissibility of the solutions against the given requirements and constraints. Taking into account the fact that requirements come to the designers as ‘given’, things seem to be clear and straightforward. Nevertheless, such assessment is very difficult when we intertwine the synthesis of design solutions with their analyses in order to tackle the initial incompleteness. What if we want to change the current problem specification in order to re-formulate the design task and test a particular interpretation or perspective? How is the success or failure of such an amendment assessed?

From what we found in the reviewed literature, the designers often use their knowledge and experience to help them re-formulate the current problem interpretations and perspectives. However, the conceptual and operational apparatus does not formally incorporate the empirically observed facts. There is also an agreement about the existence of some complementary knowledge that may be used for the re-interpretation. The assessment of the earlier-mentioned exploratory probes and re-formulations is ‘opportunistic’ rather than ‘random and intractable’. There is an empirical evidence that in addition to assessing an explicit admissibility of the proposals and probes, there is also a kind of check on the ‘tacit acceptability’ of the proposed probes. There seems to be also empirical evidence of the different types of such ‘exploratory probes’ that may be used for the problem re-interpretations. Such probes may be difficult to be expressed formally or explicitly, but this does not mean that they do not exist and do not deserve to be incorporated into formal models of design.

Cognitive science seems to be most advanced in the perception of the fact that design and other ill-structured problems require a lot of insight from the designers. Deep insight is definitely an activity that points to the ability of professionals to perceive the problems more freely, and thus discover interesting features. Note this is not to say that those features did not exist before. Those features need not be ‘invented’, but rather ‘uncovered’ or most often – ‘*brought into focus*’! Professional designer may become aware of features and concepts that are often overlooked in the familiar approach to the design situation. Some authors distinguish among ‘deep’ and ‘shallow’ knowledge, and claim that it is the deep knowledge that makes the difference between an expert and novice (Price and Lee 1988; Boyle 1989). They argue that this knowledge is also responsible for informing the strategic decisions; i.e. how to continue with the design (Brazier, van Langen *et al.* 1998; Candy 1998).

However, there is little formal account of how an ‘insight’ or ‘deeper knowledge’ work, and what constitutes these cognitive actions of the level of knowledge. Insightful acts are immensely difficult to capture in formal representations, but this difficulty with the representation should not deter us from investigating the ‘insightful’ decisions formally. Furthermore, it is an interesting challenge to locate the phases or stages in the design process when the ‘insight’ typically emerges.

As research on the formalisation of the creative design shows, it is often the ability to draw unusual analogies both in and across domains that may overcome the contradictions emerging in a particular perspectives. The process revealing such contradictions and articulating them in the explicit terms of the current or amended problem solving theory helps understand the ‘insight’. Subscribing to the view that insight addresses the hidden or tacit contradictions within the existing approach to designing a product seems to be a feasible model of this complex cognitive activity. An innovative or creative solution rarely emerges from a conventional understanding of the problem situation. In order to become aware of a contradiction in the existing problem interpretation and propose an unusual solution, the design problem itself may need to be perceived 'differently' – in the different conceptual terms.

The mentioned ability of practitioners to 'see as' and 'do as' is often associated with the their talent. However, we argue that this ability is manifested in the numerous repetitions trying to shape and re-shape one's understanding of the design situation. Designers seem to impose a kind of '*frame*' through which they attend to an ill-structured and under-specified problem and design situation. Thus, it is the process of *framing* and *re-framing* that deserves more attention of the research community. Especially, a formal account and a model of the design process incorporating the articulation of conceptual design frames is needed to provide a broader, knowledge-level description of the exercise of designing.

The essential reason for the excitement about this research topic is the fact that the operation of framing seems to be very close related to the research of knowledge-level modelling. One may see a conceptual design frame as a kind of vocabulary that is used for the explicit articulation of one's understanding of the new, under-specified problem. Typically, such vocabulary would be based on the vocabularies of the previous design tasks that are familiar and much better structured. What knowledge do the experienced designers use to recognise and activate these analogies? Isn't the recognition of a frame similar to the tacit assessment of the acceptability of a particular 'exploratory probe'?

In spite of the fact that such knowledge for triggering analogies may be very difficult to express, there is still a significant chance for improving our understanding of the process. Although it may not be possible to capture and represent the inarticulate knowledge in the explicit terms, it is interesting to look at how this knowledge is activated and applied. If we can find the triggering mechanisms for shifting a designer's attention to a certain feature, property or structure of the designed artefact, this may be seen as the incorporation of the tacit knowledge into formal models of design.

In the rest of this document, we define the concept of 'design frame' formally in the terms of knowledge being used and manipulated. We associate this term closely with the mutual interaction and co-evolution of the problem specification and the construction of a solution. We show theoretically that these activities are closely intertwined and mutually circumscribe each other's knowledge. In chapter 6, the initial version *recursive model of framing in design* is introduced and discussed.

The theoretical contribution is supported by an empirical analysis of the design experiments in terms of the concepts introduced by RFD model. The experimental work is introduced in chapter 7, with further details on analysis in chapter 8. The conclusions from the experimental work and their implications for the refinement and extension of the initial RFD model are then presented in chapter 9. The initial RFD is eventually refined and defined in deeper details in chapter 10, thus implementing the 'reflective talkback' from the analysed and evaluated empirical study (~ exploratory probe).

6 THEORY OF FRAMING IN DESIGN

In the previous chapters, we referred several times to such terms as ‘design frame’, or alternatively, the operation of ‘design framing’. These concepts are not new in the design research community, as can be seen for example, in the theoretical works of Schön (1983) who talked about ‘shaping’ of the design situation. Similar concepts appear also in the application-oriented papers, e.g. those from Smithers, Conkie *et al.* (1990), Fischer (1992), or Kolodner and Wills (1996) who talk about ‘contexts’, ‘views’ or ‘perspectives’. Nevertheless, the major gap in the current design research is an interpretation and a formal clarification of terms ‘*frame*’ and ‘*framing*’. We agree with the arguments presented in the earlier reviewed papers that framing is an important operation that precedes problem solving, and complements it. But what are the implications of ‘framing’ on the knowledge level?

Another important motivation for the present research has its origins in the existing work of the software engineering community (Fischer 1992; Nuseibeh and Easterbrook 2000). The recognition of the conceptual duality between design specification and design solution, and their equal importance have been acknowledged already by Swartout and Balzer (1982). However, the actual relationship of these two distinct conceptual spaces (‘worlds’) remains very little investigated (Nuseibeh 2001). What is actually happening with the designer’s knowledge during the framing operation? Can this ‘woolly’ operation be expressed in a formal or semi-formal language?

6.1 Essential definitions

The closest attempt in the available literature at formally addressing the issue of ‘design problem shaping’ can be found in (Nakakoji, Sumner *et al.* 1994). These authors define something called ‘design perspective’ in the following terms: *a point of view, which implies that certain design goals exist, certain bodies of design knowledge are relevant, and certain solution forms are preferred*. Nakakoji *et al.* use the term ‘design perspective’ mainly for expressing the designer’s intentions; i.e. design perspective is a kind of vocabulary of concepts used in the problem solving phase.

We shall use a concept of ‘design frame’ or ‘frame’ to represent the role in circumscribing the real-world design problem. Despite a literal similarity with Michael Jackson’s (2001) terminology of ‘problem frames’, our definition of ‘design frame’ and ‘framing’ approaches the issue from a different angle. First, we look more closely at the interaction between the problem solving and problem framing. Second, our notion of ‘design frame’ is more conceptual than Jackson’s more operational view. Third, we approach problem framing on the knowledge level. We are specifically interested in how do the frames change, how do new concepts come into the current frame and why does a frame shift occur at all at a particular stage of a design process.

6.1.1 Basic concepts for model of framing in design

Let us begin from a broader perspective. It seems to be a redundant statement to say that designers solve design problems. Is there really a redundancy? Assume a designer is tackling a design problem; let us denote this design problem as $\mathfrak{DP}$. This problem might be vague or ill structured. In order to solve design problem $\mathfrak{DP}$, the designer has to characterise it by an explicit selection and application of suitable elements from a space of potentially applicable problem specification primitives $\mathcal{S}$. This ***explicit problem specification***, denoted as $\boldsymbol{S} \subseteq \mathcal{S}^*$, provides a context for applicable design methods and domain theories. These methods and theories are tools enabling the designer to satisfy the explicit specification. However, since the problem specification $\boldsymbol{S}$ is only an interpretation of $\mathfrak{DP}$, the solution satisfying $\boldsymbol{S}$ must not necessarily be a solution to $\mathfrak{DP}$. This important distinction is discussed more thoroughly later. Now, let us develop the theory of framing step by step.

In order to solve the design problem $\mathfrak{DP}$, a designer associates it with a suitable specification space $\boldsymbol{S}$, and tackles the problem specified (~ interpreted) in this fashion. In other words, designer attempts to specify a vague design problem $\mathfrak{DP}$ in terms of some 'statements' (i.e. $\boldsymbol{S}$). These can be formulated and instantiated from a set of selected conceptual primitives ($\mathcal{S}$). In terms of logical theories, we may say that a designer *circumscribes* (McCarthy 1980; 1986) the design problem by declaring that only the statements from the explicit specification $\boldsymbol{S}$ are needed for solving the problem[8].

Such assertion, however, can be only made within certain conceptual boundaries – a conceptual ***design frame*** $\boldsymbol{\Phi}$. Design frame $\boldsymbol{\Phi}$ can be defined as a pair of two circumscribed knowledge spaces that are constructed from the relevant problem specification primitives $\mathcal{S}$ and a space of relevant problem conceptualisations $\mathcal{T}$. Thus, 'framing a design problem' means articulating a set of conceptual objects $\mathcal{T}$ that may be used for doing the design, as specifiable using concepts from $\mathcal{S}$ (relevant problem specification primitives). Formally, a design frame denoted by symbol $\boldsymbol{\Phi}$, can be expressed as follows: $\boldsymbol{\Phi} = \langle \mathcal{T}, \mathcal{S} \rangle$.

Let us introduce and define two additional symbols $\mathcal{S}^*$ and $\mathcal{T}^*$ for a formal description of the above-mentioned circumscribed knowledge spaces. First, we define $\mathcal{T}^*$ as a closure that is constituted by the selected conceptualisation $\mathcal{T}$ and an appropriate domain theory $\boldsymbol{DT}$. Domain theory $\boldsymbol{DT}$ is problem independent knowledge in the sense that it may be applicable to many different problems. Consider, for example, physics of rigid bodies. It is a domain theory applicable for a design of an elevator as well as spacecraft. However, for the differently conceptualised problems, different parts of the domain are applicable. A generic domain theory $\boldsymbol{DT}$ needs to be 'instantiated' for a particular conceptual base $\mathcal{T}$, in order to obtain a usable theory for solving the design problem $\mathfrak{DP}$. Let us therefore, refer to the closure $\mathcal{T}^*$ as a ***problem solving theory***.

Similarly, we define closure $\mathcal{S}^*$ as a hypothetical instantiation of the potentially relevant problem specification statements that can be articulated in the chosen conceptualisation of the problem $\mathcal{T}$ and selected domain theory $\boldsymbol{DT}$. Note that the knowledge spaces labelled as $\mathcal{T}^*$ and $\mathcal{S}^*$ may be defined as closures and instantiations of finite sets of conceptual primitives but they may not be enumerated in explicit terms or completely formalised!

Next, let us assign formal meanings to the terms used in the above definitions and interpret them in terms of reviewed work (see chapters 4 and 5):

a) A space of problem conceptualisations $\mathcal{T}$ corresponds to an ontology, a vocabulary of basic concepts, which the designer decides are available for the purpose of expressing a statement about a particular design problem and its solutions. This conceptual set typically includes the

[8] This is indeed a special kind of circumscription whose purpose is to 'close' designer's understanding of an incompletely defined (i.e. ill-structured) problem.

terminology for the definition of functional and structural objects. It may allow certain problem-specific mappings between the functions and structures, e.g. various types of behaviours, actions or relations (Bylander and Chandrasekaran 1985; Qian and Gero 1996).

b) An applicable domain theory $\boldsymbol{DT}_{\mathcal{T}}$ may be seen as a shared ontology, a generic vocabulary that defines the background (Wielinga, Akkermans *et al.* 1995) against which any conceptualisation may be tested. Domain theory per se is too generic and abstract to be of any direct use. In order to derive a potentially useful problem solving theory, it must be connected to and 'instantiated' by a particular problem conceptualisation (see set $\mathcal{T}$ in point a.).

c) The two spaces described in points a) and b) define what knowledge is available for the problem solving exercise for a particular design situation. Alternatively, we may say that the conceptual basis of the problem (i.e. $\mathcal{T}$) must be interpreted in the light of a relevant domain theory (i.e. $\boldsymbol{DT}_{\mathcal{T}}$) in order to become a usable theory for solving a particular design problem.

d) Similarly as set $\mathcal{T}$, set $\mathcal{S}$ is a conceptual ontology, a vocabulary; this time describing the basic terms and types of relations that the designer decides to use for the problem specification. It may be defined as combined task/method ontology, and at its simplest, it may contain concepts such as 'constraint' and 'requirement'. Some design problems can cope with these two 'types' of statement, others may need more – e.g. 'fixes', 'assumptions', 'preferences', etc.

e) A hypothetical space of potentially relevant problem specifications $\mathcal{S}^*$ that can be articulated in the particular conceptual frame is defined to complement the problem solving theory $\mathcal{T}^*$. Its principal purpose is to contain the explicitly expressible desires or intentions a designer may have in connection to a particular problem. It can be seen as a set of statements that can be formulated using the elements of a particular problem solving theory.

f) A hypothetical set of problem specifications $\mathcal{S}^*$ is an 'instantiation' of the conceptually allowed 'specification primitives' $\mathcal{S}$ by the terms known in a particular problem solving theory $\mathcal{T}^*$. It is clear that an introduction of a new 'specification primitive' such as 'assumption' may change how the terms in $\mathcal{T}^*$ are used, and what methods and operations are carried out to find a candidate design solution $\boldsymbol{T} \subseteq \mathcal{T}^*$.

As it was already mentioned earlier, a design frame does not exist 'per se'. On the contrary, it must be constructed 'on the fly' using the information that is believed to be relevant to the current design problem $\mathfrak{DP}$, and that is available as an initial design brief. Typically, a designer will use the customer's design brief to decide on the initial problem specification (***S***), and to identify similar looking design situations he or she is familiar with. Thus, a design frame may be seen as a special form of a relation of similarity with a past design case or a set of cases. This is a desirable feature because it corresponds with the empirical findings by Schön (1983) of designers 'seeing one problem as another one' (see also section 4.2.2).

To re-state the essence of the model of framing, design frames are subjectively constructed interpretations of a given, potentially vague and ill-structured design problem $\mathfrak{DP}$. Their role is to narrow down the vague space of 'all possible' interpretations to one that is better structured. Such a structured perspective consequently informs the selection of applicable domain theories, design methods, tools and so on. However, the design frame as it is defined above, is a conceptual and rather abstract boundary for tackling the design problem. In order to deploy this conceptual boundary in any meaningful way, it is important to relate the individual concepts and sets defined in this section. Let us therefore, extend the essential conceptual definitions by defining the basic relations that can be described among these abstract entities.

6.1.2 Predicates for the model of framing in design

As we mentioned in the previous section, design starts with an articulation of a conceptual frame $\boldsymbol{\Phi}$ that informs both problem specification and solution construction. A conceptual frame serves as an explicitly defined space, in which a design problem $\mathfrak{DP}$ is interpreted, i.e. specified and later solved. A designer has several knowledge sources available for performing an operation of 'framing' on the design problem. First is a design brief typically provided by a customer, and second is the designer's previous knowledge and experience. Using these two sources the designer aims to specify the vague problem $\mathfrak{DP}$ in a few explicit statements noting the essential features that are desired from the solution. Formally, this 'operation' can be represented by an explicit assertion of a relation expressing the fact that a certain set of statements $\boldsymbol{S}$ '*specifies*' the problem $\mathfrak{DP}$. Using terminology from section 6.1.1, such an assertion is shown in Eq. 6–1, below.

$$\exists S \subseteq \mathcal{S}^{*}: \textit{specifies}_{\Phi} (S, \mathcal{DP}) \qquad \textbf{Eq. 6–1}$$

Next, we would like to argue that the conceptual design continues with an attempt to formulate a minimal sub-set $\boldsymbol{T} \subseteq \mathcal{T}^{*}$ that satisfies a given problem specification. However, before any 'solution' can be proposed, the other 'half' of the conceptual frame – sets $\mathcal{T}$ and $\mathcal{T}^{*}$, needs to be articulated explicitly in order to have a 'vocabulary' for the construction of the potential solutions. The conceptual vocabulary is decided on by the designer taking into account the initial problem specification $\boldsymbol{S}$ (see the assertion in Eq. 6–1). This vocabulary is chosen so that it may serve as a problem solving theory to the given specification of the problem. A 'given problem specification' is in the scope of our theory understood as the set $\boldsymbol{S} \subseteq \mathcal{S}^{*}$, and defined as a formulation of the explicit design requirements and/or constraints (see Eq. 6–1). Thus, $\boldsymbol{S}$ will always denote only the statements specifying a design problem $\mathfrak{DP}$, to which a designer made a specific and explicit commitment.

Having broadly defined the conceptual foundation $\mathcal{T}$ and having chosen an appropriate (usually familiar) domain theory $\boldsymbol{DT}$, the designer may move to the next step. This next step comprises the articulation of a problem solving theory $\mathcal{T}^{*}$ from which a potential solution can be constructed. Thus, the vague problem $\mathfrak{DP}$ has been interpreted in the explicit terms of a particular problem specification $\boldsymbol{S} \subseteq \mathcal{S}^{*}$ and problem solving theory $\mathcal{T}^{*}$. Now, a minimal sub-set of the problem solving theory is constructed to deliver the explicitly specified desires. In other words, the designer tries to shrink the space defined by a problem solving theory $\mathcal{T}^{*}$ into a manageable chunk that can be manipulated, explored or searched. This manageable chunk can be associated with the term 'solution model' mentioned in several reviewed sources, e.g. (Altshuller 1984; Gero 1990). Due to its generative nature, however, we shall refer to it as a 'problem solving model'.

Formally, we propose to define a ***problem solving model*** as a minimal sub-set of the problem solving theory that *satisfies* the explicit problem specification. The relation '*satisfies*' would be binary, because it associates a problem solving model $\boldsymbol{T}$ with the current explicit problem specification $\boldsymbol{S}$ (with the conceptual frame $\boldsymbol{\Phi}$ as a contextual parameter). This relation is formally expressed by Eq. 6–2. The existence of a problem solving model $\boldsymbol{T}$ within the problem solving theory $\mathcal{T}^*$ 'proves' that the commitment to a particular conceptualisation $\mathcal{T}$ was 'correct'.

$$\exists T \subseteq \mathcal{T}^*: \textit{satisfies}_\Phi\,(T, S) \wedge (\neg\exists Y \subset T: \textit{satisfies}_\Phi\,(Y, S)) \qquad \textbf{Eq. 6–2}$$

One possible definition of the relation '*satisfies*' may take into account the semantic distinctions in the explicit problem specification. From this perspective, it is possible to distinguish the design requirements $\boldsymbol{R}$ from the design constraints $\boldsymbol{C}$, and proclaim that the problem specification is the union of the two – i.e. $S = R \cup C$ (Wielinga, Akkermans *et al.* 1995). Requirements would be those statements demanding the explicit presence of a particular feature, whereas constraints are typically the conditions that must not be explicitly violated by a design solution. More on this conceptual distinction appears in section 6.4.4. Furthermore, we devote the entire section 6.4 to an illustration of a typical operational model of the relation '*satisfies*'. At this stage, a simplified version introduced in Eq. 6–3 will suffice to formulate one operational definition of the relation '*satisfies*' (see below):

$$\textit{satisfies}_\Phi\,(T, S) \Leftrightarrow \{(S = R \cup C) \Rightarrow T \models R \wedge \neg(T, C \vdash \bot)\} \qquad \textbf{Eq. 6–3}$$

In Eq. 6–3, the symbols used have their usual meanings (Levin 1974; Mendelson 1979; Genesereth and Nilsson 1987). Symbol '$\models$' stands for a semantic entailment, '$\vdash$' denotes a proof-logical implication, and '$\bot$' stands for an 'empty' formula (i.e. a contradiction). The reading of this definition would be as follows: "Theory $\boldsymbol{T}$ is a problem solving model in respect to a given explicit problem specification $\boldsymbol{S}$ and a design frame $\boldsymbol{\Phi}$ if it is complete in respect to the required features ($\forall r \in R: T \models r$), and simultaneously it is admissible in respect to constraining conditions ($\neg\exists c \in C: T, c \vdash \bot$)." In other words, the problem solving theory must have a potential to deliver all desired features without any contradictions with the constraints being derivable. Further details of how an admissibility of any particular candidate solution may be assessed are provided in section 6.4. Note that the admissibility and completeness are both explicit categories forming an integral part of the computation of the relation '*satisfies*'.

We re-iterate that the explicit problem specification $\boldsymbol{S}$ is only an association with a design problem $\mathfrak{DP}$ – its clear and explicit interpretation, which is used for the purposes of problem solving. Thus, we argue that the existence of a problem solving model $\boldsymbol{T}$, for which the relation of explicit satisfiability (i.e. '*satisfies(T, S)*') holds, is a necessary but not sufficient condition of declaring $\boldsymbol{T}$ a 'design solution'. In addition to the satisfaction of the explicit problem specification, the discovered problem solving model $\boldsymbol{T}$ must be also '*acceptable*' as a solution addressing the original problem $\mathfrak{DP}$!

The irony is that it is usually not possible to define such a relation as '$\textit{acceptable}_{\mathfrak{DP}}\,(T)$' in explicit terms and in advance. This difficulty is mainly due to the fact that acceptability is often appreciated subjectively and tacitly. Often it simply cannot be completely expressed in the same languages as the conceptual sets $\mathcal{S}$ or $\mathcal{T}$ forming the design frame. Nevertheless, such a relation may be defined as a residual category. The formula in Eq. 6–4 is of great importance for the understanding of the residual relation of *solution acceptability*. What does it mean that '$\textit{acceptable}_{\mathfrak{DP}}\,(T)$' is a residual relation, in this particular situation?

$$satisfies_{\Phi}(T, S) \wedge \neg acceptable_{\mathfrak{DP}}(T) \mid \Phi \Rightarrow \neg specifies_{\Phi}(S, \mathfrak{DP}) \qquad \textbf{Eq. 6–4}$$

Well, we argue that it means exactly the same, as the statement made in section **3.1** saying that certain tacit decisions cannot be stripped of their contextual background. It may be difficult to define exact conditions of 'acceptability' in advance and evaluate them, but every designer may proclaim a certain problem solving model acceptable or not when s/he sees it and reflects on it. The tacit decision of acceptability makes sense only in a particular context such as defined by Eq. 6–4. In the formula, the context is provided by a design frame $\boldsymbol{\Phi} = \langle \mathcal{T}, \mathcal{S} \rangle$ and simultaneously $\boldsymbol{S} \subset \mathcal{S}^*$ (where $\mathcal{S}^*$ is 'a closure' over a set of conceptual problem specifications $\mathcal{S}$).

Eq. 6–4 can be interpreted so that whenever an otherwise sound problem solving model $\boldsymbol{T}$ is not accepted by the designer as a design solution, it may point to an incorrect (~ incomplete) specification of the actual design problem. Thus, a designer may not be able to articulate a precise criterion that is violated; only that s/he does not 'like it'. The candidate solution may deliver unexpected results or exhibit undesired behaviour. It means that the initial specification of the design problem $\mathfrak{DP}$ in the explicit terms of $\boldsymbol{S}$ was incomplete or incorrect. We may also say that the interpretation of the design problem $\mathfrak{DP}$ in the explicit terms of specification $\boldsymbol{S}$ does not fully reflect the real design problem $\mathfrak{DP}$. To address this tacit need for change, a frame amendment may occur to re-interpret the problem.

Let us now define the core of our theory – a recursive repetition of problem framing, acceptability assessment and problem *re-framing*. Our model is defined in terms of the sequence and 'control flow' among the conceptual predicates defined earlier, and is discussed in the following section.

6.2 Recursive model of framing in design

In the following paragraphs, we propose a recursive process model for design[9] that takes into account the theory of design frames introduced in section 6.1. The building blocks of the RFD model are the conceptual entities identified in the previous section, in particular:

a) a problem solving *theory* $\mathcal{T}^*$,

b) an explicit problem *specification* $\boldsymbol{S}$,

c) a problem solving *model* $\boldsymbol{T}$, and

d) a conceptual design *frame* $\boldsymbol{\Phi}$

The model is defined as a sequence of decisions based on the validity of the three predicates '*acceptable*', '*satisfies*' and '*specifies*', as they were defined in section 6.1. The sequence is running across several mutually dependent levels, where each level is assigned a number. Level '0.' denotes the most fundamental decision in the sequence – the initial specification of the problem in the explicit terms. Following this 'initialisation', the model continues with a repetitive – i.e. recursive sequence of the three predicates; with acceptability assessment serving as a stopping condition.

The model shows design as a kind of interplay of two distinct knowledge-level actions, both represented as predicates – '*specifies*' and '*satisfies*'. The former action/predicate is amending the problem specification, and the latter attempts to solve the specified problem. These two 'actions' have a potential to generate and change the explicit design knowledge. They are complemented by the third

[9] In the subsequent chapters, the model is often abbreviated to ***RFD*** (Recursive model of Framing in Design).

defined predicate ‘*acceptable*’ that is acting as a switch deciding on whether the design process continues or whether it may be successfully concluded by the articulation of a design solution.

Let us begin with the definition of an auxiliary predicate that is used in the design model in order to decide about the further steps. This auxiliary definition is useful from an interpretative point of view, so that the model can be read more intuitively as a sequence of decisions followed by actions, or alternatively, it may be seen as the results of the explicit operations assessed by inarticulate decisions.

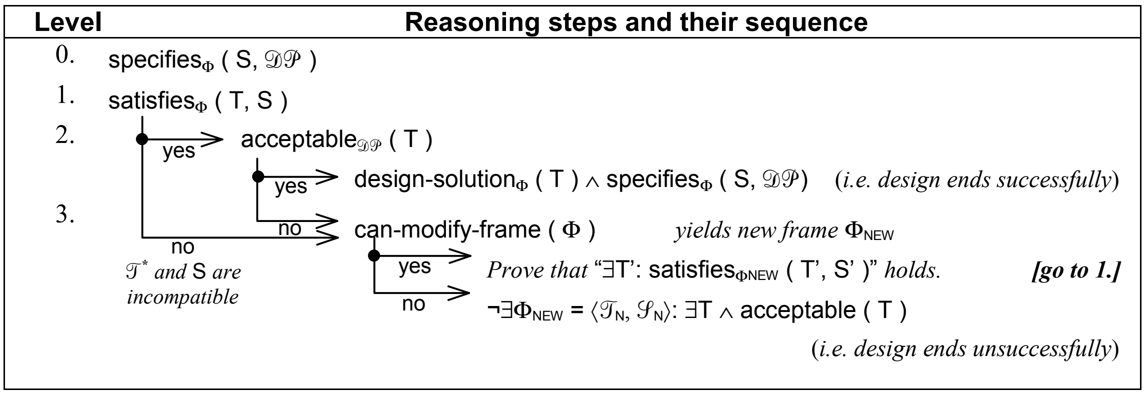
In addition to the mentioned decision on the acceptability of the design problem, there is only one other decision defined in the initial version of RFD model. This decision complies with the reviewed literature, especially Schön’s theory of reflecting in-action (see section 4.2.2). The purpose of this particular decision is to construct an exploratory probe in an attempt to modify the current conceptual frame. Thus, a frame can be modified if it is possible to explicitly articulate two ‘new’ sets $\mathcal{T}_N$ and $\mathcal{S}_N$ that provide a different perspective for attending to the design problem $\mathfrak{DP}$. The decision of the frame modification is formally presented in Eq. 6–5.

$$\textit{can-modify-frame}\ (\Phi) \Leftrightarrow \exists \Phi_{NEW} = \langle \mathcal{T}_N, \mathcal{S}_N \rangle : S' \subseteq \mathcal{S}_N^* \wedge \textit{specifies}_{\Phi NEW}\ (S', \mathcal{DP}) \qquad \textbf{Eq. 6–5}$$

The predicate ‘*can-modify-frame*’ defined above can make a decision whether to continue with the design or not, based on the existence of a new conceptual frame that can be applied to the particular, ill-structured design problem $\mathcal{DP}$. In the definition, the new frame is expressed using primes and primed conceptual sets, i.e. $\boldsymbol{\Phi}_{\mathbf{NEW}} = \langle \mathcal{T}_N, \mathcal{S}_N \rangle$. The new, modified or shifted frame $\boldsymbol{\Phi}_{\mathbf{NEW}}$ is not selected randomly, but so that the same design problem $\mathfrak{DP}$ can be meaningfully *specified* in this amended frame. Since the frame changed, its vocabularies for the specification of the problem changed as well; this corollary is expressed in the definition by using symbol $\boldsymbol{S'}$ in the predicate ‘*specifies(S’, $\mathcal{DP}$)*’.

As it is defined, this auxiliary predicate represents a decision that can be made in a certain specific phase of tackling any given design problem $\mathfrak{DP}$. However, this decision has one important and unusual feature. Namely, it is not merely a ‘yes/no’ decision but at the same time, it is a generative operation that can be used to improve the designer’s understanding of the design problem. The careful reader may have noted that the decision predicate ‘*can-modify-frame*’ deals only with the specification of an incompletely given design problem, and basically generates new knowledge about how the problem may be specified ‘differently’. This is an important side effect of the definition that actually underlines the *recursive* nature of the RFD model, and features the interaction between the two predicates as argued in the conclusion to the review (see sections 4.3 and 5.4).

It is clear from the schematic representation of RFD model in Figure 6–1 that the positive outcome of the decision contemplating the frame modification is directly followed by a potential change in the solution space. This potential change is represented by a predicate ‘*satisfies*’ that manipulates the problem solving theories and models. Thus, the mentioned interplay of (or oscillation between) the two knowledge sources is clearly observable through an exchange of information and control in the design process as modelled in Figure 6–1. Frame $\boldsymbol{\Phi}$ with a particular explicit problem specification $\boldsymbol{S}$ is verified by an attempt to find a satisfactory solution to such a specification. In the case that at least one explicitly admissible solution $\boldsymbol{T}$ is found (i.e. a problem solving model *satisfying* the explicit problem specification exists), it must be assessed for its *acceptability*.



Level
Reasoning steps and their sequence
0. specifies$_\Phi$ (S, $\mathscr{DP}$)
1. satisfies$_\Phi$ (T, S)
2.
yes
acceptable$_{\mathscr{DP}}$ (T)
yes
design-solution$_\Phi$ (T) $\wedge$ specifies$_\Phi$ (S, $\mathscr{DP}$) (i.e. design ends successfully)
3.
no
no
can-modify-frame (Φ)
yields new frame Φ_{NEW}
$\mathscr{T}^*$ and S are incompatible
yes
Prove that "$\exists$T': satisfies$_{\Phi NEW}$ (T', S')" holds.
[go to 1.]
no
$\neg\exists\Phi_{NEW} = \langle\mathscr{T}_N, \mathscr{S}_N\rangle$: $\exists$T $\wedge$ acceptable (T)
(i.e. design ends unsuccessfully)

Figure 6–1. Recursive model of framing in design as an interaction between problem specification and problem solution

Provided that the acceptability 'check' is accomplished successfully, this may be seen as a designer being satisfied with the particular solution proposal and its completeness. In other words, it is now possible to declare a *design solution* to the original, under-specified design problem $\mathfrak{DP}$. If the designer does not accept the candidate solution as addressing the original problem, frame modification is triggered (see level 3. in the figure). In this operation, the designer *reflects* on the inadequacies of the current design frame $\boldsymbol{\Phi}$, and articulates a new frame $\boldsymbol{\Phi}_{NEW}$ as a probe to rectify them.

If another frame can be found that can help the designer to update the vocabulary for the problem *specification*, and introduce appropriate new requirements or constraints, the exploratory probe is confirmed. The next step comprises the validation of the exploratory probe in terms of finding a new solution candidate ***T'*** that would address the amended explicit understanding of the problem (***S'***). In other words, this hands-on validation incorporates the *reflection-in-action*. The designer's inarticulate feelings about solution unacceptability are 'validated' through the construction of a new admissible design solution. Simultaneously, the model of the design process goes back to the level 1, and the sequence repeats in a recursive manner (note the 'control command' *[go to 1.]* in the figure).

Nonetheless, in addition to the reflective exploration, the RFD model accounts also for a specific case when it is not possible to *satisfy* a particular specification using the selected problem solving theory. This case is reminiscent of the review scenario described in section 4.1.4, where the designer discovered an explicit *contradiction* in the problem specification when he adopted a certain frame containing familiar conceptual objects for a solution construction. Using the language of RFD model, the frame consists of two mutually incompatible knowledge spaces, where the incompatibility is explicitly provable, which triggers design problem *re-framing*.

The breakdown depicted in Figure 6–1 above, is developed to address the theoretical assumptions that came out of the review of the relevant literature. The particular sequence of decisions however, obeys a few simple rules that circumscribe the category of design problems that are addressed by the proposed model. The following list contains a summary of such practical restrictions:

1) When using terms 'requirements' and 'constraints', we always mean hard, strict demands that *must not be relaxed* – see e.g. (Fox 1994) for constraint relaxation techniques.
2) We *do not consider* the design problems, where a problem specification can be simplified or otherwise relaxed; these issues are covered by other research, e.g. (Zweben and Fox 1994).

3) However, problem specification may be changed by the *monotonic extension* of an initial set; i.e. new, refining statements about a specification can be made under specific circumstances.
4) The monotonic extension of a problem specification corresponds to a designer's attempt to '*fine tune*' a problem solving model; to *narrow down* the number of derivable alternatives.
5) A sentence in the form '*Prove that (...) holds…*' represent *<u>a recursive step</u>* starting always at the level 1. of the recursive model.
6) The recursive step represents a designer's attempt to address a given design problem by some modification to the knowledge sources available for the design. It can also be understood as an order to an agent to '*evaluate*' a particular predicate '*with the new arguments provided*'.

Let us now explain the proposed recursive model by identifying formally three sub-models of reasoning that are explicable directly from the RFD model. We propose them as abstract models of different types of reasoning that can be observed in a design process. They are intended to depict the mutual dependence of different knowledge sources, as well as the interaction of explicit reasoning strategies with those using a designer's inarticulate knowledge. Particularly important for the purpose of this thesis are sub-models 2 and 3 that feature the non-monotonic introduction of new knowledge to the design process. Predicate chains in the schema definition (symbolised by arrows) refer to the paths between the particular predicates in Figure 6–1.

<u>Sub-model 1</u>

$$\begin{tikzcd}\text{satisfies}_{\Phi}(\text{T,S}) \arrow[r, "\text{yes}"'] & \text{acceptable}_{\mathscr{D}\mathscr{P}}(\text{T}) \arrow[r, "\text{yes}"'] & \textbf{design-solution}_{\boldsymbol{\Phi}}\textbf{(T)}\end{tikzcd}$$

This is perhaps the most trivial design model from the perspective of interaction between articulate and inarticulate reasoning. It represents a straightforward scenario when the selected conceptual frame $\boldsymbol{\Phi}$ contains a suitable problem solving theory $\mathscr{T}^*$ and a sound problem solving model $\boldsymbol{T}$, which can be inferred from the given theory $\mathscr{T}^*$. Furthermore, there is little need to distinguish between admissibility and acceptability of the inferred problem solving model (solution candidate). The criteria for assessing the admissibility are well defined, and a solution is acceptable whenever it is admissible and complete.

This sub-model should typically account for the well-defined problems or parts of the ill-structured problems that are well defined. Typically, this is a class of problems, where the design requirements, constraints, fixes, and elements are all well known in a particular phase/step of design. In this case, no 'tacit' knowledge or frame amendments are needed, and the problem can be solved using a generic or dedicated algorithm. In our opinion, the essence of this sub-model can be expressed as follows: "*I know how to define the problem, and I have all the necessary information available.*"

<u>Sub-model 2</u>

$$\begin{tikzcd}\text{satisfies}_{\Phi}(\text{T,S}) \arrow[r, "\text{yes}"'] & \text{acceptable}_{\mathscr{D}\mathscr{P}}(\text{T}) \arrow[r, "\text{no}"'] & \text{can-modify-frame }(\Phi) \arrow[r, "\text{yes}"'] & \text{satisfies}_{\Phi\text{NEW}}(\text{T',S'}) \\ & & \substack{\textit{yields new frame} \\ \Phi_{\text{NEW}} = \langle \mathscr{T}_{\text{N}}, \mathscr{S}_{\text{N}} \rangle} & \textit{iterate with new frame}\end{tikzcd}$$

This sub-model tackles a situation when there is at least one logically sound solution available but the designer does not accept it. We suggest that this situation has its origins in the difference between the explicit conceptual frame $\boldsymbol{\Phi} = \langle \mathscr{T}, \mathscr{S} \rangle$ that was used for the problem specification and solution construction, and the designer's expectations. The currently used conceptual sets $\mathscr{T}$ and $\mathscr{S}$ do not contain a vocabulary for expressing those expectations explicitly. Moreover, the designer became aware of these 'expectations' only when s/he proposed a particular candidate solution. In other words, the logically sound solution was subjected to a 'tacit assessment', and during this assessment a new feature or object emerged that was unaccounted for in the original frame $\boldsymbol{\Phi}$.

A kind of 'synchronisation' of the explicit, incomplete conceptual frame and the tacit expectations is performed during the process of reflection. The designer reflects on the proposed solution, the explicit problem specification and tries to pinpoint the part of the problem s/he is dissatisfied with. However, this process of uncovering the tacit expectations typically requires a construction of an exploratory probe (or more probes) that would test the different amendments to the original conceptual frame. In general, each of the probes may yield a new frame $\boldsymbol{\Phi'}$, which can be used to interpret the design problem $\mathfrak{DP}$ differently than the original frame $\boldsymbol{\Phi}$.

The scenario covered by this sub-model is clearly distinct from that described for sub-model 1 above. The main difference is in the fact that new requirements, constraints or other conceptual objects can be formulated only after the shift of the conceptual frame. Without such a shift the requirements remain 'hidden' among the designer's tacit expectations. In the original frame $\boldsymbol{\Phi}$ there are no conceptual means to express such tacit expectations explicitly. Verbal description of this class of problems may read as follows: *"I cannot solve the problem in this way, something is missing in my perspective. What if I took a different view* (…and assumed…)?" Unlike in sub-model 1 that contained sufficient conceptual apparatus, the conceptual apparatus for expressing the missing statements is not available in this scenario, but the designer knows how to construct a new apparatus (i.e. design frame).

Sub-model 3

$$\begin{tikzcd}
\text{satisfies}_{\Phi}(T,S) \arrow[r, "\text{no}"'] & \text{can-modify-frame }(\Phi) \arrow[r, "\text{yes}"'] & \text{satisfies}_{\Phi NEW}(T',S')
\end{tikzcd}$$

yields new frame $\Phi_{NEW} = \langle \mathcal{T}_N, \mathcal{S}_N \rangle$ *iterate with new frame*

This is a situation, in which the selected conceptual frame $\boldsymbol{\Phi}$ contains a particular problem solving theory $\mathcal{T}^*$, and a particular explicit specification $\boldsymbol{S}\subseteq\mathcal{S}^*$ that are mutually inconsistent. In other words, there is a logical contradiction pointing to the axioms of deployed domain theory $\boldsymbol{DT}$ conceptualised by $\mathcal{T}$. Consequently, no sound problem solving model (a design solution) can be found in an unsound theory. After changing a conceptual frame from the original $\boldsymbol{\Phi}$ to a new $\boldsymbol{\Phi}_{\mathbf{NEW}}$, new conceptual objects may be identified (i.e. $\mathcal{T}_N$) or problem specification may be re-visited (i.e. $\mathcal{S}_N$). Within a new conceptual frame $\boldsymbol{\Phi}_{\mathbf{NEW}}$ the conflicting axioms (usually constraints) of the originally used domain theory $\boldsymbol{DT}$ are interpreted differently. The new interpretation of an axiom is clearly a non-monotonic operation influencing the selection of an appropriate domain theory – a new domain theory $\boldsymbol{DT}_{\mathbf{NEW}}$ may suit the amended conceptual frame better than the original one that was responsible for the contradiction.

Nevertheless, the class of problems addressed by this sub-model is rather well structured. There is available explicit knowledge of a logical contradiction in the problem solving theory. There is an explicit (typically constraining) condition that 'describes' the contradiction in the vocabulary of the original frame $\boldsymbol{\Phi}$. It may be used in the search for a new frame that can be assessed for compliance with this particular condition. We believe the following statement can describe the class of problems tackled by this sub-model: "*I know exactly what is wrong, and I have certain options available that may fix it.*"

Like in schema 2, the designer does not have the expressive means in the current frame to remove the contradiction; he needs a new conceptual frame. As mentioned earlier, this sub-model is close to the research on the resolution of physical contradictions in the inventive problems review in sections 4.1.4 and 5.1.4.

The sub-models following directly from our RFD model that were described above are defined on an abstract level of concepts and conceptual design frames. They are not computational or operational models. Their main purpose is to model certain reasoning strategies, which conceptually underpin a designer's decisions on the level of his or her knowledge. Any detailed discussion on the operational models of the individual predicates is not within the scope of this section. Nevertheless, it is interesting to investigate how problem specification may be satisfied, how different solutions can be generated from particular problem solving theories, or how to discover a suitable familiar representation for framing the design problem.

Some of these topics are discussed later in the thesis, for instance, one of the operational models for the predicate '*satisfies*' is briefly discussed in section 6.4. The main focus of this thesis, however, is to develop a conceptual model and validate it empirically. Therefore, most of the discussion on the implementation is left for section 12.4 mentioning further work. However, before presenting the details of the empirical study, let us exemplify the proposed RFD model (and its three sub-models) on the practical scenarios that are reported in the reviewed literature.

6.3 RFD model in examples

Having proposed a formal definition of the concept of design frames thus forming the foundations of the recursive model of framing in design (RFD), we believe that the best way to justify and explain the claims is by examples. The identified 'classes' of design problems from the previous section that are covered by the RFD model are expanded in the subsequent paragraphs, and 'instantiated' using the empirical findings found in the reviewed literature. The accounts given below are abbreviated and focus on the relevant phase that best illustrates a particular sub-model. Additional details and the step-by-step transcripts of our experimental study can be found elsewhere in this thesis; see chapter 8.

Before discussing the details of the individual 'instances' of the RFD model, we would like to note that any of the proposed sub-models might characterise an entire design task. However, in most cases, several of the sub-models may appear during the process of designing. This must not be viewed as a shortcoming of the theory of conceptual frames in the design! In our opinion, the situation when the multiple portions of a single design case correspond to different 'branches' of the RFD model only underlines the ill-structured nature of design problems. Simultaneously, the observation of the different branches of the RFD model supports one of our initial assumptions that designers apply a variety of reasoning techniques when they are tackling ill-structured design problems. The point of 'multiple paths to a design solution' emerged very early in this research, and directly relates to the 'one-case-multiple-models' scenario (Dzbor 1999; Dzbor and Zdrahal 2001b); see also chapters 4 and 5.

Rather than labelling a design task with a single branch of the RFD and following a procedure that may be typically associated with such a schema, design is by nature exploratory. Several paths may lead to an acceptable solution – some of them more effective than others. The acceptable solutions may be innovative, other time more prototypic, but it cannot be said in advance, which path guarantees the result. We argue that this interplay of different reasoning strategies within a single design case is the fundamental consequence associated with Herbert Simon's (1973) 'poor structure'. Furthermore, we argue that the application of different sub-models of our RFD theory is also in accordance with the theory of reflection-in-action authored by Donald Schön (1983), and reviewed in chapters 3 to 5. Reflection is an opportunistic technique for resolving the gaps between the tacit expectations and explicit commitments, and the designer may try different approaches to overcome this uncertainty.

From a theoretical point of view, we may expect that certain schemas are more likely to coincide with a particular 'class' of design problems. For instance, the less structured the task is, the higher shall be the frequency of observing sub-models 2 or 3, rather than number 1. Inventive and highly creative problems may comply with sub-model 3 more frequently than with the other ones. On the other hand, more routine problems re-using the prototypes may comply with sub-model 1. A short correlation analysis that precedes the RFD analysis and investigates these and similar questions is presented to conclude chapter 7. We may say that the analysis rigorously confirms these intuitive expectations.

Nevertheless, in this section, we present the exemplar design scenarios in terms introduced earlier in sections 6.1 and 6.2. We show what is a conceptualisation, domain theory, problem solving theory, problem solving model, problem specification, constraining condition, etc. for each of the examples. First, the class of well-defined problems is represented by Sisyphus-2 elevator configuration in section 6.3.1. The second class featuring reflection refers to Schön's experiment with architects, and appears in section 6.3.2. Finally, the class of contradictory problems exemplified earlier in sections 4.1.4 and 5.1.4, will be translated into the terminology of the RFD model in section 6.3.3.

6.3.1 Sub-model 1: well-defined design problems

As already mentioned, this class represents well defined and straightforward design situations[10]. From the review in chapters 4 and 5, we know that these are often 'routine problems' (Gero 1990; Gero 1996) or 'algorithmic problems' (Chandrasekaran 1990). Problem solving theories available for this category of problems tend to be very thoroughly defined, often contain details of design elements, relations, functions and properties. From a semantic point of view, various requirements, constraints, preferences, and assumptions are formulated with clearly defined contexts and scopes of validity. In addition to a thorough knowledge of problem specification, the applicable problem solving methods (PSM) are available. Typical PSM include constraint satisfaction techniques (Kumar 1992) or propose-revise fixes (Yost 1992). Well-defined problems usually have precise criteria for solution admissibility and optimality, which are used instead of tacit checks on subjective acceptability of the solutions.

This class of problems tends to be strictly circumscribed in advance, and this circumscription does not change during the design process. This certainty implies that all the conceptual objects that are needed for a problem specification or solution construction are known before the process starts. The task is usually to select those objects, components and relations that perform optimally according to a

[10] Note that the adjective 'straightforward' refers to the complexity of conceptual design frames and their articulation, rather than the complexity or effectiveness of the problem solving methods.

certain criterion (e.g. price). The circumscription has no negative impact on creativity, mainly because there is little potential for innovation in so well defined design spaces.

A typical representative of this category is an elevator configuration design according to the Sisyphus-2 specification (Yost 1992). Further particulars for this illustrative example as well as the essential details of the domain are provided in Appendix B. On the level of conceptual frames, this problem is very simple; it is typically tackled in a single design frame that is thoroughly described in Yost's report. There are numerous implementations of our predicate '*satisfies*' in the form of various PSM and/or knowledge representations; nonetheless, they all share the same conceptual frame (Yost 1992; Brazier, van Langen *et al.* 1996b; Motta, O'Hara *et al.* 1996; Wielinga and Schreiber 1997). No modifications of the design frame are observable throughout the task; the initial conceptual frame is sufficient to generate a solution. In other words, the design of an elevator in Sisyphus-2 is conceptually a routine task.

Let us list a few statements specifying the Sisyphus-2 task. In line with the RFD theory, we list also the conceptual primitives for the construction of solutions, and extracts from the domain theory. Since Sisyphus-2 is very thoroughly defined, these are only illustrative statements. More details are presented in appendix B, or a complete definition can be found in (Yost 1992). For the sake of clarity, the conceptual primitives of the frame are printed in italics in the following extracts:

a) *Circumscribing assumptions*:

- '(This document) applies to *cable-operated elevators* driven by an *overhead motor* assembly';
- 'The *car* speed must be between 200 and 400 feet per minute';
- 'The *car* may have *doors* only in the front of a *cabin'*;
- 'Dimensions of the *floor openings* are the same on all floors'

b) *Problem specification (typically required features)*:

- 'My *elevator* should have the *car* capacity of 3000 pounds';
- 'I want the *platform* width and depth to be 70 and 84 inches, respectively';
- 'My *elevator* will serve and *stop* on exactly 6 floors';
- 'Width and height of my *building's wall openings* are 42 and 84 inches, respectively.'

c) *Problem specification (typical constructional constraints)*:

- '*Car buffer blocking* height must be at least zero inches, and at most 120 inches';
- 'There must always be at least one *car buffer*';
- 'If the *safety beam model* is *B1* or *B4*, there can be at most one *buffer*. If the *safety beam model* is *B6*, there can be at most two *buffers*'

d) *Problem solving methods (fixes to violated constraints)*:

- 'If *car buffer* height constraint is violated, try increasing/decreasing the *hoistway pit* depth by one-inch steps' (has-severity D9);
- 'There is no fix if minimal number of *car buffers* is violated';
- 'If the maximum number of *car buffers* is violated, try upgrading the *safety beam model*' (has-severity D8)

e) Preferences (i.e. classes of fixes by their severity, impact on the partial design solution):
   - 'Severity D4 = changes *minor equipment* selection or sizing';
   - 'Severity D8 = changes *major equipment* selection or sizing';
   - 'Severity D9 = changes the *building* dimensions'

f) *Conceptual object for solution construction (design structures)*:
   - 'Elevator *doors* are identified by a *door model* code ∈ {SSSO, 2SSO, SSCO, 2SCO}';
   - 'We have three *platform models*, 2.5B, 4B, 6B. (listed from the smallest to the largest)';
   - 'There are five *safety beam models* – B1, B4, and B6 (from smallest to largest)';
   - 'There are five *sling models*. In order from smallest to largest, they are 2.5B-18, 2.5B-21, 4B-HOSP, 4B-GP, and 6C'

g) *Domain theory (relations among conceptual objects, properties of the concepts, ...)*:
   - 'The major components of an elevator system are: *hoistway*, *car*, *counterweight*, etc.';
   - '*Car assembly* contains a *passenger cab*, its *supporting structure*, and *safety mechanisms*';
   - 'The *supporting structure* comprises a *platform* and a *sling*';
   - 'The *counterweight* consists of a stack of iron *plates*';
   - 'Parameter weight of *platform* is defined as follows (in pounds):
     – *Platform model* 2.5B: S+0.318*(5.06W+0.11WD+(D–7.6)*(3.14+0.8AP)'

What makes Sisyphus-2 a well-defined problem is the fact that all the statements similar to those listed above are available before the designer starts his or her work. In addition, the statements defining the Sisyphus-2 problem are sufficient to design a Sisyphus-2 elevator; no additional knowledge is required. There are even various value functions available that help the problem solving methods choose the most suitable components and the optimal design fixes. Therefore, we refer to this class of problem (or sub-problem) that exhibits the behaviour covered by sub-model 1, as *well-defined*.

6.3.2 Sub-model 2: frame modification with reflection

As we already mentioned, this part of the RFD model accounts for ill-structured problems that exhibit one interesting feature. Namely, the designer is able to construct a sound and admissible solution, but this is not accepted. Consequently, the designer has to articulate the tacit feeling of the unacceptability and justify the decision not to accept the current solution. This articulation and justification, however, happens in a new conceptual frame. A designer amends the original frame so that it shows the features that may have been responsible for its unacceptability and rejection in explicit terms.

Typically, this class of problem commands sound problem solving theories, which are consistent with the explicit problem specification. Therefore, one purpose of the articulation of a new frame is to introduce new knowledge (requirements, constraints or structural primitives) that would reflect a contradiction between the current solutions and the tacit expectations on the explicit level. However, an important characteristic of these problems is that the new frame is articulated somewhat tentatively, to explore if this amendment is addressing the tacit feeling or not. The purpose of this exploratory probing is to look at the problem from a different perspective and *specify* it differently – 'acceptably'.

As we mentioned in the review, these perspective shifts are usually reflective (section 4.2.2). Designers frame the problem in a different conceptual vocabulary to reflect, uncover and criticise their tacit understanding of the design situation. The whole process of framing is driven by the goal to produce a more thorough understanding of the design problem together with a more complete explicit problem specification and problem solving theory. Consequently, the belief justifying the re-framing is to construct an acceptable design solution. If this is achieved and a solution in a new frame is (more) acceptable, the exploratory probe reflecting on the tacit expectations has been 'successful'.

Consider a typical scenario illustrating the above-mentioned reflective probing. This extract is taken from Schön's (1983) case study of an architecture student whose task was to design a site for a future school (pages 79-104). The student framed a very vaguely specified problem using a few explicit commitments and requirements she wanted to impose on the design situation. For instance, she decided on the conceptual primitives such as 'classroom', 'gym', 'playground', and 'slope'. From the theory of architecture, she opted for a diagrammatic plan of the site, in which all the buildings were in an easy-to-reach distance from each other. Thus, an extract from the first iteration may look as follows:

a) Circumscribing conceptual frame:

- '*School site* comprises *classroom*, *gym*, *playground*, *…*';
- 'The *school site* is located on a *sloping terrain*, the elevation of *slope* is 15 feet'

b) Problem specification (required features):

- 'Individual *buildings* shall be in a close walking distance from each other';
- 'The *buildings* on the *site* need to fit the shape of the *slope*';
- 'The *school* shall have six *classrooms*'

c) Problem specification (constraining conditions):

- 'There should be at least one *playground* on site';
- 'The minimum height of a *classroom* is 5 feet (from safety regulations)

d) Domain theory (based on the theory of architecture, norms, etc.):

- 'A diagrammatic layout features *geometrical objects*, mainly *rectangles*';
- '*Open space* and *playground* are typically inside the 'circle' of other *buildings*'

e) Problem solving theory and one solution candidate:

- 'My *school site* complying with a simple diagrammatic plan is shown in Figure 6–2a';
- 'A more significant diagrammatic plan is achieved by grouping the geometrical shapes into larger units (see L-shaped classrooms in Figure 6–2b)

Choosing a prototypic approach enabled the student to draw an early sketch of her design. This sketch (shown in Figure 6–2) incorporated the 'primitive' blocks of her school site into the slope. However, the early commitment was not successful. The student reflected on the proposal with her tutor, and they agreed that the site 'was ugly' in terms of relating the buildings and the slope. Thus, the tutor and the student had certain expectations from the site and their respective designs, which were not met by the early solution. As the tutor put it, one thing he did not like was that the approach lacked 'discipline', which he considered even more important for such an atypical site.

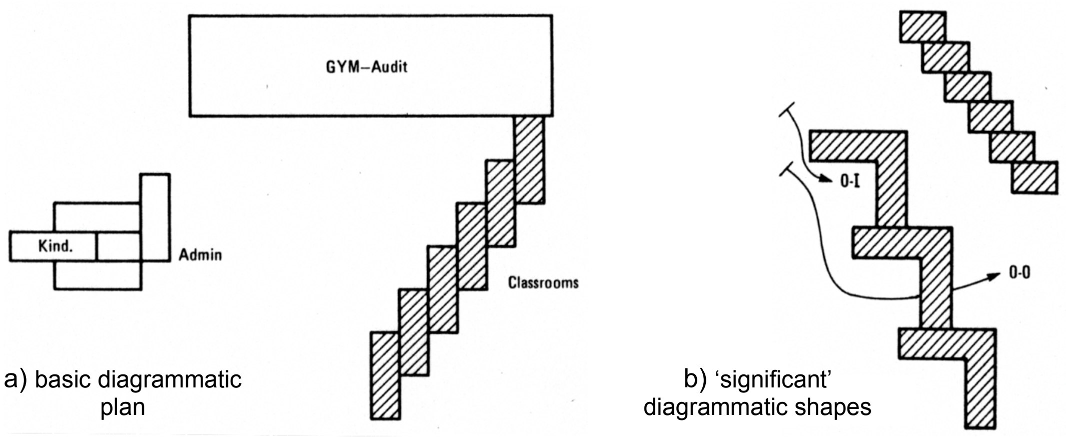

GYM–Audit
Kind.
Admin
Classrooms
a) basic diagrammatic plan
0-I
0-0
b) 'significant' diagrammatic shapes

Figure 6–2. School site plans in the basic 'diagrammatic' frame

He suggested that the incorporation of the site uniqueness (the site was on a gently inclined slope) into the layout may restore the 'beauty'. This restoration could include 'carving' the classrooms into the slope, thus achieving a unique geometrical discipline both horizontally and vertically. The slope was no longer attended to as a monolithic unit of the site; it was divided into three levels. Thus, each significant L-shape from the original student's drawing acquired a counter-measure in the slope, which improved the discipline of the site. Thus, a re-framed design situation can be expressed in the following conceptual terms:

f) *New circumscribing conceptual frame*:

- 'A ***coherence*** between *school buildings* and *site geography* is emphasised';
- 'The ***discipline*** can be achieved by incorporating *significant shapes* of the *school site* into the *landscape* of the *slope*'

g) *Problem specification (required features) re-visited*:

- 'The *site* should fit into the *geometry* of the *slope*';
- 'The *buildings* should be connected by the *sloping landscape* of the *site*';
- 'The *fit* shall be observable in both the ***horizontal*** and ***vertical*** plans of the *site*'

h) *Domain theory (based on the theory of architecture, norms, etc.) re-visited*:

- 'A *sloping landscape* requires *splitting* the *site* into smaller coherent *units*'

i) *Problem solving theory and acceptable solution candidate*:

- 'My *school site* complying with the new perspective is shown in Figure 6–3';
- 'It also implements interesting *extensions*, such as *precincts* and *galleries* (see Figure 6–4)'

Thus, a modified conceptual frame for attending to the problem contains new concepts of 'site geography' and 'discipline'. Furthermore, new requirements emerged for the adaptation of a prototypic diagrammatic school plan to this unique landscape. The new frame helped to impose the diagrammatic plan onto the slope rather seamlessly (see plan A in Figure 6–3). From an initial tacit need of 'discipline', a new concept 'nooks' emerged that made the site more compact (see plans B and C in Figure 6–3). In addition, entirely new opportunities opened in the re-framed situation, e.g. concepts 'precincts' and 'galleries' suddenly arose from a disciplined landscape (see Figure 6–4). A new perspective was based on the original one; nonetheless, it was much more coherent and it naturally fit, complemented, and used the specifics of the 'screwy shape' of the slope.

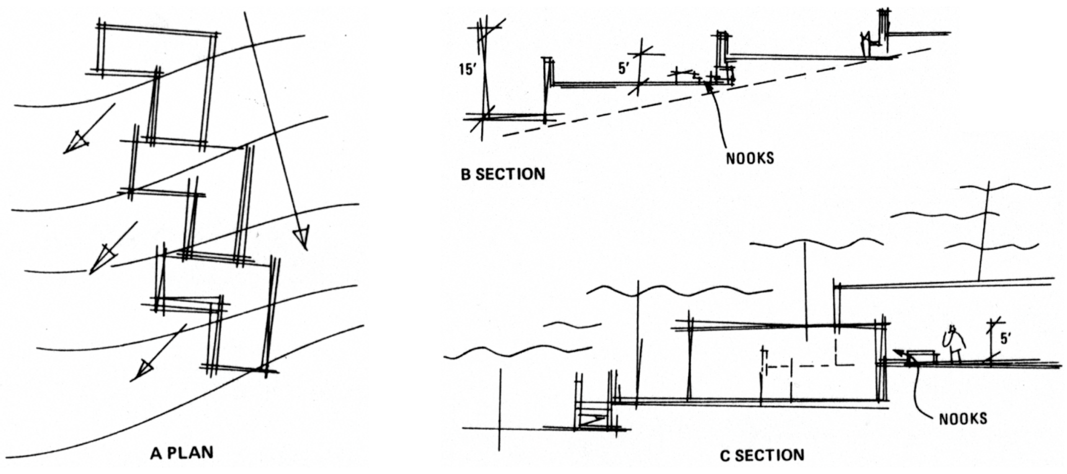


Figure 6–3. School site with the imposed horizontal and vertical discipline

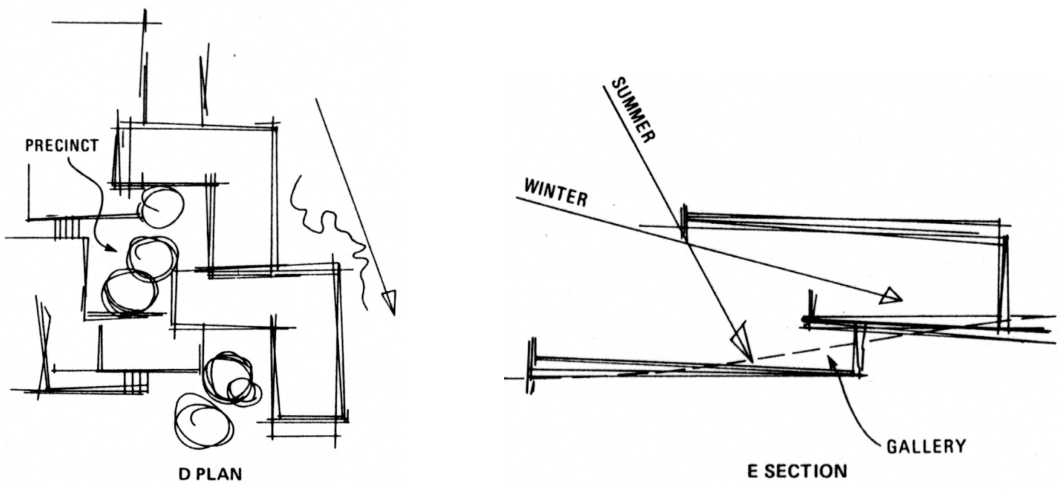


Figure 6–4. New concepts unexpectedly emerging in a re-framed situation

Note that this is only a short episode from Schön's case study, which continued with more re-framing. New features emerged from an attempt to satisfy the specification in the amended frame, which led to the repetition of the reflective criticism and further frame development. Nevertheless, the above extract shows how a tacit perception of 'site ugliness' led to an incorporation of new knowledge thanks to the shift in the design student's perspective. All the figures presented above are borrowed from (Schön 1983).

6.3.3 Sub-model 3: frame modification for contradictory problems

This part of the RFD model is typically triggered when it is not possible to generate any sound solution in the current conceptual frame. The reason for this is the contradiction between two or more explicitly required features. Such a contradiction typically appears when the selected conceptualisation of the problem does not produce a sound theory in respect to the explicit problem specification. How is this possible? The reader may remember that we defined the problem solving theory $\mathcal{T}^*$ as an instantiation of a particular, generic domain theory $\boldsymbol{DT}$ using the selection of conceptual primitives $\mathfrak{T}$. The conflict addressed by this sub-model can be typically traced back to the domain theory.

However, this does not mean the domain theory itself must be flawed or unsound. On the abstract level of a generic domain theory, everything may be consistent. It is the interpretation of the generic theory in terms of a particular chosen conceptualisation and the commitment to particular explicit problem specification that clash. The clash eventually produces an unsound conceptual frame for the particular problem, and any problem solving theory explicitly articulated in the unsound frame, will be unsound and inconsistent, too.

This perspective corresponds to the argument of Altshuller (1984). He argues that most of the inventive designs stem from deep contradictions. Such 'deep contradictions' are more than just a simple absence of a desired feature in the current state (Coyne, Rosenman *et al.* 1990). This type of conflict addresses more fundamental issues that are very deeply immersed in certain, usually familiar conceptual frames. The invention or innovative aspect of a solution thus appears as a response to challenging these deeply rooted familiar interpretations rather than tackling the superficial absence of a desired feature. Thus, Altshuller's argument of seeing the inventive problems differently before attempting to construct any innovative solutions is emphasised in the RFD model.

A design problem must be given a new conceptual frame in order to continue with the design. The articulation of a new frame is essential for moving the design from the 'deadlock' of a contradictory theory. Note that we do not claim that any frame amendment would eventually lead to an innovative solution. Our RFD model simply claims that a frame amendment in the case as described above, is necessary to construct *any* solution at all.

Consider the scenario described earlier in section 4.1.4 designing a cooling system for the lens polisher. The familiar frame and subsequently, conceptualisation and domain theory includes the following statements:

a) Circumscribing conceptual frame:
- '*Polisher* is made of *resin*';
- '*Resin* contains tiny *abrasive* particles for machining the *glass*'

b) Problem specification (required features):
- 'The *glass lens* need to be polished to a high precision';
- '*Coolant* needs to be poured on the lens *surface* to prevent its *damage*'

c) Problem specification (constraints):
- 'The *solid* polisher cannot be replaced by several smaller ones because this would compromise the quality of polishing'

d) Selected conceptual primitives (structures):
- 'My *polisher* is a *solid* block of *resin*';
- 'My *coolant* consists of cold *water*'

e) Domain theory (based on physics of solid bodies):
- '*Liquids* cannot pass through *solid* bodies';

- ‘*Solid* bodies cannot be *liquid* or *gaseous* at the same time’

⇒ <u>*Contradiction in the problem solving theory*</u>:

- ‘I need to pour water (*liquid*) through a *solid* polisher, and this is not allowed by the domain!’

It is the selected part of the domain theory, as following from the circumscribing conceptual frame, that prevents an object from being monolithic and at the same time broken into pieces. Nothing can be done with this constraint as long as we are in the familiar conceptual frame. Nonetheless, the introduction of ‘transitional’ vocabulary below shifts the frame, and uses a different domain theory. The concept of ‘phase transition’ removes the conflict of a polisher being a decomposable monolith. Simply, if frozen, it is monolithic and solid, and it can polish the lens. The re-visited problem solving theory draws on the (re-visited) domain knowledge that ice melts and changes into water when warmed up. Thus, water is delivered onto the lens surface without breaking the polisher into pieces:

f) <u>*New circumscribing conceptual frame*</u>:

- ‘*Polisher* may not be made of *resin* but some ***transitional substance*** (e.g. *ice*)’;
- ‘*Ice* may contain tiny *abrasive* particles for machining the *glass*’;
- ‘*Ice* as a transitional substance may ‘turn into’ *water*, i.e. we may achieve its ***melting***’

g) <u>*Problem specification (requirements) re-visited*</u>:

- ‘*Ice* must be ‘turned into’ *water* during polishing – melted’

h) <u>*Domain theory re-visited*</u>:

- ‘*Ice* melts into *water* when exposed to a *temperature* higher than 0°C’;
- ‘*Temperature* on a *surface* increases in effect of a *friction* between two bodies’

i) <u>*Problem solving theory re-visited and **sound***</u>:

- ‘*Ice* of the polisher generates *friction* on the *surface* of the *lens*’;
- ‘*Ice of the* polisher melts because of increased *friction* – surface temperature’;
- ‘*Melting ice* releases *water* onto the surface of polished *lens*’

6.4 Applicable operational models for ‘satisfies(T, S)’

Let us propose one possible operational model of how the relation ‘*satisfies*’ can be calculated and evaluated. This is only a brief overview of a few techniques refining the conceptual level of the RFD model towards a lower, computational level. This section illustrates what was meant in sections 6.1 and 6.2, when we said that a designer searches for a sub-set of the problem solving theory ($\boldsymbol{T} \subseteq \mathfrak{T}^*$) that would satisfy the explicit problem specification (***S***). Note that the operations mentioned below do not form a part of RFD analysis, and they were not included in the experimental study.

We formalise the reasoning during the satisfaction phase using the language of logic. *Predicate calculus* or first-order logic is an explicit representational framework that may be used for the purposes of knowledge representation. Logic is considered a base to most other studies, and is defined as a study of methods of reasoning (Mendelson 1979). Since logic is concerned with the form of reasoning rather than content, many people consider its claims self-evident and hardly usable in an everyday, ill-structured life. We admit that logic may not be a suitable means to explain *all* peculiarities of human reasoning in general and about design in particular. Nevertheless, thanks to its great expressive power, logic is a useful and sufficiently generic 'language' for the explicit representation of the design phase, in which explicitly specified design problems can be solved.

In accordance with Newell's (1982) definition of 'knowledge level', the medium of knowledge may be implemented in various symbol-level representations. Since we are now concerned with the explicit reasoning, we do not see any reason why we could not use the predicate calculus (first order logic) as a particular symbol-level representation to express the elementary reasoning steps for modelling the 'problem satisfaction' phase. Below, we show that such a replacement of other languages with the predicate calculus is only an isomorphic projection. In other words, the form changes, but the information content is preserved. Anyway, it should be emphasised again that what is covered by the logical theory is the reasoning on the explicit part of design knowledge as it may appear in real life.

In the following sub-sections, we selected three basic operations, a designer may perform from a logical point of view. We begin with the logical abduction, and relate it to the synthesis and divergence, as mentioned in the earlier research studies (see section 6.4.1). Then we attend to the deductive reasoning step, and focus on the analysis and the explication of the consequences of the given set of premises (in section 6.4.2). Finally, we address the evaluation of solution soundness and admissibility. In sections 6.4.3 and 6.4.4, we discuss the explicit evaluation of the deduced theorems in terms of truth maintenance and consistency of the problem solving theory.

6.4.1 Abduction from a logical theory

Abduction is one way of connecting a triplet of logical formulae that have this particular (abstracted) form: ϕ, γ, and '$\phi \Rightarrow \gamma$'. Simply said, knowing that γ and '$\phi \Rightarrow \gamma$' are valid formulae, we are allowed to conclude that ϕ may be true as well, because in our theory, it accounts for γ. As Levesque (1989) points out, abduction is a form of a *hypothetical* reasoning. He argues that abduction is a kind of quest for a formula ϕ that together with our underlying background knowledge[11] would *sufficiently* account for a formula γ that is known to be valid. Therefore, to re-phrase the definition from the beginning of this paragraph, we say that knowing γ and '$\phi \Rightarrow \gamma$' we 'hypothesise' the validity of ϕ. In other words, abduction performs the reasoning process 'backwards'; i.e. from the known consequences it hypothesises the possible reasons/premises/explanations.

The fundamental term in the definition above is the word '*sufficient*'. A typical illustration of abductive reasoning is diagnostics, when one tries to explain a certain observation γ, and at the same time, this observation is known to be a consequence of a symptom ϕ. We may not know for sure the exact reason why the effect γ has occurred. However, if we hypothesise symptom ϕ as its (sufficient) cause, then we can explain the effect γ as a consequence of premise ϕ. Next, we may actually look for a proof of the conjectured and hypothesised symptom ϕ in the investigated system. If we later find a proof of the symptom ϕ (e.g. by measuring it in a system), we also confirm the conjectured hypothesis suspecting this symptom being responsible for the effect γ. It is indeed, sufficient to account for the observation made but it is not inevitable. More on diagnostics and the application of abduction can be found in (Reiter 1987; Poole 1989b; de Kleer, Mackworth *et al.* 1990; Poole 1990).

There are different strategies for abduction (Poole 1989a). However, reasoning by abduction is (and always will be) a hypothetical form of reasoning. It is based on the existing logical theory but unlike deduction, it does not conclude the *inevitable* consequences, only the *sufficient* reasons! There is no guarantee that a formula that has been abduced, is the real cause of a particular effect. In other words, abduced formulae are not theorems of the logical theory and abduction is not a sound step in any logical proof.

Basic principles of abduction in design can be illustrated as follows. Suppose $\boldsymbol{R}$ is some desired functionality of a designed artefact; i.e. $\boldsymbol{R} \subseteq \boldsymbol{S}$, where $\boldsymbol{S}$ is a set of explicitly specified functions or properties (see definitions in 6.1). At the same time, statement $\boldsymbol{R}$ appears in the logical problem solving theory $\mathfrak{T}^*$. Next, assume that an artefact exists that is represented by a set of formulae $\boldsymbol{T} \subseteq \mathfrak{T}^*$. If it is known in the problem solving theory that $\boldsymbol{T}$ delivers functionality $\boldsymbol{R}$ (e.g. $\boldsymbol{T} \vdash \boldsymbol{R}$) and $\boldsymbol{T}$ is consistent with the explicit problem specification (i.e. $\boldsymbol{S} \models \boldsymbol{T}$), then we say that $\boldsymbol{T}$ is abducible from the theory $\mathfrak{T}^*$. Artefact $\boldsymbol{T}$ has a sufficient means to ensure that the requested feature(s) $\boldsymbol{R}$ will eventually occur, as desired. Of course, we have to add with the same breath – as long as nothing else invalidates this hypothetical choice. In this sense, we discovered potential solution $\boldsymbol{T}$ to a given partial specification $\boldsymbol{R} \subseteq \boldsymbol{S}$, albeit by a speculative rather than sound logical reasoning.

Nevertheless, from the perspective of speculations and hypotheses, abduction may be a suitable strategy to implement the exploratory paradigm in design (as reviewed in section **4.1.1**). This is mainly a corollary of the fact that the abduction makes a tentative choice of further directions based on the current position in the design space. Tentatively abduced formulae may be re-considered later, when new design constraints and/or requirements emerge (e.g. by a frame shift).

6.4.2 Deduction from a logical theory

Unlike abduction that constitutes a hypothetical reasoning and looks for sufficient means, deduction from a logical theory is the only sound operation of the predicate calculus. To reason by deduction means to derive the *inevitable* and *necessary* conclusions from an arbitrary given set of logical theorems and axioms in a given logical theory. Provided that certain premises are believed to be valid, deduction only 'propagates' this belief throughout the logical theory, and explicitly formulates the theorems that consequently must be also believed. The 'necessity' is based on the assumption of rationality (Newell 1982). It means that if an agent believes in a particular axiom, it must 'necessarily' believe in all consequences of that axiom.

Logical deduction starts with formulae α and '$\alpha \Rightarrow \gamma$', and concludes validity of formula γ. Logical deduction is defined as a sequence of well-formulated formulae $\alpha_1, \ldots, \alpha_n$ such that $\alpha = \alpha_1$, $\gamma = \alpha_n$. In addition, each α_i is either an axiom in the given logical theory, a logical tautology or a direct consequence of the existing theorems by an allowed rule of inference (Mendelson 1979). If such a

[11] Here most sources assume that the background knowledge contains among other sentences, also the logical implication relating formula ϕ as a premise and γ as a conclusion.

sequence is found, we may alternatively say that formula γ can be *logically proved* from the premise α, or it can be *deduced* from α. In both cases, the formal notation is the same; i.e. $\alpha \vdash \gamma$.

Let us go briefly through the conditions of proof-logical deduction. An axiom is such a theorem that is always believed to be valid in a particular model of the world, and this belief does not require a logical proof. A tautology is a formula that is valid in any conceptual model of the world, regardless of its semantic meaning. For instance, schemas $\omega \vee \neg\omega$ or $\omega \Rightarrow (\omega \vee \beta)$ are typical tautologies; they are always 'true'. In addition to the 'conceptual' or 'context-dependent' axioms, the tautologies are also a form of axioms – only they are 'context-independent' (see also classification in section 3.2.1).

The only allowed rule for the inference in the conservative predicate calculus is *modus ponens* that acknowledges implication ('$\Rightarrow$') as the only logical operation denoting necessity (Mendelson 1979). The application of an inference rule for a justification of a particular reasoning step in a deductive proof, is verbally translated to the sentence: '*Formula δ follows from the previous step ω by virtue of modus ponens as an inference rule.*' A usual definition of modus ponens looks as follows:

ω
$\underline{\omega \Rightarrow \delta}$ ... *premises*
δ ... *consequence*

Thus, it is possible to see a logical proof of one formula using the other ones as a formal representation of the dependency between them. Such a dependency is typically expressed using the operator '$\vdash$' to emphasise that in general, a long chain of direct implications ('$\Rightarrow$') may lie between the two formulae. However, according to the basic deduction theorem (Levin 1974; Mendelson 1979), the 'proof' operator can be replaced by the direct implication to denote the dependency. This is a powerful theorem because it enables skipping the lengthy deductive chains, and expressing the deduced consequence using an operator of direct logical implication.

Such an abbreviation can significantly reduce the amount of effort in the future, when we may want to deduce additional consequences. The newly formulated implication can be used in a way that is similar to that of the other theorems of the logical theory. Thus, based on deduction it is possible to explicitly formulate new dependencies, i.e. '*represent new knowledge*'. However, the reverse side is that if such abbreviations are used as a justification of a reasoning step, it may confuse an external observer. We suspect that this 'abbreviation' may also be one of the reasons for labelling an inference as 'spontaneous' (Dominowski and Dallob 1995).

Logical deduction does not create new conceptual structures or generate new knowledge. However, it explicitly articulates consequential knowledge about the known conceptual objects. This analytical character of deduction was used e.g. in a qualitative modelling of devices (Gruber and Iwasaki 1991). However, some authors prefer to talk about causal links/chains instead of deduction for various reasons. The 'causal' terminology only emphasises the relations that are represented by logical theory.

Let us return to deduction in design. As we already mentioned, designers start with an initial explicit specification ($\boldsymbol{R} \subseteq \boldsymbol{S}$) of a tackled design problem $\mathfrak{DP}$. Drawing on their domain knowledge, they believe that their design must contain a component represented as $\boldsymbol{T}$ – in other words they proclaim $\boldsymbol{T}$ being a hypothetical axiom of the problem solving theory $\mathfrak{T}^*$ (in a given conceptual frame).

However, they must prove the logical dependency between the axiom and the explicitly required features ***R***; in other words, ***R*** must be deducible from the believed axioms and theorems of theory $\mathcal{T}^*$.

If a logical proof $\boldsymbol{T} \vdash \boldsymbol{R}$ is found, then ***R*** is indeed a proof-logical consequence of using the component ***T*** as a (partial) design solution. By virtue of the completeness theorem (Levin 1974; Mendelson 1979), the component ***T*** is also consistent with deducible consequences (i.e. $\boldsymbol{T} \models \boldsymbol{R}$). However, in addition to this deduction, component ***T*** may also imply other consequences – let us mark them as ***D***, that were not among the initially explicated ones (i.e. $\boldsymbol{D} \not\subset \boldsymbol{S}$). Since ***D*** is a logical deduction from the axiom ***T***, it is also consistent with the underlying axiom. Nevertheless, the logical deduction cannot give any more information about the consistency between the explicitly desired specification ***S*** and ***D*** – the additional deduced consequence. This consistency, between the two objects, is crucial for any further development of design, and must be evaluated accordingly. However, this is already a task we leave for section 6.4.3.

One more remark however, this scenario almost mirrors the case from Donald Schön's (1983) book (see sections 4.2.2 and 4.3). With the desired goals, a solution may also deliver additional, unexpected, and potentially undesirable consequences. Thus, all deduced and implied consequences must therefore be explicitly evaluated, in order to ensure that the compatibility and consistency is maintained during the design.

6.4.3 Admissibility – evaluation of logical consistency

In the two preceding sections, we first learned how the designers may tentatively articulate candidate solutions addressing a particular interpretation of the design problem and its explicit specification using abduction. We next learned how they may explore the consequences and analyse the hypothesised solution by deductive reasoning. Section 6.4.2 concluded with a suggestion that when certain consequences were deduced from the believed axioms, these needed to be assessed for their admissibility[12] with respect to the explicit specification ***S***.

The purpose of a logical evaluation of the current solution and its consequences against the explicit requirements and constraints may seem to be redundant. The usual justification of the term 'redundant' is that if some consequences were logically unsound then there would be something wrong with the underlying logical theory that allowed such a contradiction. Nevertheless, this argument is not applicable to the reasoning in design. First, the abduction is a speculative – not a sound form of reasoning; therefore the abduced statements must be 'approved' by a consistency check. Second, the conflicts may be carried in during the construction of a problem solving theory $\mathcal{T}^*$ by instantiating a generic domain theory ***DT*** for a particular conceptualisation $\mathcal{T}$ (see definitions in section 6.1). While the generic domain theory may be consistent and sound, its interpretation in the particular conceptual terms may exhibit conflicts. A proof of a common occurrence of such logical conflicts in inventive problems can be found in (Altshuller 1984), and it was also discussed in section 6.3.3.

Another reason may be articulated around the fact that several axioms (components) may be believed at the same time, each being perfectly sound. Each of these 'axioms' may play role in a particular module or sub-system addressing a sub-set of objectives. However, when several such

[12] Note the deliberately different term to distinguish this process from the 'acceptability' check in section 6.1.2.

‘independent’ modules are involved in the solution construction, there may occur clashes and conflicts among them. Therefore, it is important to check the admissibility of the speculative abductions and partial deductions for consistency and compatibility.

A lot of research on this topic dates back several decades. This research produced a technique suitable for these purposes known as *truth maintenance systems* (TMS) (Doyle 1979). Basic TMS maintain the consistency through justifications; i.e. each and every new assertion of a theorem in a logical theory must be justified by a combination of already known (and valid) formulae. These formulae must be believed per se (i.e. they are axioms) or they must be justifiable by other previously accepted theorems. In a recursive manner, the validity of the whole base of assertions depends upon an agent’s belief of a small set of basic axioms and theorems. During theory construction or problem solving, certain formulae may be explicitly marked as ‘false’ (i.e. not believed or proved invalid). At this moment, the danger of having a contradictory theory is avoided by the capability of TMS to trace the justification chains. The theorems are found that depend on, and follow from the ‘falsified’ formulae. These potentially contradictory and unsound statements that depended on the original values of the modified formulae, are ‘removed’ (i.e. also ‘invalidated’). In such a manner, any amendment of an agent’s set of beliefs propagates through the problem solving theory, maintaining only the consistent beliefs until the soundness and consistency of the whole theory is ensured.

Several authors developed the TMS further; for instance, de Kleer (1986a; 1986b) came up with an *assumption-based truth maintenance system* (ATMS). His ATMS was more powerful and robust than the original version of TMS, and as such removed many weak points. De Kleer introduced so-called *contexts* that were bounded by the largest consistent sets of assumptions. An *assumption* was a novel concept of great importance in de Kleer’s theory. For the purposes of ATMS, an assumption is understood as a special kind of logical formula that expresses an agent’s *tentative belief* of some knowledge or fact (e.g. a default value of some parameter). If a contradiction is encountered that may disturb the consistency of a logical theory with assumptions, the sets of assumptions, and consequently the contexts that are constructed upon the assumptions, are re-grouped. Context maintenance ensures that within any single context an agent’s beliefs (and the deducible consequences) remain consistent. ATMS thus, handles the potential cases of inconsistency by separating the conflicting assumptions into several different contexts.

In the design terminology, suppose that we take some specific elementary structural blocks as the ‘believed assumptions’ of our design theory, in the sense in which ATMS understands them. All deduced consequences would be justified by some combination of these structural blocks and possibly other believed statements (‘proper’ axioms). Further, assume that one of the deduced consequences from a problem solving theory with these assumptions is contradictory and undesirable. ATMS may be able to split the assumed structures into two or more distinctive ‘contexts’ following a deduction of a formula that is in conflict with the explicit specification.

In other words, such a separation means that the basic building structures, as they were proposed initially, cannot co-exist, if the overall design solution is to be sound and admissible. The presence of inconsistent assumptions may be observed when these are due to performing several separate deductions from the mutually disjunctive sets of premises, in parallel. Nevertheless, ATMS can trace the hidden inconsistency back to the source through a perfectly straightforward, explicit operation. ATMS divides the incompatible modules or the sets of not consistent structures, so that the soundness of a problem solving theory is restored. Examples of an application of ATMS in design is described in deeper detail by Smithers, Conkie *et al.* (1990) or Tang (1997). Examples of explicitly conflicting requirements are discussed in detail in (Altshuller 1984) or (Sushkov, Mars *et al.* 1995).

6.4.4 Admissibility – compliance with constraints

The process of appreciating the admissibility of a candidate solution also involves a check of its compliance with the known constraints. Abduction and deduction discussed in the previous sections were mostly concerned with the explicitly *required* design objectives. Now, we devote some space to another important part of problem specification – *constraints*.

Constraints are defined as such conditions that must not be violated by the (partial) design solution under any circumstances (Wielinga, Akkermans *et al.* 1995; Motta and Zdrahal 1996). The concept of a *not violated* condition is strictly distinguished from the concept of a *satisfied* condition! A condition denoted as Λ below, is satisfied only if it evaluates to truth in a particular conceptual model. Nevertheless, the same condition Λ is not violated if at least one of the following situations occurs:

a. Condition Λ is computable in a certain context, and it evaluates to truth, or
b. Condition Λ is not computable at all in a certain context.

The scenario marked as a) above upholds the constraining condition and declares it 'not violated' by virtue of a positive proof that the condition is explicitly satisfied. On the other hand, the scenario marked as b) considers the constraining condition as 'not violated', because it cannot be proven explicitly that the condition is violated. It may not be computable; however, a kind of 'presumption of innocence' prevents the agent from rushing to premature conclusions about its *dis*satisfaction or violation. Thus, it is possible to imagine the constraints placed on a particular artefact as borderlines that separate generally *admissible* solutions from the *inadmissible* ones. Such a border may be violated, if the candidate solution is clearly 'outside' the designated space. Nonetheless, a border would not be violated if the constraint were not applicable in the context of a particular solution.

Just as there are many methods deploying abductive or deductive reasoning, there can be found numerous techniques for constraint satisfaction and validation (Kumar 1992). The computational power of constraints found its way into design, and there were several projects investigating various aspects of using constraints (Brown and Birmingham 1997; Zdrahal *et al.* 1997). A typical area for constraint satisfaction was constituted by well-defined problems that mainly addressed the configuration of designed artefacts. A typical representative of the tasks that are considered well-defined and that may be tackled by the explicit operational methods as presented in sections 6.4.1 to 6.4.4 is an elevator configuration (Yost 1992).

6.5 Place of RFD model in design lifecycle

The purpose of this section is to relate the RFD model presented in sections 6.1 and 6.2 to a general lifecycle of a designed product. This section shows, which phase of a design process the conducted experimental study focused on, and what issues the RFD model addresses. The schematic position of the research reported in this thesis, in the world of design, should help the reader understand particular settings and objectives of the experiment. The details of the experimental work are available in chapters 7 and 8.

First, we argue that a majority of the design approaches, as they were reviewed in chapter 4 and 5, address only the satisfaction of a given problem specification. Whereas our RFD model sees the 'explicit satisfaction' as one of three intertwined actions. These actions were expressed as predicates '*satisfies*', '*specifies*' and '*acceptable*' at the beginning of this chapter. The first predicate from this triplet – '*satisfies*' was illustrated in section 6.4, and as we mentioned, we did not investigate any of its operational or computational details. The schema shown in Figure 6–5 represents one of several models of such an explicit design process that can be found in literature. For instance, this representation takes into account different knowledge roles and knowledge sources in accordance with the theory of generic task models – GTM (Tansley and Hayball 1993).

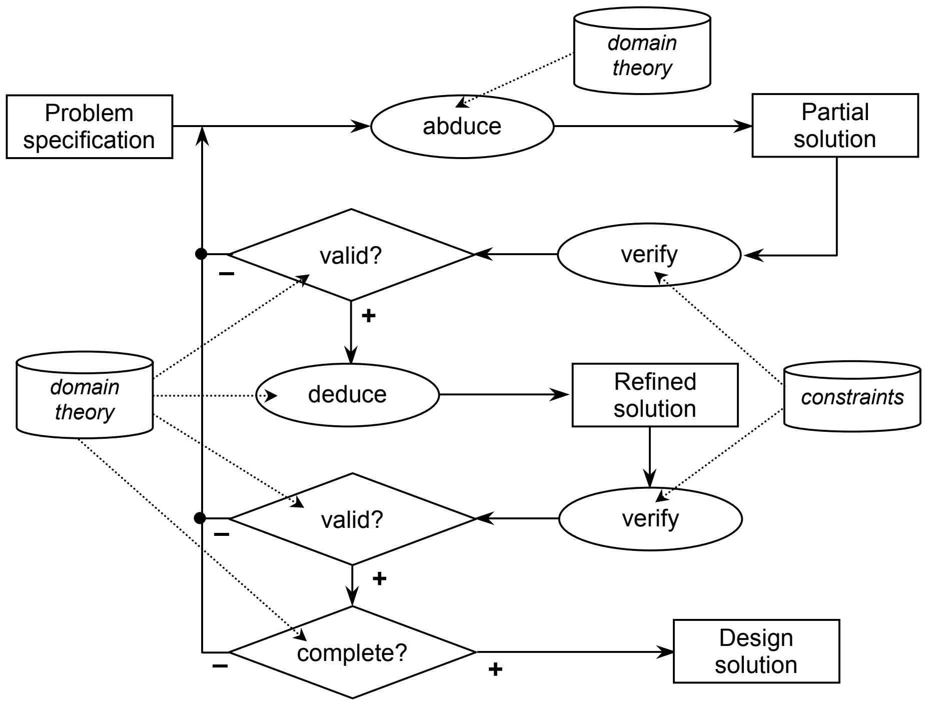
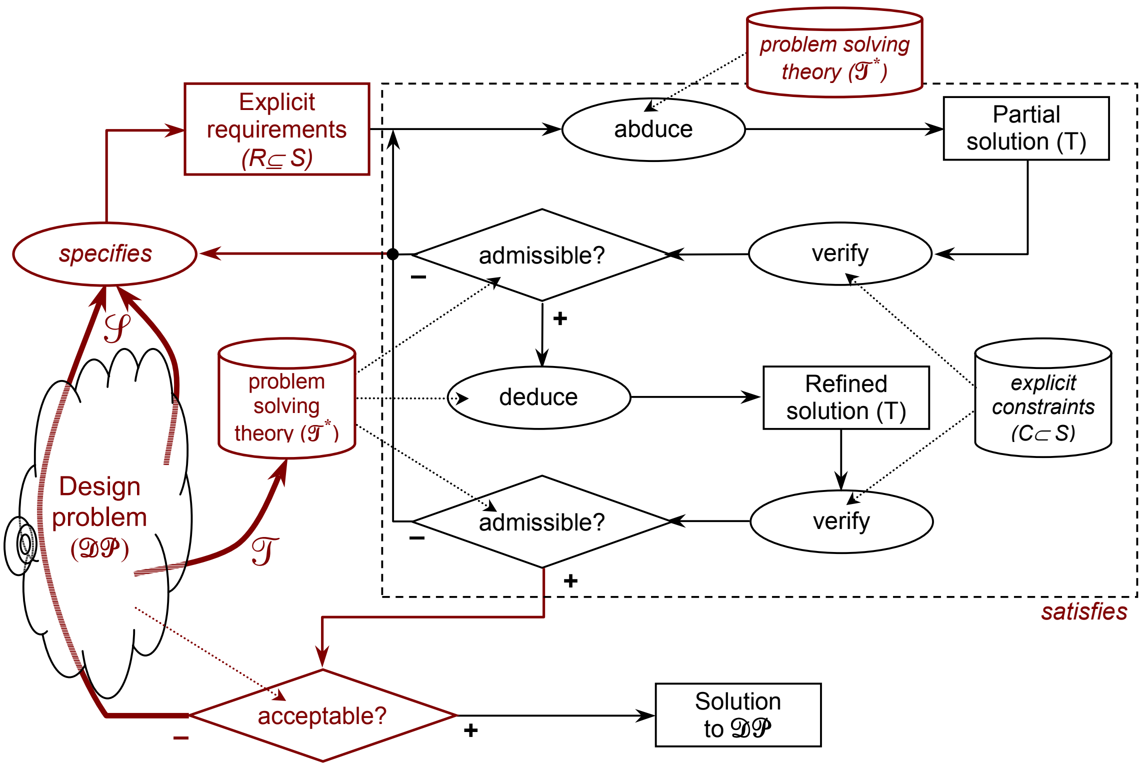


Figure 6–5. Generic task model of (explicit) design

Let us now extend the GTM accounting for predicate '*satisfies*' so that it accounts also for the remaining two predicates from our RFD model. There are several important changes to the GTM depicted in Figure 6–5. First, we distinguish between a design problem ($\mathfrak{DP}$) and its explicit specification (***S***). Consequently, we have to distinguish between a solution (***T***) to such an explicit specification (i.e. $\boldsymbol{T} \models \boldsymbol{S}$) and a solution to the actual design problem. Further, our extension of the above-mentioned GTM contains one additional test – namely for the solution acceptability: '*acceptable*$_{\mathfrak{DP}}$ *(T)*'. Only those candidates that pass this check successfully, can be considered 'design solutions' (i.e. solution to the original design problem).

If the acceptability check fails for any reason, it shows that the explicit problem specification (***S*** or $\mathcal{S}$) and/or the conceptual foundation of the applied problem solving theory ($\mathcal{T}$ and $\mathcal{T}^*$, respectively) is somehow lacking. Consequently, the explicit problem specification may be amended to reflect the encountered gaps. Alternatively, a conceptual foundation of the problem solving theory may be shifted, or in some situations both amendments of the sets forming a conceptual frame may take place.

The extension of the GTM of design to include the terms and concepts introduced by our model is shown in Figure 6–6. For better navigation, the parts that are specific for the RFD model and that are not present in the reviewed models are drawn with thicker lines (and in a different colour). The parts that are re-used from the literature are now 'grouped' into one operation that can be performed on the explicit level – the operation of satisfaction of explicit problem specification (see label '*satisfies*').



problem solving theory ($\mathcal{T}^*$)
Explicit requirements ($R \subseteq S$)
abduce
Partial solution (T)
specifies
admissible?
verify
−
+
$\mathcal{S}$
problem solving theory ($\mathcal{T}^*$)
deduce
Refined solution (T)
explicit constraints ($C \subset S$)
Design problem ($\mathfrak{DP}$)
admissible?
verify
−
+
$\mathcal{T}$
satisfies
acceptable?
−
+
Solution to $\mathfrak{DP}$

Figure 6–6. Place of RFD concepts in the explicit model of design.

Particularly interesting operations that were investigated during the experimental study are those interacting with the 'cloud' of a design problem ($\mathfrak{DP}$). We refer to these operations as re-framing or modification of a conceptual frame. Taking into account the definition of a conceptual frame as a pair of two conceptual sets – $\boldsymbol{\Phi} = \langle \mathcal{T}, \mathcal{S} \rangle$, the outcomes of the 'operation of re-framing' are the thick arrows leaving the $\mathfrak{DP}$ 'cloud' and labelled as $\mathcal{T}$ and $\mathcal{S}$, respectively.

According to the definition of the initial RFD model in section 6.1, the amended conceptual foundation $\mathcal{T}$ affects the articulation and deployment of a particular problem solving theory ($\mathcal{T}^*$), from which various candidate solution can abduced or deduced. Similarly, the amendment to the conceptual set $\mathcal{S}$ affects the explicit specification of the problem (i.e. $\boldsymbol{S}$ including the explicit requirements as well as constraints). This set is used for the verification of the admissibility of any conclusion inferred from a particular problem solving theory. Note that the dashed combo-box labelled as '*satisfies*' can consist of different building blocks that depend on a specific approach (e.g. logical design, design by analogy, case-based design). The 'insides' of such box is however, not the topic of this thesis.

Let us therefore, move towards the experimental work in which the theoretical claims presented in this chapter are validated by referring to the actual design scenarios. First, a general overview of the experimental work is given in chapter 7. The annotation and analysis of experimental sessions in terms of the RFD model, as defined in this chapter, follow the generic introduction and preliminary assessment in chapter 8.

7 BACKGROUND TO EXPERIMENTAL STUDY

In the previous chapters, we presented engineering design from several different perspectives and highlighted some interesting relationships between these perspectives. In chapter 6, a formalism was proposed for finding a place of 'inarticulate', reflective reasoning techniques in a design process. The *recursive model of framing*[13] in a design process using conceptual frames was presented as a natural extension of the usual approaches to working with explicit knowledge of design elements, algorithms and practices. However, the aspect of 'a natural extension' came to our attention with hindsight. It emerged from a number of initial experimental studies that were conducted to validate our expectations in respect to the theory of design and design frames (Dzbor 2001; Dzbor and Zdrahal 2001a; Dzbor and Zdrahal 2001b). The objective of this chapter is to acquaint the reader with the experimental background of the research.

From the reviewed literature on the subject of design and knowledge-based design support (see chapters 4 and 5) we learned several important lessons, and identified some of the outstanding gaps. A particularly strong argument that appeared in several sources claimed that design does not have to be necessarily seen as a kind of intractable artistry (Dzbor 2000a; 2000b). This position was supported by several experiments conducted by different researchers and reviewed earlier (Candy and Edmonds 1996; Cross 1997). The 'artistry' was related to a designer's expertise, deep knowledge of a particular field, ability to transfer and adapt the existing knowledge, to reflect and make less common decisions (Schön 1983).

However, most of the studies reported in the literature, either described design projects requiring a number of collaborating designers, or involved highly creative (inventive and unique) design tasks. In the first case, the focus of research was (understandably) shifted towards the issues of collaboration, communication, co-ordination, negotiation, and similar issues. None of these 'social aspects' of design is the primary concern of the work reported in this thesis. The studies from the second category are more informative, especially from the point of reflective reasoning. However, they focus on unique achievements, unique design situations that are not so common in everyday practice. Such studies may unintentionally widen the common misconception of many lay persons that the 'everyday' design practice is mundane and routine, whereas innovation and invention can occur only in a few, very rare, unrepeatable situations, typically involving a 'talented individual'. And such a misconception would lead us back to the point of a designer's artistry, which we aimed to explain…

7.1 General overview of experimental study

To address the identified gap in the empirical research of design we conducted a study with the individuals tackling rather complex, but not unique or exceptional design situations. The participants were design practitioners specialising in the design of controllers for large-scale systems. The problems the participating designers were asked to solve, were formulated so that they were manageable within a 12 hours long design session. At first glance, the majority of the problems seemed to call for rather standard, if not routine solutions. However, the design problems were specified so, as to allow for some freedom. It depended entirely on the designers which path would be chosen. We expected them to propose a solution that they would be willing to present to their peers and more senior colleagues. Formally, our assessment of the conducted experimental work is done in two steps:

1. validate the relevance of initial assumptions ('generic' assessment prior to any deep analysis)
2. validate the initial RFD model ('model-specific' analysis in terms of RFD theory)

The sequence of the experimental sessions and their analyses is best depicted in Figure 7–1. The first evaluation of the observed and recorded data is shown in the figure as an auxiliary box labelled '*assessment by a panel*', and was done by a panel and participants. The second step is addressed by the box labelled '*RFD analysis of annotated data*' and runs independently of the auxiliary pre-assessment. We expect that the RFD analysis yields rigorous results that would correlate with the subjective assessment by a panel, and allow a formulation of implications and/or amendments to the RFD theory.

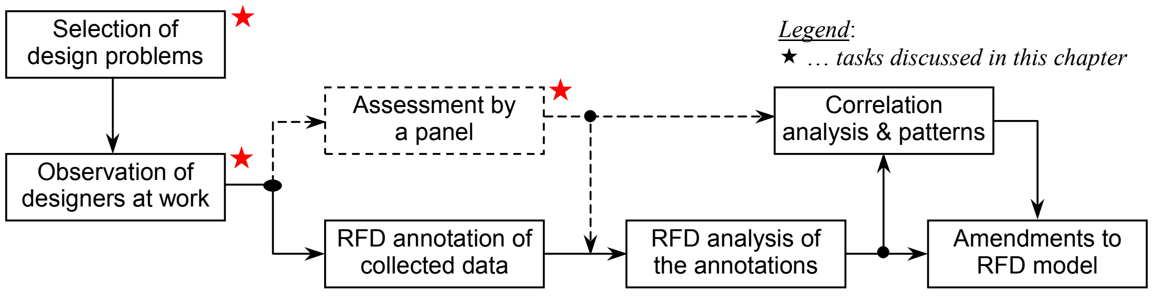

Selection of design problems
Legend:
★ … tasks discussed in this chapter
Assessment by a panel
Correlation analysis & patterns
Observation of designers at work
RFD annotation of collected data
RFD analysis of the annotations
Amendments to RFD model

Figure 7–1. Overview of the experimental study and analysis

It is the purpose of this chapter to acquaint the reader with the details of the experimental cases (section 7.2), chosen tools and their applications in tackling the design problems (section 7.3). In addition to the introduction and overview, the preliminary assessment of the design sessions by a panel of participating designers, domain experts and observers is provided in section 7.4. This sketches a broader picture of the experimental work. Following the preliminary, superficial assessment provided in this chapter, chapter 8 gives detailed RFD annotation and analysis of two experimental cases.

7.2 About experimental sessions

As already stated, in order to validate the expectations and theoretical ideas, we decided to conduct a set of experimental design sessions. In the preparation for this experimental study, we were fortunate to receive support and interest from one of the partners involved in a major European project that had Knowledge Media Institute as its co-ordinator. The representatives from this partner institution were keen to collaborate, and participate in the development of a coherent and realistic framework of reasoning in design. One of the incentives was surely the fact that the research project in question sought to follow a similar goal, with a greater emphasis on the collaborative and support tools. Another incentive for the collaboration was the partner's professional interest, and their involvement in 'everyday' design activities.

Eventually, we decided to conduct the experiment in the domain that was familiar and close to the author's background. Therefore, the domain of designing controllers for large-scale systems was

[13] As before, we also use the abbreviation '*RFD model*' –Recursive model of Framing in Design.

chosen as the main testing ground. Altogether, 24 design sessions were recorded; each tackling a specific problem of a controller or control strategy design. The assignments were not trivial, textbook examples but abbreviated extracts from real or close-to-real projects. The author's gratitude extends to the lab leader from the partner institution for his assistance in sharing his extensive experience from the large-scale design projects. This was re-used in the form of design briefs for the experimental sessions, and during the initial discussion on complexity of the experimental problems.

Each of the experimental sessions was given a unique identifier (e.g. *T21*) that was used in the correlation analysis, summary, and conclusions (see section 7.4). In addition to a formal identifier, each session was named by the main artefact that was to be controlled/designed. The details of all experimental sessions are available in Appendix A. Below, we present sample accounts of two problems that are annotated and analysed in-depth in chapter 8. This brief description gives a flavour of what can be expected from the next chapter. In the descriptions of the sessions, we look at the following 'characteristics' of the problems:

- expected primary goal(s);
- essential milestones of the designer's approach;
- quick description of the problem's character;

T11 … Active shock absorber (see also a more detailed account in section 8.2)

Primary goals:

- design of a shock-absorbing device that adapts to changing road conditions and driving styles
- measurement of disturbances caused by uneven road surfaces;
- selection of applicable devices and respective controlled/controlling parameters (spring vs. pneumatic tubes, hybrid combination);

Essential milestones:

- at the beginning, two different modes of operations considered – manual and automated adjustment of the shock dampening characteristics;
- problem attended to in two parallel 'contextual worlds', the difference between them growing with a refined solution;
- selection of a pneumatic approach where dampening characteristic depends on pressure rather than the actual material of the absorber (as was the case with a common spring);
- a set of assumptions formulated to clarify the requirement of 'active adjustment of chassis clearance' that forms a part of the initial problem specification ('active' in the context of chassis clearance is interpreted differently to that of shock absorption);
- eventually, a compromise is considered and different contexts are 'merged'

Problem character:

- task is clearly solved in several separate contexts (manual vs. automated tuning, shock absorption vs. clearance), and these are highlighted by totally different physical principles and components suggested in each particular case;
- presence of multiple contexts resolved by separate controllers for specific situations with a hierarchically superior 'co-ordinator' (a switchboard activating an appropriate sub-module)

T21 … Paper smoothing plant (see also a more detailed description in section 8.4)

Primary goals:

- design of a plant for achieving a vaguely defined functionality (smoothing raw paper);
- refinement of a vague specification with various quality- and process-related requirements (thickness reduction, prevention of paper damage, simple maintainability, etc.);
- selection of a suitable physical principle and subsequently layout of a plant

Essential milestones:

- early solution features a pair of rolling drums applying pressure on paper thus reducing its thickness and smoothing its uneven surface;
- in order to comply with qualitative concerns, pre- and post-processing units are added (dampen paper before rolling and dry it afterwards – lower danger of tearing);
- articulation of restrictive assumptions and additional constraints (e.g. regarding dimensions);
- re-formulation of a linear layout of a solution to a zigzag layout of the rolling drums;
- eventually, principle of rolling is considered in a broader context (re-interpretation of rolling from pressure application to abrasion → instead of drum pairs a zigzag layout of single rolls is sufficient for rolling);
- incorporation of pre- and post-processing units into feeding and driving units

Problem character:

- task starts relatively routinely, then becomes innovative (a sequence of re-formulations);
- simultaneous design of plant layout/structure and appropriate control strategy;
- oscillation between refinement of a vague specification and solution development;
- initial simple solution gets more complex as additional constraints and assumptions are considered; after re-interpretation the complexity is radically reduced

7.3 Methodological remarks

The experimental work was conducted in the form of several standalone design sessions, each of them tackling a particular design problem. All design tasks presented to the participating designers were taken from the domain of designing controllers for large-scale and complex systems. A typical assignment included the design of a control strategy, a control loop with the controller, or both. Brief details of each design problem are listed in section 7.2 and more in Appendix A.

All problems involved technical or technological systems. That is, we did not test the RFD theory on tasks involving artistic design, such as house interior design or architecture. Architectural design was a favourite research topic of many researchers in the past decade. A lot of high quality research was done in this area and can be found e.g. in (Schön 1983; Gero 1990; Fischer 1992; Gomez de Silva Garza and Maher 1996; Maher, Poon *et al.* 1996; Watson and Perera 1997). Neither did we include the sociological or economic design problems, such as described by Schön (1983). Primary attention focused on technological systems, devices and controllers. The referential literature includes, for instance, (Schön 1983; Altshuller 1984; Bylander and Chandrasekaran 1985; Smithers, Conkie *et al.* 1990; Iwasaki, Fikes *et al.* 1993; Bhatta, Goel *et al.* 1994; Choueiry, McIlraith *et al.* 1998; Cook and Brown 1999).

From the methodological point, we note that we were focusing on the early – or better, *conceptual* phase of design. The most serious implication of such a focus for the designers was restricting them to the use of sketchy drawings on paper, conceptual descriptions and justifications of their decisions in words. They did not have any powerful modelling packages at hand, such as Matlab/Simulink™ and CAD tools. However, unlike similar conceptual design studies (Cross 1997; Cook and Brown 1999; Cross 1999), our participants were given a computational tool to structure and record their reasoning on-the-fly. We admit that the presence of such a tool can be a double-edged sword; i.e. it may have both, positive and negative influences on a designer's performance (Davies 1995). Since the experimental settings are crucial for the validity of the results, let us first attend to the usage of the customised computation tool in the design experiments, and compare it to more common observational methods.

The advantage of the naturalistic and ethnographic methods is that the experiment is conducted in the setting that is natural for the participant (Rose, Shneiderman *et al.* 1995; Jagodzinski, Reid *et al.* 2000). In other words, the investigator is involved in the everyday activities of the participants, and typically takes the role of an external observer or interviewer. Different versions of the ethnographic approach include video and tape recording, note taking, looking over the participant's shoulder, or thinking aloud (Anderson and Crocca 1993; Kensing and Munk-Madsen 1993). All these methods could be applied on-line or off-line. In the former case, the experiment is made overtly, and the observers have an *active* role. They may ask questions immediately, when an unclear decision or an unexpected line of thought is observed. Alternatively, the observer works covertly, without disturbing the participants. In this case, the ambiguities are usually clarified at the end of the experiment, when the investigator goes through the notes, and interprets the observations with the domain experts. Very often, the direct observation is followed by an interview with a participant.

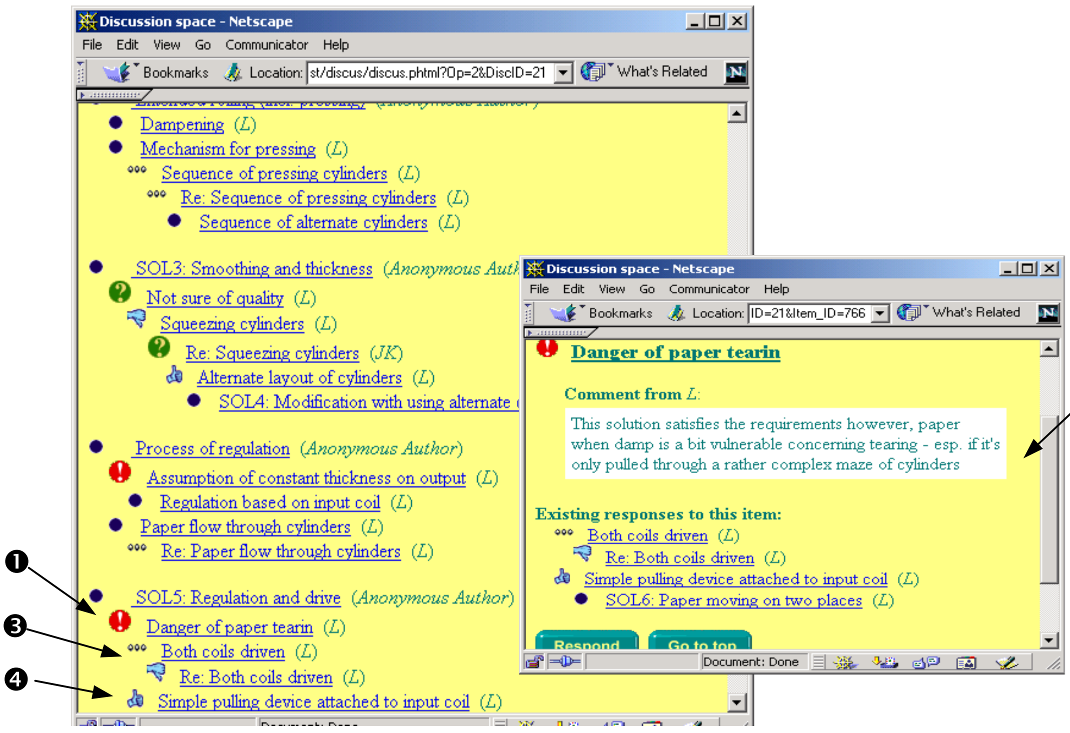
It is commonly accepted (ACM 1993) that the on-line methods are better, when it comes to the precision of the observation. However, because the observer directly interacts with the designer, and intervenes in the design, the validity of the observation may suffer (Davies 1995). The interventions of the observer might destruct the natural setting of the experiment, and in the extreme case, the recorded process may differ from the situation, when the designer was left alone. On the other hand, the off-line approach preserves the natural setting, and minimises the observer's interventions. However, because the ambiguities are discussed and justified after the session, the participant may produce a totally different line of reasoning to justify a particular decision compared to the one that actually occurred earlier during the design.

Therefore, we decided to take a combined approach to capturing participant's reasoning and justifications in the course of a design process. In practice it meant that we left the participant working on his or her own, without any direct intervention by the investigator. The designers chose their own pace, strategy, levels of granularity, etc. Simultaneously, they were asked to record their progress using a networked computational toolkit that enables the external observers to follow the explicit portion of their reasoning. Such a set-up also gave designers an opportunity to post a query or an interesting idea to the debating environment. From there, any ambiguous or otherwise interesting statement may be picked up by the observer, who can answer it, provide a comment on it, or leave it unnoticed.

This form of interaction was chosen after the initial interviews with the domain expert from the participating organisation. The leader of the group, from which participants were selected, claimed that they were used to interacting with their peers on an informal level, but preferred a more formal intervention from the senior staff, tutors or customers. When we later interviewed the participants and raised this point, they explained that formal interaction gave them the opportunity to both acquire the requested information and have a record of it for future reference.

We asked the participants to record their notes in a kind of on-line notebook. Each statement they entered was automatically organised into threads (Sumner and Buckingham Shum 1998), if it extended an existing (i.e. previously recorded) statement. A thread typically contained the reasoning step behind one design decision or a set of closely related decisions. It could be a decision concerned with a specification of the design task or a generation of the solution. New threads were typically started, when the designer's focus moved to a different module or extension of an existing module beyond the scope of the explicit context, as defined at the particular moment.

A snapshot of the environment recording the justified threads of a designer's reasoning is shown in Figure 7–2. The right-hand window shows details of one such justification. The left-hand window depicts sequences and threads of reasoning steps. A detailed transcript of this threaded record is available in Appendix D. Threading is a useful vehicle for tracing the development and extension of a particular reasoning chain. Figure 7–2 shows the situation when the designer discovered an unexpected flaw in his approach to smoothing a paper (details in section 8.4). The danger of damaging the paper was noted in the 'active' thread (see label ➊), and detailed in the small window (labelled as ➋). Next, a follow-up trying to fix the conflict appeared that was unacceptable (label ➌ and '*thumb-down*' icon). Thus, another – acceptable modification was proposed (see ➍ and '*thumb-up*' icon).


Discussion space - Netscape
File Edit View Go Communicator Help
Bookmarks Location: st/discus/discus.phtml?Op=2&DiscID=21 What's Related
➊
➋
➌
➍
Discussion space - Netscape
File Edit View Go Communicator Help
Bookmarks Location: ID=21&Item_ID=766 What's Related
Respond
Go to top
Document: Done

- Dampening (*L*)
- Mechanism for pressing (*L*)
  - Sequence of pressing cylinders (*L*)
    - Re: Sequence of pressing cylinders (*L*)
      - Sequence of alternate cylinders (*L*)

- SOL3: Smoothing and thickness (*Anonymous Auth*
  - Not sure of quality (*L*)
    - Squeezing cylinders (*L*)
      - Re: Squeezing cylinders (*JK*)
        - Alternate layout of cylinders (*L*)
          - SOL4: Modification with using alternate

- Process of regulation (*Anonymous Author*)
  - Assumption of constant thickness on output (*L*)
    - Regulation based on input coil (*L*)
  - Paper flow through cylinders (*L*)
    - Re: Paper flow through cylinders (*L*)

- SOL5: Regulation and drive (*Anonymous Author*)
  - Danger of paper tearin (*L*)
    - Both coils driven (*L*)
      - Re: Both coils driven (*L*)
    - Simple pulling device attached to input coil (*L*)

Danger of paper tearin

Comment from *L*:

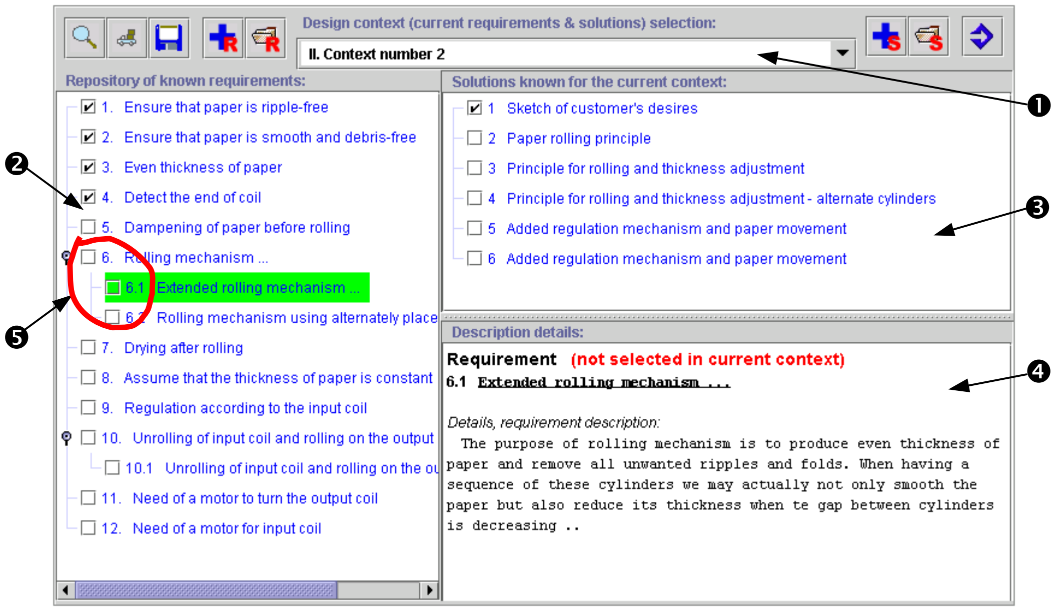
This solution satisfies the requirements however, paper when damp is a bit vulnerable concerning tearing - esp. if it's only pulled through a rather complex maze of cylinders

Existing responses to this item:

- Both coils driven (*L*)
  - Re: Both coils driven (*L*)
- Simple pulling device attached to input coil (*L*)
  - SOL6: Paper moving on two places (*L*)

Figure 7–2. Threaded justification of reasoning steps

Another tool that was available to the designers aimed to capture and structure the co-development of the problem specifications and solutions. In line with the RFD theory (see chapter 6), the tool depicted in Figure 7–3 defined the term 'context' (see label ❶) as a selection of explicit requirements or constraints (in the left pane, labelled ❷) articulated by the designer in respect to the designed artefact. The explicit commitments to a certain problem specification were usually accompanied by articulations of respective solutions addressing the particular design context or focus; see pane labelled ❸. 'Active context' was incremented automatically by the tool whenever the designer proposed a new (partial) solution. The explicit details of a commitment to a particular requirement or solution are displayed in the bottom right-hand pane (label ❹ shows details of requirement 6.1).



Design context (current requirements & solutions) selection:
II. Context number 2

Repository of known requirements:
- ☑ 1. Ensure that paper is ripple-free
- ☑ 2. Ensure that paper is smooth and debris-free
- ☑ 3. Even thickness of paper
- ☑ 4. Detect the end of coil
- ☐ 5. Dampening of paper before rolling
- ☐ 6. Rolling mechanism ...
  - ☐ 6.1 Extended rolling mechanism ...
  - ☐ 6.2 Rolling mechanism using alternately place
- ☐ 7. Drying after rolling
- ☐ 8. Assume that the thickness of paper is constant
- ☐ 9. Regulation according to the input coil
- ☐ 10. Unrolling of input coil and rolling on the output
  - ☐ 10.1 Unrolling of input coil and rolling on the ou
- ☐ 11. Need of a motor to turn the output coil
- ☐ 12. Need of a motor for input coil

Solutions known for the current context:
- ☑ 1 Sketch of customer's desires
- ☐ 2 Paper rolling principle
- ☐ 3 Principle for rolling and thickness adjustment
- ☐ 4 Principle for rolling and thickness adjustment - alternate cylinders
- ☐ 5 Added regulation mechanism and paper movement
- ☐ 6 Added regulation mechanism and paper movement

Description details:
Requirement (not selected in current context)
6.1 Extended rolling mechanism ...

Details, requirement description:
```
 The purpose of rolling mechanism is to produce even thickness of
paper and remove all unwanted ripples and folds. When having a
sequence of these cylinders we may actually not only smooth the
paper but also reduce its thickness when te gap between cylinders
is decreasing ..
```

Figure 7–3. Design context capture

The snapshot of the tool in Figure 7–3 shows design problem T21 annotated and analysed in section 8.4. It was made at the moment when the designer returned to a previously attended context II, after articulating several additional extensions. Thus, the designer was able to keep several lines of enquiry open at the same time – but grouped into separate 'parallel' contexts. As the design progressed, the lists of explicit requirements and solutions grew accordingly. New requirements could extend the 'old' ones (e.g. 6.1 is a refinement of 6 – see also pointer ❺), or address a new feature/issue (e.g. 7 or 8). A detailed transcript of records of such a co-evolution between specifications and solutions for design session T21 is presented in Appendix C and its annotated version in section 8.4.

Further details on the deployed capturing tools can be found in (Dzbor 2001; Dzbor and Zdrahal 2001a). Let us therefore mention only the main advantages of the applied methodology. The utility of these two tools was that they collected data about design reasoning in a structured form, and simultaneously provided the observer with a discrete opportunity for intervention. The intervention could be masked as a design comment or recommendation within a threaded sequence. It fitted well into the work style the designers were acquainted with. They were not distracted and did not have to stop their reasoning to answer a query from the observer. They could go on with their development, and answer the query, when more suitable. This would be impossible, if the observer's interventions were formulated verbally and presented directly in person (as in the more usual approaches).

Taking into account the 'non-invasive' collection of data, the automation of threading and context creation we would like to argue that the danger of imposing the investigator's opinion onto designers was minimised. The recordings of the sequences of reasoning steps constituted the primary source of data for the analysis. However, to elicit participants' subjective opinions regarding the toolkit intuitiveness or user friendliness, post-session 'interviews' were conducted. During these debriefing conclusions, each session was summarised, with the intention to highlight the main 'milestones' in the interpretation of the design problem.

'Debriefing' also played an important role in explaining the positions of various accompanying drawings and sketches in the explicitly recorded sequence of reasoning steps. This post-processing phase was needed because of the usage of multiple media for capturing the designer's reasoning; including paper notebook and pencil. 'Debriefing' thus merged data from the separate sources into a single design case. It included the location of exact spots of where the designer moved from words (computational tools) to drawings on paper, or vice-versa. The information about these 'interruptions' or shifts in reasoning was useful especially for the validation of the RFD model of design process.

7.4 Assessment of experiment prior to RFD analysis

Assessment of the recorded design sessions by a panel of experts was conducted in order to validate the selection of the design tasks for the experiment. Another purpose was in validating the feasibility and correctness of the initial assumptions of what features typically characterise design as a problem solving activity. Thus, the panel was asked to assess the collected data records (prior to any RFD analysis) focusing on scope and criteria as detailed in section 7.4.1 below. Section 7.4.2 provides a brief summary of these subjective opinions in a tabular form. Later in chapter 9, these subjective opinions are compared with the results yielded by the RFD analysis, which is described in chapter 8.

7.4.1 Scopes and criteria of assessment by a panel

From the first category of 'generic' objectives validating the basic assumptions about the nature of design, we choose the following main sub-categories (these are explained in more depth below):

a) Presence of the phenomenon of co-evolving design specifications and solutions;
b) Reflection on the partial design solutions (and solution candidates);
c) Re-formulation of design problems (incl. articulation of additional requirements/constraints);
d) Presence of knowledge transfer featuring re-use of the familiar 'frames'

Combining different perspectives on design, as they were reviewed in sections 4.1 and 5.1, we concluded section 4.3 with a list of typical features that in our opinion characterised design as a reasoning process. The purpose of the objectives summarised in this section is to validate such a perception of design. These generalised statements form a foundation on which our RFD model is constructed. Therefore, it is important to validate these features on a rather generic level before making any conclusions about the actual RFD model and the implication it may have.

The list below serves to refresh the memory as it repeats the facets of the design process from section 4.3. The subsequent paragraphs associate the above-mentioned four categories of objectives with the respective features, and provide a context in which these are investigated. These are the aspects we believed characterised a ‘typical’ design activity:

- Design develops new artefacts that satisfy certain desired goals.
- Design is inherently an iterative process.
- The understanding of the ‘desired goals’ is not static.
- Design relies on the exploration of consequences through conjectured experiments.
- Design uses a variety of knowledge sources including a designer’s previous experience.
- Analogy and familiarity are important vehicles for ‘seeing-as’ and ‘doing-as’.
- Multiple parallel lines of enquiry are typically attended to during design.
- In addition to rigorous assessments, a ‘tacit feeling’ for the situation is also a driving force in the non-trivial designs.

Let us refine the four categories of ‘generic’ objectives of the experimental study:

a) Co-evolving specifications and solutions

Objective a) emphasises the hypothesis that problem specification (incl. various requirements and constraints) evolves in parallel with a design solution. First, this means that design is iterative; a sequence of reasoning steps and decisions amendments would be observable. Second, the phenomenon of co-evolution means that the iteration is not necessarily driven by a completeness of a problem specification. Iterative steps may be opportunistic and exploratory, and may be conducted to acquire deeper understanding of particular behaviours, unexpected results, etc.

In other words, we expect a richer interaction between the problem specification and solution construction than can be explained from simple operational models of design (such as ‘waterfall’ mentioned in section 4.1.2). In many models, the amendment of a problem specification is only allowed if there are explicit violations by a proposed solution. However, we argue that in addition to the ‘explicit’ evaluation there are also ‘internal’ evaluations – a kind of ‘exploratory probes’. Thus, we expect to observe such ‘probing’ through co-evolution of specifications and solutions.

b) Reflection embedded in design

This objective subscribes to the view that reflection is an attempt to uncover and rectify the unexpected and/or undesired behaviours through ‘exploratory probing’ (see also point a) above). What we want to observe is that internal or ‘tacit’ evaluation occurs continuously as the development of a design solution and specification progresses. This form of assessment would address the partial solutions and assess them not for their logical soundness or completeness, but ‘acceptability’ or ‘desirability’.

Thus, design would exhibit ‘checkpoints’ where solution ‘acceptability’ is checked in addition to explicit truth and consistency maintenance. We can compare this situation to a practice of software developers. Any piece of code, which passes through a compiler or interpreter, is ‘sound’ and may be assessed as a developmental ‘build’. On the other hand, not all ‘builds’ enter the evaluation phase. Only those that are ‘acceptable’ to the developer may act as a ‘released version’, which can be presented to the external world. This objective aims to observe an element of such ‘tacit check of solution acceptability’ as contrasted to the solution admissibility.

Reflection may occur with incomplete artefacts or solutions, for instance, for the purposes of testing a particular 'probe', assumption or approach. In such scenarios, reflection would aim to validate and refine the existing problem specification that may be vague, incomplete or 'ill-defined'. Thus, reflection would typically lead to a re-formulation of a design problem.

c) Re-formulation of the design problems

In similarity with point a) above, this objective addresses the assumption that problem specification in design is subject to development, refinement and/or extension. The same design problem taken from a different angle may be expressed as a presence of certain dynamics in the designer's understanding and interpretation of it. Such dynamics would be manifested through discoveries and articulations of new conceptual terms specifying the design problem as well as the concepts for a construction of solutions. We do not expect that any of these two conceptual sets would be fully defined and enumerated during the design process.

On the contrary, the designer is expected to attend to the selected sub-set of the problem specification and propose partial and conceptual solutions verifying the relevance and completeness of the particular focus. We expect that the work in a particular context would yield both expected and unexpected results. Subsequently, the unexpected and/or undesired features need to be explained and rectified if possible. However, new requirements, assumptions, constraint and/or structural concepts may need to be brought into the design in order to provide such explanation. In other words, the design problem may need to be re-formulated or re-interpreted. Problem re-formulation is investigated from the perspective of co-evolution and reflection as introduced in the preceding paragraphs a) and b).

d) Knowledge transfer and familiarity

This objective aims to observe the application and re-use of familiar vocabularies when structuring (framing) the design problem. Familiar or past design cases, experience and analogy-based reasoning are expected to be important knowledge sources and reasoning strategies, and they would also serve the purposes of interpretation and (re-)formulation of the design problems. Following from the literature review in sections 5.1 and 5.2, it is reasonable to ascribe such importance to the analogy- and similarity-based reasoning.

In general terms, we seek a flexible application and re-use of different knowledge sources, as well as being able to familiarise oneself with the current problem using the previous experience. This expectation would require the presence of such operations as retrieval of familiar cases or adaptation of a re-usable component. We believe that various design prototypes, patterns or models of the devices, and the articulation of additional conceptual primitives by analogy would dominate over rules and algorithmic procedures. Referring to points a) to c), we also expect that such prototypes and familiar scenarios would be useful for the (re-)interpretation of the design problem.

In accordance with the cognitive perception of design (see section 4.1.3), the design solutions may often originate in the transferred knowledge. A transfer may occur between the similar cases both tackling the same issues (so-called '*in-domain transfer*'), or it may be inspired by an interesting case that is similar on a very abstract conceptual level (i.e. '*cross-domain transfer*'). Regardless, we expect designers trying to transfer chunks of previous experience to refine or extend both design specifications and solutions.

Assessment criteria

The following criteria were selected – first, to characterise each experimental session, and second, for a simple correlation analysis. The list below defines the criteria as 'pairs': 〈*feature evaluated*〉 {list of possible values}:

- 〈*Overall impression*〉
  values = {routine, partly innovative, innovative}
- 〈*Form of problem specification*〉
  values = {detailed, complete without details, functional outline, vague functionality}
- 〈*Dominant source of solution*〉
  values = {prototype-based, in-domain transfer, cross-domain transfer}
- 〈*Conceptual abstractness*〉
  values = {high-level abstract, conceptual, implementation}
- 〈*Amendments in problem specification*〉
  values = {frame amendment, re-formulation, extension, clarification}
- 〈*Prevailing operational method*〉
  values = {analysis, synthesis, balanced A/S}

The criterion 'overall impression' looked at a broad picture of the particular design task. The evaluation covered all aspects, i.e. designed artefact, re-use of previous knowledge, novelty of the solution, presence or absence of usually expected and unusual decisions during the design, etc. A task was considered routine, when no unusual decisions were made, or the design followed a well-described, standard algorithm. Partial innovation meant that in some aspects, the design exhibited an uncommon feature or knowledge transfer occurred. Innovative problems contained a significantly unusual approach and corresponded to major improvements of the existing control systems.

Under the criterion 'form of problem specification', a completeness of the initial design brief was assessed. The four values reflected the increasing vagueness, starting with a complete specification that contained all the facts and details needed to proceed with the design. Next, it was a specification that contained basic conceptual requirements but lacked the details (e.g. dimensional data or exact chemical composition). A more abstract specification was labelled as a 'functional outline', roughly describing what was expected from the designed artefact in functional terms. The problem specification marked as 'vague' permitted several alternative interpretations, and only broad hints about the expected functionality and performance were given.

In the category 'dominant source of solution', the prevailing knowledge source used to develop and refine the solutions was assessed. That is, the different types of knowledge transfer and re-use that might occur in design were in focus. Therefore, the first group contained those cases, where rather clear prototypes existed, and these did not require any extensions or amendments. The second group was marked as 'in-domain transfer', and it contained the cases, where an analogous approach on a medium conceptual level was observed. Previous cases would be typically taken from the same or conceptually similar problem domain (e.g. heat propagation was analogous to air conditioning). Finally, 'cross-domain transfer' labels those that occurred on a very high conceptual level; the analogy was 'inspirational', and not common (e.g. measurement of electric resistance using the ball pressure instead of the conductivity of ball material).

The category ‘conceptual abstractness’ looks at the main focus of the design. The designed product might have been developed on a highly abstract level, with little attention given to the structural details of the high-level building blocks. Alternatively, the designer might also use and develop medium-level conceptual terms, and attend to the selected issues of implementation. The details of realisation were limited to qualitative concepts, such as safety, user comfort, or user friendliness. Finally, the most detailed solutions would go well down to the level of system implementation, attending in great detail to the selected qualitative and also quantitative criteria of an optimal performance.

Further, the experimental sessions were assessed according to the amount of changes required in order to formulate a specification reflecting the actual product being designed. At one extreme, a significant re-formulation took place, and the task needed to be completely re-interpreted. On the medium level, a limited amount of additions or minor amendments was observed that did not change the principal points in the overall ‘frame’. Finally, the opposite extreme was marked as ‘clarification’, and included those tasks, where no or very little re-formulation was necessary. Typically, the amendments to the problem specification in this category were restricted to a translation from the customer’s language to the language of an engineer.

The last category attended to the dominant operational method. The panel was asked to comment, whether the particular problem forced designers to draw upon their analytical knowledge, and approach the tasks as an analysis, or whether they applied a constructional, synthetical attitude. An alternative between these two poles was a ‘balanced interplay’ of analysis and synthesis. For instance, new analytical knowledge was obtained through a synthesis of a partial artefact/component, or a specific part/ component was added based on the analysed behaviour of an incomplete artefact.

7.4.2 Summary of the assessment by a panel

As mentioned in section 7.4.1, each experimental session was assessed by six criteria. Each criterion provided several values for the description of a particular session in respect to that criterion. The sessions were judged only from the captured data; i.e. electronic design protocols, raw records, and paper-based resources (drawings and sketches). The assessors did not have any access to the RFD analysis (chapter 8). The panel consisted of design students, practitioners and theorists (plus the author), and covered a range of expertise in engineering design and/or controller design. Two evaluators were university graduates with a major in control systems, one was a design practitioner with a significant empirical experience. Also consulted were three professionals with a general overview of the design research. And to complement this group, a cognitive scientist was asked to comment on more general aspects that did not require any specific domain expertise.

The assessments were compiled into tables that served as a basis for the correlation analysis. When compiling the judgements from various people, we counted the individual responses and the ‘panel response’ was determined by the most frequent judgement. When two values had the same or very close occurrence, both were accepted as valid scores. Such ambiguities were expectede from the ill-structured problems we were investigating. For instance, design task *T11* (shock absorber, see also section 8.2) was assigned the value ‘incomplete functional outline’ for the criterion ‘form of problem specification'. However, the same task has two values – 'in-domain transfer' and 'cross-domain transfer', when evaluated in respect to the source of solutions (see Table 7–1).

A dual valuation typically occurred when two (or more) distinctive parts of a design task were observable, and all of them were equally important. For instance, an interesting reasoning step could appear through an inspirational, high-level transfer of previous knowledge. This particular shift of a perspective was very important for the delivery of the 'final solution'. However, the task as a whole was judged differently. A typical example is task *T3* (ball and horizontal rail), that exhibited a typical prototypic application of resistance measurement, but at the end, a more innovative and unusual one superseded the prototypic approach. Nonetheless both solutions – routine and innovative were still acceptable.

The following table shows a correlation between the selected criteria that were used for evaluation by a panel. From the perspective of this thesis, Table 7–1 is of particular interest. It shows the relationship between the different criteria for the ill-structured design problems. The highlighted 'cells' in the table represent a re-occurring pattern in assessment. Where two 'cells' of the same attribute are highlighted, it means that value A or value B or both were observed. Such a device exposes the already-mentioned blurred boundaries between the individual values and the 'fuzziness of the assessment'. For example, if values 'functional outline' and 'vague outline' are highlighted, it means that either of them was typically accompanying 'partly' or 'significantly' innovative tasks.

Table 7–1 shows coincidences of the following assessments that typically emerged from the panel's collective opinion. This coincidence is shown in the table using orange background (🟧):

- Overall design process: potential for 'partial' or 'significant' innovation
- Problem specification: 'functional outline' or 'vague'
- Solution 'source': 'in-domain' or 'cross-domain knowledge transfer'
- Conceptual details of solutions: 'higher' or 'moderately abstract'
- Changes to problem specification: 'significant amendments' or 'extensions'
- Operational method: 'analysis and synthesis in a mutual interplay'

A complementary selection described by the statements below is shown in the same table using bright green background (🟩). Typically, this coincidence highlights the complementary values as compared to the orange ones (Table 7–1). However, strictly speaking, the coincidences are not exclusively disjunctive. There are cases conforming to both patterns, for instance case T3 – ball and horizontal rail strongly exhibited both routine and innovative patterns. Nevertheless, such an overlap only confirms the fact that the coincidence rule is based on the usual and typical experience. The overlapping assessments are depicted with a striped background (▨). This complementary co-occurrence (in 'green') takes into account the following assessments:

- Overall design process: potential for 'routine' or 'little' innovation
- Problem specification: 'complete' or 'some details missing'
- Solution 'source': 'prototype re-use'
- Conceptual details of solutions: 'moderately abstract' or 'implementational'
- Changes to problem specification: 'minor amendments' (clarifications)
- Operational method: 'more synthesis' than 'analysis' (little interplay)

Taking into account relative occurrences of the individual valuations we can say that in some cases, there existed a clear 'winner', whereas in others, the differences were less striking. For instance, a majority of tasks was judged as routine (37%) or only partly innovative (41%). A similar distribution over a much larger base of cases is reported in (Altshuller 1984). According to Altshuller, only a fraction of all inventive designs is groundbreaking; most are only minor or medium improvements. Next, in our experiments, the initial problem specification usually fell between the 'functional outline' (38%) and 'not detailed but otherwise complete' category (35%). This shows that it is rare to have a completely specified problem, or absolutely vague and abstract specifications. A 'moderate' level of specification details seems to be more suitable for innovation than an absolute vagueness. Too detailed specification may restrict the freedom of exploration; on the other hand, vagueness may complicate the first approach, the formulation of the first conceptual frame.

The majority of cases exhibited either a direct transfer (39%) or an adaptation of an existing prototype (43%). A clear winner in the criterion judging the conceptual details of solutions was a 'medium level' (64%). These two observations also support Altshuller's claim about the rarity of designs that may alter the highly abstract conceptual foundations of a specific field.

Most tasks upheld our assumption of a co-evolving problem specification and solution; typically new design requirements were added to or extended the initial specification (44%). Most problems were tackled by an interplay between analysis and synthesis (46%), which supports the school of thought represented by e.g. Oxman (1990), Smithers, Conkie *et al.* (1990), or Logan and Smithers (1992). As mentioned in section 4.1, trying to split a task into analysis and synthesis is often trying to differentiate the indistinguishable. Even in the prototypic cases, the distinction was not an easy.

Table 7–2 shows a summary of the opinions of the panel in respect to the quality and complexity of the designed product. The first category assesses the designed product, and the values range from the design implementing a routine controller and focusing on the control algorithm, through customised controllers, and ending with designs of both controlled systems and controllers. In the category of design completeness, the allowed values were: a single detailed solution, a single conceptual solution, more alternative solutions, or highly abstract conceptual ideas. The rules for ticking the appropriate cells were similar to those for Table 7–1.

This second table will be used later in chapter 9 to show interesting patterns in the analysed results. The blue background of selected cells (🟦) covers those tasks that were judged as generally 'algorithmic' and yielded rather detailed though simple solution. For the interpretation, see section 9.2.

However, we stress that these percentages and relative co-occurrences were obtained prior to any rigorous analysis in terms of our RFD theory. Therefore, we shall return to these subjective assessments later, after analysing the collected data by the RFD theory and model. Model feasibility would be supported by a discovery that the tasks highlighted in orange in Table 7–1 are those that exhibit more radical shifts and amendments in the respective design frames. A similar finding will be presented for the two 'categories' distinguished in Table 7–2 (blue vs. white backgrounds).

The details of the RFD analysis can be found in section 8.1. This and subsequent sections in chapter 8 also provide examples of how the collected data captured in the form of threaded records (see Appendices C and D) was interpreted in English and translated into the language of RFD theory. The conclusions are drawn in terms of the RFD model being a realistic explanation of the collected and annotated data.

Table 7–1. Correlation between: more innovative approach ≈ ill-specified problem + transfer of solutions + co-evolving specification & solution

Task	Category	overall design process			problem specification				source' of solution			conceptual details			prb. specification changes			operational method		
		routine	partly innov.	innovative	complete	no details	outline	vague	prototypic	in-domain	cross-domain	high abstract	medium	implementation	significant	addition	clarification	analysis	balanced	synthesis
T2		x				x			x					x			x			x
T3		x	x		x			x	x		x		x			x	x		x	
T4			x			x			x				x			x			x	
T5		x				x	x		x					x			x	x		
T6				x		x				x			x		x				x	
T7			x					x		x				x			x		x	
T8		x				x			x				x			x		x		
T9			x				x				x		x			x			x	
T10			x					x	x	x				x		x			x	
T11			x				x			x	x		x			x			x	
T12		x				x			x				x			x				x
T13		x			x				x				x	x			x	x		
T14		x	x				x			x			x			x			x	
T15		x	x				x			x		x					x	x		
T16				x		x			x		x		x		x			x		
T17			x			x			x				x			x				x
T18				x			x			x		x			x				x	
T19		x			x				x				x				x			x
T20				x			x				x		x		x				x	
T21				x			x				x		x		x				x	
T22				x			x			x		x			x			x		
T23			x					x		x			x		x			x		
T24			x				x			x			x			x		x		
T25		x				x			x	x		x				x				x
Counts		37%	41%	22%	12%	35%	38%	15%	41%	38%	21%	16%	64%	20%	28%	44%	28%	33%	46%	21%

Table 7–2. Panel assessment in respect to the complexity, completeness, and quality of the designed product

Task	Category	*designed product*			*solution completeness*			
		control algorithm (simple controller, clarified system)	*controller (new device, system refinement)*	*extension of controlled system*	*one detailed*	*one conceptual*	*conceptual alternatives*	*high-level ideas*
T2		x				x		
T3		x	x		x		x	
T4		x				x		
T5		x			x			
T6			x			x		
T7			x			x		
T8		x			x			
T9		x	x			x		
T10			x		x	x		
T11			x		x		x	
T12		x			x			
T13		x			x			
T14		x	x			x		
T15				x				x
T16			x				x	
T17		x			x			
T18				x		x		x
T19		x				x		
T20		x			x			
T21				x			x	
T22		x				x		
T23			x		x			
T24			x		x			
T25		x					x	
Counts		***52%***	***37%***	***11%***	***39%***	***36%***	***18%***	***7%***

7.4.3 Remarks on assessment of user acceptability

As mentioned in section 7.3, the use of a particular tool and the methodology may influence the observations. This cannot be avoided; however, the real threat of potentially biased results can be assessed. As we mentioned earlier, the main objective of the experiment was to find out the details of how designers do their job. Nonetheless, the acceptability by the designers of the experimental settings may also be an interesting criterion for the evaluation of the tools. The remarks in this section are not directly relevant to the assessment by a panel or the RFD analysis (see chapter 8). They only summarise the elicited opinions of the participants on how close to the 'natural settings' the experiments were, and on the benefits of such tools. To that extent, a direct feedback was sought from the participants after they completed their tasks but prior to the RFD analysis.

After initial apprehension, the participants welcomed the idea of recording their decisions and progress. As expected, compliance with the suggested methodology placed an additional demand on their time. However, when a session took more than one day, the benefits outweighed the time invested into justifications. Designers A expressed this shift along the following lines. "*If someone wants to continue a well-advanced design, and still keep to the original ideas, it makes a lot of sense to retrieve it in its full and original state. I can continue where I stopped yesterday or a week ago without having to go through the entire procedure just to remember what I meant by this sketch or that note.*"

Particularly useful was the opportunity to record both the formal 'milestones' of the explicit problem specifications and solutions, and the informal thoughts. This was also a rather novel approach, compared to various existing CAD or modelling tools. According to one participant, the informal justification gave him often more information than the list of formal records or the actual design solutions. All designers expressed their opinion that such an approach could be useful for the people practising design that are not experts. By cross-referencing the formal records with the 'informal' statements, the essence and the background of a particular decision could be elicited much easier. The threaded records also helped with learning and re-using the contextually embedded 'best practices' for how design may be conducted in a particular domain.

The need for informal justifications and exploratory probes may seem redundant for the narrowly specialised practitioners. However, our participants thought that an interesting idea often emerged while they articulated the justifying records. Typically, the sequence of explicit statements triggered their awareness of the features they may have overlooked earlier. It seems that the threaded 'justification' cross-referenced with the explicit design has an 'added value' when compared to other creative methods, such as brainstorming (Thompson and Lordan 1999). This finding also supports the complementary nature of formal and informal knowledge records we subscribed to in section 3.2.4. Informal, 'tacit' awareness indeed helps to uncover interesting corollaries between the explicit records.

In order to make the methodology usable in practice, it was suggested to make it more customisable by adding a choice of additional features such as searchable case archives, sketching tool and so on. Lack of support for sketching was perceived as a major drawback of the experimental setting. As many justification records show, a sketch or schema were frequently referred to by the participants. They used sketches not only as solution blueprint but also for a simple, conceptual simulation or an illustration of certain behaviours or principles. We acknowledge this need and agree with Vinod Goel (1995) that sketches are in certain professions more important than any ‘verbal’ languages. It would be very interesting to see how they influence design frame amendments or reflection; unfortunately, this theme lies beyond the scope of the reported research. It may perhaps form a topic for further research. For example, one such ‘practical’ extension is currently investigated in a research project running at KMi[14]. More about future extensions is presented in concluding chapter 12.

Nevertheless, the overall user’s acceptability of the tool and the underlying methodology can be evaluated as positive. It was very encouraging to see that participants found the methodology helpful for particular conceptual phases of design. They all underlined that the tool could not replace the reasoning and creativity of human designers, but it definitely showed a potential to give assistance in learning good practices and structuring the knowledge about a design problem. Thus, we may say that the observations from the experimental study are in line with our initial expectations.

The driving force of the whole research was articulated in terms of ‘reflection’, ‘explicit and tacit knowledge’, and ‘design frame amendments’. In chapter 6, we proposed an initial model of such a recursive process using conceptual frames. Let us therefore, present the annotation and analysis of the experimentally collected data in the terms of the proposed RFD model. The next chapter provides details of two experimental sessions together with a summary of the conclusions drawn from all the experiments.

[14] For details, see the information on Clockwork Joint European Project funded by the European Commission under the contract number IST 12566. Project’s web site is currently available on URL http://clockwork.open.ac.uk.

8 ANNOTATION AND ANALYSIS OF EXPERIMENTS

The purpose of this chapter is to acquaint the reader with the experimental work that was carried out to address the issues raised after the review of relevant literature in chapters 4 and 5. The raw data collected during the experimental study are available in the respective appendices (Appendix C and D). In sections 8.2 and 8.4, it is shown how the raw data was interpreted as sequenced design episodes and reasoning steps interspersed with the explicit proposals of (partial) solutions to the particular explicit specifications.

After annotating the collected data in the form of a narrative, which is more reader-friendly than the raw data in Appendices C and D, the two tasks are analysed in sections 8.3 and 8.5. The analysis is conducted in terms of the theory and model presented in chapter 6. Each of the two analytical sections is concluded with a summary of 'reasoning episodes' that were identified and annotated in the interpreted data from the two experiments. These summaries are then used in chapter 10 for a further extension and elaboration of the initial theory of conceptual framing in design.

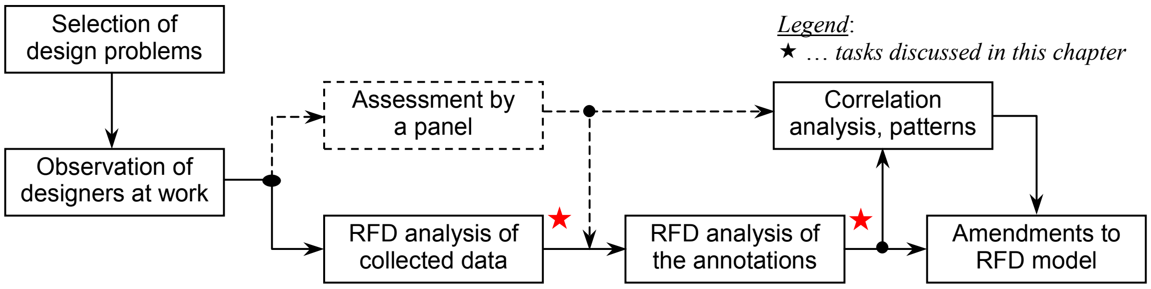

Selection of design problems
Observation of designers at work
Assessment by a panel
RFD analysis of collected data
RFD analysis of the annotations
Correlation analysis, patterns
Amendments to RFD model
Legend:
★ … tasks discussed in this chapter

Figure 8–1. Phases of the experimental study tackled in this chapter

As we did in the previous chapter, Figure 8–1 shows which phases of the experimental study are tackled in this chapter. For an easier cross-referencing, the individual episodes have assigned particular design contexts (taken from Appendix C) and a set of relevant justification threads (corresponding to those in Appendix D).

8.1 Principles for the annotation

Before presenting the RFD analysis of the annotated experimental scenarios, let us define 'measures' and principles for the annotation. These 'measures' represent the knowledge-level actions that were introduced as a part of the RFD model in chapter 6. Their chief purpose is to provide a succinct account of each analysed session in the form of a simple table that will relate the individual design episodes, the frequency of occurrences, and the order of the investigated actions. The same 'measures' are used also in chapter 9 to show and discuss the aggregated results from all the experiments, as well as interesting patterns that re-occurred in the experimental sessions.

First, let us clarify the vocabulary that is used during the annotation and analysis. The term **design task** or **design session** both refer to one particular design problem that was tackled by the designer over a certain period of time. Each design session started with the presentation of a **design brief** to the designer. The brief was written in a customer's language, and its purpose was to express the essential requirements and expectations from the designed product. These briefs were prepared in advance by the investigator, in consultation with a panel of experts.

Each design session lasted for several hours, and typically contained several explicit ‘milestones’. The designer articulated such a milestone in order to record and emphasise his **explicit commitment** to a particular problem specification and/or partial solution. One significant result of each explicit commitment was a definition or re-definition of a **design context**. A design context always consisted of a list of selected (attended) **requirements**, **constraints**, **assumptions**, and other statements that clarified the designer’s position in respect to the problem interpretation. In addition to the explicit specification, each context may contain a (partial) **candidate solution**. The evolution of the explicit commitments and respective design contexts is transcribed in Appendix C.

However, the records of the evolving and changing design contexts are too coarse-grained, and only show the design problem at the moment when a designer made an explicit commitment. The design contexts do not record any reasons, justifications or alternative choices that were available to the designer. Therefore, in order to complement design contexts, we use the term ‘**design episodes**’. These episodes consist of one or usually more **reasoning steps**, and their purpose is to provide the transitions between the explicit commitments to particular design contexts. Typically, there were two design episodes between two consecutive design contexts at the early stages of a design session. Toward the end of the session, there was usually one episode bridging the two explicit commitments.

Each design episode described a meaningful standalone portion of the designer’s reasoning. Usually, we associated the episode with a particular thread or sub-thread from the justification records. Nevertheless, we took into account not only the syntax of the recorded threads but also their content and focus. Thus, a single design episode was typically concerned with the elaboration of a particular viewpoint, analysis and comparison of the alternative viewpoints, product decomposition and development of the sub-components. In the identification of ‘boundaries’ between the design episodes we also took into account the order in which the justifications were recorded. For a typical transcript of the ‘raw’ justifications, see Appendix D.

Finally, the **reasoning steps** of a design episode corresponded to the ordered selection of the knowledge-level actions that are presented in Table 8–1. These actions followed from the review conclusions in chapter 5 and the RFD theory introduced in chapter 6.

Table 8–1. Definition of reasoning steps forming a design episode

Reference	Verbal description of referenced action
1a	Articulate design frame (set $\mathcal{S}$ and $S \subseteq \mathcal{S}$: *specifies(S, $\mathfrak{DP}$))*
1b	Articulate design frame (set $\mathfrak{T}$ and/or $\mathfrak{T}^*$)
2	Retrieve similar problem/frame ($\phi_{SIM} = \langle \tau, \theta \rangle$)
3a	Knowledge re-use – articulate *S*, $\mathcal{S}$ similar to θ
3b	Knowledge re-use – articulate *T*, $\mathfrak{T}$ similar to τ
4	Is re-used/adapted *T*/$\mathfrak{T}$ and/or *S*/$\mathcal{S}$ *admissible*?
5	Find candidate solution ($T \subseteq \mathfrak{T}^*$: *satisfies(T, S)*)
6a	Develop *S*, $\mathcal{S}$ into details (incl. alternatives, refinements)
6b	Develop *T*, $\mathfrak{T}$ into details (incl. alternatives, refinements)
7	Is candidate solution *T acceptable*?
8	Re-interpret conceptual frame (i.e. $\Phi \rightarrow \Phi'$)

Let us briefly recapitulate how we assigned the individual reasoning steps to the annotated design episodes and contexts. Reasoning steps *1a* and *1b* were assigned to the episode that featured a formulation or a selection of design objectives/objects. Typically, these two steps represented an explicit commitment to the particular requirements, constraints, and conceptual primitives. Usually, these steps were associated with an articulation and recording of a design context.

Reasoning step *2*, and consequently *3a* and *3b*, represented any reference the designer made to the previous design cases, applicable domain theories or design models/prototypes. These reasoning steps typically occur together as knowledge retrieval and re-use/adaptation. 'Keywords' that can be found in the justifications include for instance, '*from the theory we know*', '*in order to create a typical control loop*' and similar. Reasoning step *4* was concerned with the explicit assessment of admissibility. An admissibility check was assigned to the episodes, in which clear criteria existed against which the designer could evaluate his partial solutions, compare alternative approaches, or check the compatibility of the re-used/adapted design knowledge.

Reasoning step *5* corresponded to an attempt to articulate an explicit solution, solution model or a partial solution. Typically, this step was accompanied by a drawing or sketch depicting the essential components, behaviours or functionality of the (partial) solution. In similarity with steps *1a*/*1b* this step was typical in the records of explicit commitments and design contexts.

The next two reasoning steps *6a* and *6b* correspond to further development of a particular design context and commitment. They may include the elaboration of existing (partial) solutions, further clarification of design objectives, and various deductive refinements in the problem specification or solution description. They differ from steps *3a*/*3b* in the fact that no specific reference is made to past knowledge, nor do any new conceptual terms appear in these 'refinements'. Unlike steps 1a/1b that merely state the facts explicitly, reasoning steps 6a/6b tend to show the refinement of such facts in the form of causal or deductive chains.

Reasoning step *7* was assigned to those episodes that exhibited certain signs of evaluation, although there were no specific and clear criteria defined at the beginning of the episode. Typically, the new criteria/objectives emerged from the episode featuring this *acceptability* check. Finally, step *8* is our reference to a potential perspective shift. In other words, when new concepts or criteria emerged in the design episode as a result of reflecting on the previous designs or experience, we assigned the episode this *re-framing* step. Often, frame shift or amendment was accompanied or followed by an introduction of new assumptions or preferences and a tentative commitment to them. Thus, we may see step *8* as the creation and application of an exploratory probe.

Let us begin with the annotation of the design session T11 in section 8.2, where we split the design session, and show several design contexts with various design episodes connecting them. Then, in section 8.3, we shall interpret the annotated design episodes and explicit commitments in terms of RFD theory and the reasoning steps we defined earlier. The same strategy is repeated for design session T21 in sections 8.4 and 8.5.

8.2 Annotated task T11 – active suspension

The initial problem specification or design brief given to the designer for this particular problem reads as follows. *Suspension and shock absorption are crucial elements in the car design, particularly with regard to the passengers' comfort. Design an active shock absorber and suggest a control strategy, so that the whole suspension mechanism would be able to adjust its properties according to the current state of the road. In addition to the absorption of shocks caused by an uneven road, we would like to adjust also the chassis clearance, so that the car has the optimal aerodynamic coefficients for a particular type of surface.*

Design episode 8.2a
Initialisation of frame DC-I using similar conceptual frames
Design context: DC-I Justifying threads: J-489 – J-494

The designer received a design brief as shown above and began with an initial task analysis that was based on his knowledge of existing approaches to car suspension. First, he expressed the term 'active suspension' using three tenets in a language similar to that of the initial problem specification {J-490}. Defining and clarifying the terms and customer's requirements, he also introduced certain criteria to evaluate whether the desired objectives have been met.

In the justifying threads appears an issue of what can be considered 'active' when referring to car suspension. These notions were important because they were recorded allowing the follow up of any conclusions reached at the end of the initial stage {J-491 to J-494}. The initial analysis focused on two features that can be both 'active' and 'passive' – chassis clearance and shock absorption. As records numbered J-492 and J-493 show, the designer framed the problem of 'active' suspension in terms of a modifiable shock absorption and constant chassis clearance {J-491}. The initial frame led to the formal expression of the goals as listed in DC-II below.

Design episode 8.2b
Development of DC-I and the emergence of basic concepts
Design context: DC-I Justifying threads: J-495 – J-502

The initial framing of the design problem interpreting the requirements from the design brief led to an articulation of an 'ideal' solution {see notes J-495 and J-496}. The ideal solution was formulated as a suspension system whereby a car driver would not feel or observe any unevenness of the road surface. This ideal state was further translated to the formal language of design as witnessed in J-497 to J-502.

At this stage, the designer defined basic conceptual objects of the respective design frame that he intended to use for a solution construction. For instance, a decision has been articulated that proposed using a softer suspension for uneven surfaces and a tougher one for smooth roads {J-497}. Another concept resulting from the initial framing was a monitor of the surface unevenness and a processing unit that would modify the shock absorber's properties accordingly {J-498}. In addition to an automated response, the processing unit was required to respond to the explicit demands of the driver, hence the concept of 'manual adjustment' {J-499}.

Although the initial frame focused on the constant chassis clearance, the designer also articulated basic conceptual terms for adjusting the clearance. However, in comparison to the conceptual terms defined for the shock absorption, these were less precise {J-500 to J-502}.

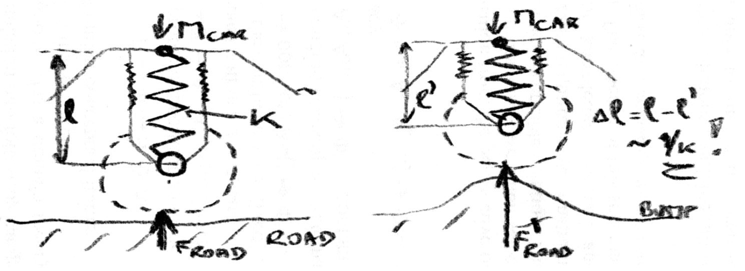


Figure 8–2. Behaviour of a prototypic shock absorber

Design episode 8.2c
Retrieval of applicable components and the move towards the first principled sketches
Design context: DC-I (+ Figure 8–2) Justifying threads: J-503 – J-507

At this stage, the designer expressed the concepts identified in the initial frame in a sketch as depicted in Figure 8–2. The sketch showed how an absorber reacted to an uneven surface, and it helped to introduce more conceptual objects to the attended design frame. For example, a displacement of the wheel against the chassis was noted as a typical reaction to the car entering an uneven surface (Δl). This observation was also noted down as records J-503 and refined in J-504. A new concept emerging from the sketch re-defined a generic concept of 'monitoring unevenness' {J-498}. The new method of monitoring comprised measuring the displacement of the wheel axle against the chassis.

Another concept that emerged in the sketch was a helical spring acting as a familiar and simple shock absorber prototype. However, despite the familiarity of the spring, it was not suitable for designing a sound solution. According to J-498 and also J-504, the shock absorber's toughness had to be modified in response to an immediate state of the road surface. A spring as a mechanical component did not exhibit the desired flexibility and simplicity for this.

Nevertheless, the spring as a prototype was instrumental in articulating what exactly had to be amended in order to change the shock absorption properties of a device. The characteristic the designer became aware of after using a spring as a prototype was 'toughness constant K' and its influence on the quick shock elimination (see also the next episode).

Design episode 8.2d
A new frame emerges 'informally'
Design context: DC-II (+ Figure 8–3) Justifying threads: J-508 – J-513

As already mentioned in the episode 8.2c, a new requirement has been articulated in terms of amending the toughness constant K. Since with a spring, the constant K depended on its material (e.g. steel), the designer looked for other materials, where this constant could be more easily modified. A frame containing similar conceptual behaviours as the current one but different conceptual objects was brought forward in the form of pneumatic/hydraulic devices – pistons.

A simple pneumatic piston-based shock absorber was sketched (see Figure 8–3) to replace the old spring-based prototype (shown in Figure 8–2). The shock absorption properties of the piston depended on the pressure of the substance inside (e.g. air). Since air pressure was easily varied by pumping in more air or releasing some from the piston, the designer had a sound device for achieving the objectives recorded informally in the frame DC-I. Next, he recorded a new explicit frame DC-II.

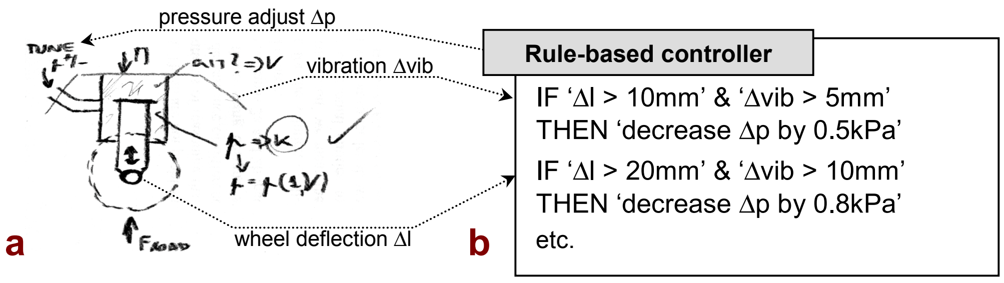

pressure adjust Δp
TUNE
vibration Δvib
Rule-based controller
IF 'Δl > 10mm' & 'Δvib > 5mm'
THEN 'decrease Δp by 0.5kPa'
IF 'Δl > 20mm' & 'Δvib > 10mm'
THEN 'decrease Δp by 0.8kPa'
etc.
wheel deflection Δl
a
b

Figure 8–3. New prototypic device (a), and the first control loop – solution (b)

<u>**EXPLICIT ACHIEVEMENT 8.2/DC-II**</u>
First explicit context with a solution is formally recorded (DC-II)

Let us recapitulate what has been achieved so far. First, the designer articulated the initial specification of the problem in the following terms (see records R-1 to R6 recorded within DC-II):

– the system should be able to absorb shocks {R-1}, and it must do so automatically depending on the current surface {R-1.1}
– the system should be able to adjust the chassis clearance {R-2} but for simplicity we assume a constant clearance whilst driving {R-2.2}
– the system should be able to measure the displacement of the chassis against the road and/or wheels against the chassis {R-3, R-4}
– the system should be able to adjust dynamically the shock absorber's properties, especially its toughness constant K {R-5}
– the system should contain a processing unit able to calculate a suitable reaction to a particular value of the wheel/chassis displacement and accordingly amend the constant K {R-6}

The concepts for constructing a solution have been discussed in the design episodes 8.2a to 8.2d so they shall not be repeated here. However, the first milestone was accomplished by articulating the solution S-1 (see also note J-514). The first solution in the form of a rule-based control loop is presented in Figure 8–3 sketch B, and its behaviour is discussed in the thread starting with J-505.

<u>***Design episode 8.2e***</u>
Proposal of an exploratory probe in a new frame – re-addressing the concept of 'activity'
Design context: DC-II (solution S-1)   Justifying threads: J-514 – J-517

From the notes following the articulation of solution S-1 {records J-515 to 517}, there was a visible emergence that the first iteration was not entirely acceptable in respect to the 'ideal state' {J-496 earlier}. The designer's comment J-515 suggested that the reason for not accepting S-1 as a fully-fledged solution, might be in its inflexibility. It took into account only two inputs (displacements of wheels and chassis), and based on these, the air pressure was reduced to achieve a softer suspension.

Also, the system was not able to perform a reversed action, i.e. returning to a higher pressure. The designer became aware of the fact that not only an uneven surface might have an adverse impact on the driver's comfort but so would a too soft suspension; the car would start 'oscillating'. Therefore, he proposed addressing this newly discovered requirement by amending and extending the simple control algorithm from context 8.2/DC-II {see also J-516 and J-517}.

The amended algorithm took into account the history of measurements done on the wheels and the chassis, so that when the initial shock has been eliminated, it starts increasing the pressure. The designer named this strategy as 'try & see' – which we believe underlines its heuristic nature. He also formulated a few new concepts that could be useful in the prediction of the type of 'shock' (e.g. a car entering an uneven road having a different ordering of events as a car entering a smooth road).

Design episode 8.2f

Another probe – re-addressing the chassis clearance (initial assumption of its constancy)

Design context: DC-II (with solution S-1) Justifying threads: J-518 – J-527

In a similar manner to that described in design episode 8.2e, the designer was not satisfied with the restriction of constant chassis clearance. He claimed it was helpful to produce the initial solution, but it negated his understanding and definition of 'active suspension'. Thus, he addressed this inconsistency in the threaded records starting with J-518 and ending with J-527.

The designer opted for a familiar prototype for controlling a particular variable (in this case chassis clearance). First, he established a referential value (for a stationary, empty vehicle). The displacement was measured after loading, and the individual wheels were appropriately balanced. Then a table was drawn up associating the measured shock on the chassis with the concept of 'terrain types'. To each terrain, a recommended chassis clearance was assigned, which was to be maintained by a controller.

New requirements (or re-specified and re-addressed requirements) triggered an articulation of new conceptual objects from which sound solutions could be derived. One such new concept included a kind of 'screw-jack' (see sketch in Figure 8–4) that could be combined with the pneumatic piston known in the previous frame; this would implement an 'active suspension' {J-521}.

Design episode 8.2g

Exploratory probes accepted, move to a new context DC-III (incl. new concepts for implementation)

Design context: DC-II, DC-III Justifying threads: J-528 – J-537

Before implementing the two major extensions to the context DC-II, the designer also refined his knowledge of the concepts that were already known in that context. He focused on the articulation of physical and engineering principles behind the respective functional requirements and on refining abstract concepts (see design episodes 8.2b to 8.2d). The brevity of the records and the selected concepts reflect the designer's decision to use the existing modules for these purposes if possible.

Thus, he proposed measuring chassis vibration and its displacement against the road surface using a technique similar to that for activating airbags {J-528, J-529}. For measuring the displacement of a piston from some initial position, he proposed a resistance-based measurement. Also, potentiometer was the selected means for measuring even diminutive changes in incremental displacement {J-530 and J-531}. Records J-532 and J-533 re-iterate the pressure modification in the pistons; the designer added an alternative of using oil instead of air (thus broadening previous frame and introducing the hydraulic components). Figure 8–4 shows the parameters measured on the system labelled as 'IN'.

In addition to refining the 'old' concepts (from DC-II), the designer proposed the same level of conceptual details for the newly-emerged concept of the chassis clearance amendment. In this case, the prototypic incremental motors drove the screw-jack and move it up or down {J-536, J-537}. These motors are commonly used in controlling motion in the engineering.

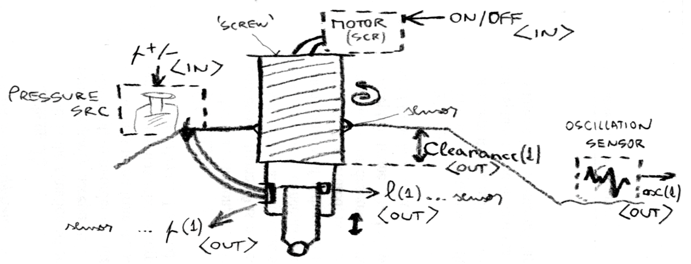


Figure 8–4. Extended candidate solution with new concepts and details

<u>**EXPLICIT ACHIEVEMENT 8.2/DC-III**</u>
Solution significantly extended and refined (DC-III)

As it is possible to observe in the explicit problem specification that served as a basis for this context, two statements appear refining those already formulated in context 8.2/DC-II. In addition, two new requirements were added. Below is the list of both refinements and extensions:

– the system should be able to respond to both an immediate state of the road surface and explicit demand of the driver {R-1.3} → refines {R-1, R-1.1}
– the system should adjust the clearance of the chassis when safety conditions allow (e.g. vehicle is stationary, steadily moving, not turning, braking, etc.) {R-2.3} → refines {R2, R-2.2}
– the system should contain a specific unit for controlling and adjusting the clearance, as well as appropriate 'action body' realising the adjustment {R-7}
– it is necessary to co-ordinate the controls coming from the suspension controller and the clearance controller to ensure smooth and safe operation {R-8}

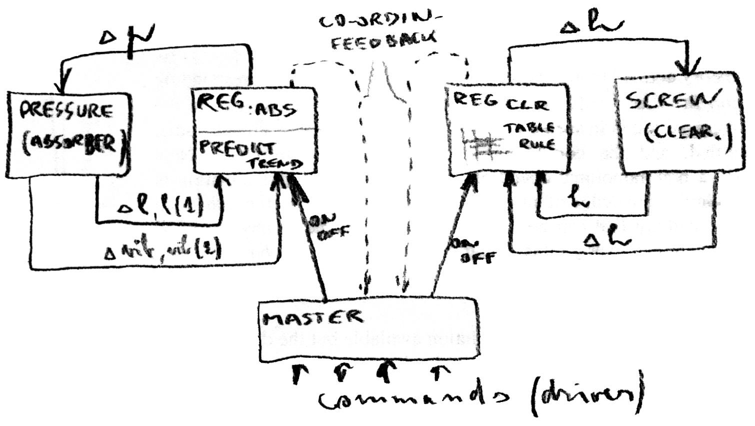


Figure 8–5. Another solution with a co-ordinating master controller

As an embodiment of the above-presented explicit interpretation of the problem, the designer constructed a new solution that contained the informally proposed new conceptual objects in addition to those brought in earlier. The solution was referred to as S-2 in the decision sequence records, and it served as a basis for further investigation as visible from thread starting with record J-538. A sketch of the two control loops, featuring also the newly-introduced concept of a co-ordinating loop are depicted in Figure 8–5.

Design episode 8.2h
Discovery and rectification of the unacceptable co-ordination
Design context: DC-III (with solution S-2) Justifying threads: J-538 – J-541

The solution mentioned in the context 8.2/DC-III was constructed to remove the unacceptable inadequacies of the previous iteration. As the follow-up records to J-538 show, the designer did not accept S-2 as a final 'design solution' yet. From J-539 it seems that the designer was not satisfied with the co-ordination of the two distinct control strategies.

The candidate solution S-2 was more complex than S-1, and included several different modes of operation. The designer tackled the emerged complexity in notes J-540 and J-541, in which he made more specific commitments about the interactions between different modes, and suitable interfaces conveying the information to/from the driver. For instance, he introduced such concepts as driver overriding the automated control or 'turning off' the clearance adjustment under specific conditions. These newly-articulated requirements addressed the flaws that emerged from the current solution. Therefore, they rendered the candidate solution S-2 unacceptable and inconsistent, thus requiring further elaboration and development.

Design episode 8.2k
Consolidation of the extensions and refinements to the previous frames
Design context: DC-III Justifying threads: J-543 – J-546

The fundamental concepts differentiating the new frame from the previous ones were 'interface' and 'man-system interaction' {J-543}. The designer articulated the main functional requirement that needed to be added to the system in J-544. The functional purpose of an interface was to 'translate' the intuitive and qualitative values suitable for a driver to quantitative values recognisable by the suspension controllers. In the subsequent notes {J-545 and J-546}, the designer articulated the necessary concepts for implementing the new functional requirement.

First, he distinguished an 'overall mode' of operation, such as choosing and switching between fully automated and 'manual' adjustment of the chassis clearance and the shock absorbing pistons. Another important extension was in the movement from purely quantitative controllers to fuzzy controllers. The main feature of this new type of controller was its capability to work with ranges rather than crisp values. Ranges could be associated with suitable 'qualitative' terms such as 'hard' or 'medium' for the suspension, and 'low' or 'very low' for the chassis clearance.

As seen in the records J-545 and J-546, this new conceptual approach benefited also the individual controllers and the control algorithms. The rules could have been written more robustly and the overall behaviour was expected to become more flexible and less prone to harmful oscillations or destabilisation. For instance, the designer pointed out that the new approach enables better implementation of his original 'try & see' heuristic {J-516 and J-517}. Thus, formulating a new requirement and attending to it, helped to resolve and improve the system's performance also in respect to another issue.

<u>EXPLICIT ACHIEVEMENT 8.2/DC-IV</u>
New conceptual approach implemented as a solution (DC-IV)

After identifying the potential issues and elaborating them, the designer proposed yet another refinement in his iterative construction of a design solution. The only new requirement that emerged in the explicit specification of the problem in context DC-IV was regarding the interface:

– system should feature an interface for mediating between different modes of operation and driver's immediate choices {R-9}

The solution associated with this particular context did not feature any major structural changes. The designer recorded it as S-3, and in J-547 noted that it was similar to S-2 with a different internal implementation. He re-iterated that the most important addition would be a '(de)fuzzification' table associating the ranges of measured values (e.g. piston displacement, chassis oscillation, pressure, etc.) to and from the qualitative concepts (e.g. 'small', 'low', 'high', etc.), see also Figure 8–6.

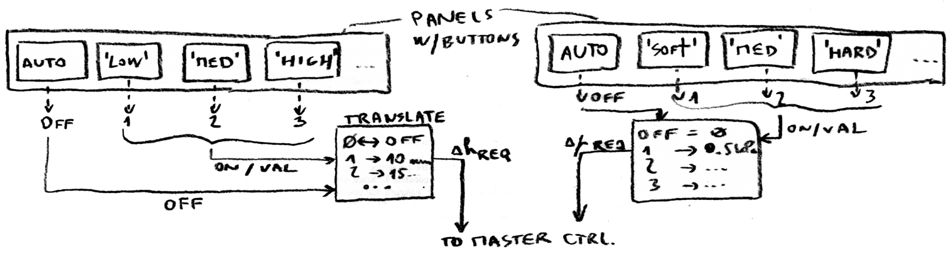


Figure 8–6. 'Interface' for communicating driver's requests

<u>EXPLICIT ACHIEVEMENT 8.2/DC-V</u>
Partial component re-visited (DC-V)

In addition to the refinement of an overall solution and the construction of a new, more robust solution (S-3), the designer returned to a minor issue left from the early phases of design. For instance, he looked at the actual (physical) inputs from and outputs to the controlled system, and their mutual relationships. Once again, it seems that he tackled two issues while attending to the explicitly articulated problem of interactions between the individual controllers. In this very last refinement to his solution, he clarified and formalised the interaction between the different sub-elements responsible for the individual measurements. Moreover, by doing so, he actually refined the previous vague articulations of the principles of measuring the desired values. The 'solution' resulting from this last frame was encoded in the form of control flow typical for control engineering.

As the designer explained verbally, this was only a different re-coding of the previous solution. He admitted that this form was much better for establishing the relations and interactions than any of the previously used. The designer finished his task noting (verbally) that this schema was a reasonably complete and acceptable conceptual solution to the problem he was given. Further work already required building a prototype. The physical outputs were shown in Figure 8–4.

DESIGN TASK ENDS BY PROPOSING A SOLUTION

8.3 RFD analysis of task T11

This section aims to summarise each of the episodes presenting partial narratives of the design story in terms of the experimental objectives and the vocabulary of the RFD model from chapter 6. Thus, each episode is associated with an appropriate design context and design frame. Some are clearly embedded in a single frame (e.g. 8.2c or 8.2d), whereas others serve as a transition from one frame to another one. Typically, the second scenario appeared in the cases when the original frame was found insufficient or incomplete, and a need arose to articulate new conceptual terms (e.g. 8.2b or 8.2e).

The narrative flow of the design story is interspersed regularly by major 'achievements'. These achievements express the explicit commitments of the designer to a particular specification and all contain a solution associated with them. The designer recorded such 'milestones' when he decided to commit himself to certain conclusions that typically arose from the justifying threads. Thus, there is a visible relation between the two forms of talking about the same design problem. Explicit achievements represent separate design contexts or frames without any causal relations among them. The narratives serves as a lead capturing certain decisions or chains of decisions that shift the design from one frame to the next one. In some cases, the narratives are longer (e.g. between 8.2/DC-II and 8.2/DC-III), which may suggest that the designer spent more time developing the next step or decided to merge several discovered issues together. Other sequences of narratives are shorter, which may suggest an immediate verification of the proposed resolution of the discovered conflict.

Design episode 8.2a

Design episode 8.2a serves as an initialisation of the problem solving space. It takes as an input the design brief – we may label this using our symbol $\mathfrak{DP}$. Broadly, the reasoning steps performed by the designer were categorised as a retrieval of familiar frames (past design cases) that were similar to the current problem. The retrieval was followed by the articulation anf selection of the relevant concepts that helped specify the current problem $\mathfrak{DP}$. In other words, we believe the purpose of design episode 8.2a can be expressed as our relation '*specifies(S,* $\mathfrak{DP}$*)*' (see also section 6.2).

Reasoning step(s) observed in episode 8.2a:	
	Retrieval of a familiar frame (action #2) Articulation of S, $\mathcal{S}$ similar to a familiar frame (action #3a)

Design episode 8.2b

The second chain of reasoning (design episode 8.2b) can be seen as a development of the initial frame. Thus, it features a commitment to certain explicit specification ***S*** (goals, objectives of the session). However, unlike 8.2a, this episode already introduces the conceptual primitives that could be used to implement a solution satisfying the given specification. In the terms of our theory, this chain explicitly articulates the conceptual elements $\mathcal{T}$, and the appropriate knowledge space $\mathcal{T}^*$ – a problem solving theory for this design frame. We also noted that some of the concepts were only introduced, others were also elaborated into deeper details.

Reasoning step(s) observed in episode 8.2b:	
	Frame articulation – specifies (S, $\mathcal{DP}$) (action #1a)
	Articulation of **T**, $\mathcal{T}$ similar to a familiar frame (action #3b)
	Further development of articulated frame $\langle \mathcal{T}, \mathcal{S} \rangle$ (action #6a, 6b)

Design episode 8.2c

Design episode 8.2c further extends the approach started in 8.2b. However, unlike in the previous narrative, first conceptual sketches appear at this stage – prototypic solution or in language of our theory, one potentially applicable problem solving model $\boldsymbol{T} \subseteq \mathcal{T}^*$ is explicitly articulated. This articulation happened in terms of a familiar design frame (spring as a body of the shock absorber). The specific commitment to a particular problem solving theory enabled further refinement of the problem specification (***S***) and conceptual foundation ($\mathcal{T}$) – e.g. specific physical quantities were specified. First time in the design process, we observed the search for an admissible problem solving model $\boldsymbol{T} \subseteq \mathcal{T}^*$ trying to satisfy the explicit problem specification developed so far (see predicate '*satisfies(T, S)*' in chapter 6).

Reasoning step(s) observed in episode 8.2c:	
	Retrieval of a familiar frame (action #2)
	Articulation of **T**, $\mathcal{T}$ similar to a familiar frame (action #3b)
	Further development of articulated $\mathcal{T}$ (action #6b)
	Re-using a familiar component (action #5)
	Is re-usable component admissible? (action #4)

Design episode 8.2d

The informal proposal of a solution candidate in episode 8.2c was followed by the assessment of consistency. At this stage, the designer uncovered the first potential conflict (see 8.2d). He addressed this conflict by refining the specification of the problem (certain property must be variable) and bringing in a new frame (new familiar case). New frame introduced new conceptual terms, especially at the level of 'structural primitives' for the construction of solutions (set $\mathcal{T}$), e.g. a pneumatic piston.

Reasoning step(s) observed in episode 8.2d:	
	Further development of articulated $\mathcal{T}$ (action #6b)
	Retrieval of a familiar frame (action #2)
	Articulation of **S**, **T**, $\mathcal{T}$ similar to a familiar frame (action #3a and 3b)
	Is re-usable component admissible? (action #4)

EXPLICIT ACHIEVEMENT 8.2/DC-II

Having informally assembled the pieces, the designer decided to make the first 'official release' of a solution to the problem as he framed it. He confirmed his commitment to the informally considered requirements (see R-1 to R-6 in 8.2/DC-II) and sketched a solution addressing them. In other words, he constructed a problem solving model ***T***, which satisfied the earlier-mentioned explicit requirements. Thus, he arrived at the first candidate solution that could be assessed for acceptability.

Reasoning step(s) observed in context 8.2/DC-II:	
	Frame articulation specifies/satisfies (action #1a and 1b)
	Further development of articulated $\langle \mathcal{T}, \mathcal{S} \rangle$ (action #6a and 6b)
	Applying admissible components (action #5)

Design episode 8.2e

As the two subsequent episodes (8.2e and 8.2f) prove, the candidate solution was not entirely acceptable, and the designer modified his current interpretation of the problem. One part of the frame that showed potential for a modification was identified as inflexibility in the control algorithm. In order to fix this issue, the designer articulated a new requirement covering this incompleteness and extending his existing interpretation of the term 'active suspension' that emerged in one of the initial frames.

Reasoning step(s) observed in episode 8.2e:	
	Solution acceptability assessment (action #7)
	Frame amendment (action #8)
	Refinement of problem specification (action #6a)
	Applying admissible components (action #5)

Design episode 8.2f

In a similar manner, another issue was brought forward in design episode 8.2f, which eventually led to a modification of the initial assumption R-2. Drawing on his past knowledge, the designer refined the original statement R-2 to reflect better the changed 'definition' of the term 'active suspension' from the previous episode. Thus, episodes 8.2e and 8.2f focused particularly on the refinement and extension of the original problem specification ($\boldsymbol{S} \rightarrow \boldsymbol{S'} \subseteq \mathcal{S}^*$).

Reasoning step(s) observed in episode 8.2f:	
	Solution acceptability assessment (action #7)
	Frame amendment (action #8)
	Retrieval of a familiar frame (action #2)
	Articulation of **S** similar to the familiar frame (action #3a)

Design episode 8.2g

Before validating these modifications of the original frame, the designer also tackled the issue of having too abstract conceptual primitives to work with (i.e. set $\mathcal{T}$ and consequently $\mathcal{T}^*$). Therefore, in episode 8.2g he refined and elaborated into details the specific implementation of the components known in the previous explicit context (DC-II), such as measurement of piston displacement or the chassis clearance. Conceptually, he focused more on refining the 'structural primitives' rather than extensions or modifications of the problem specification.

Reasoning step(s) observed in episode 8.2g:	
	Solution acceptability assessment (action #7)
	Admissibility/completeness check (action #4)
	Retrieval of a familiar frame (action #2)
	Articulation & refinement of T, $\mathcal{T}$ similar to familiar frame (actions #3b, 1b)

EXPLICIT ACHIEVEMENT 8.2/DC-III

Having found potential remedies to the identified inadequacies, the designer made the second explicit commitment to propose a candidate solution – i.e. to *satisfy* the amended specification. In the transcript, this step is recorded as an explicit achievement 8.2/DC-III and respective solution (sketch) is labelled as S-2. As it happened with solution S-1, the acceptability of the proposed candidate had to be established.

Reasoning step(s) observed in context 8.2/DC-III:	
	Frame articulation specifies/satisfies (action #1a and 1b)
	Further development of articulated $\langle \mathcal{T}, \mathcal{S} \rangle$ (action #6a and 6b)
	Applying admissible components (action #5)

Design episode 8.2h

Although more complete than S-1, the latest proposal still contained unacceptable features. Thus, as the design episode 8.2h shows, the candidate failed the acceptability test. The designer refined the current specification, and decided that the potential reason for failure could be the lack of co-ordination between the two sub-controllers of the overall solution. Thus, a new requirement was added and a corresponding set of structural primitives was identified. In other words, this frame modification focused more on the clarification of existing specification (***S***) and the extension of the current 'conceptual base' ($\mathcal{T} \rightarrow \mathcal{T}$').

Reasoning step(s) observed in episode 8.2h:	
	Solution acceptability assessment (action #7) Frame amendment (action #8) Refinement of problem specification (action #6a)

Design episode 8.2k

The concepts that emerged in the design episode 8.2h were further elaborated in 8.2k pushing the vague notions from 8.2h into more specific terms, such as 'interface' and 'translation'. On the level of problem specification (***S***, $\mathcal{S}$), the purpose of the interface was expressed in terms of typical scenarios that needed to be addressed. In the space of conceptual primitives ($\mathcal{T}$), a fuzzy controller emerged in addition to the original 'quantitative' ones in order to translate the driver's 'qualitative' requests.

Reasoning step(s) observed in episode 8.2k:	
	Retrieval of a familiar frame (action #2) Articulation of T, $\mathcal{T}$ similar to familiar frame (actions #3b) Is re-usable component admissible (action #4) Applying admissible components (action #5)

EXPLICIT ACHIEVEMENT 8.2/DC-IV

The proposed modification of the frame used in context DC-III was verified by an explicit commitment to the new requirement and construction of another solution (S-3). This sequence is recorded as an explicit achievement 8.2/DC-IV. The new problem solving model ($\boldsymbol{T} \subseteq \mathcal{T}^*$) was articulated so that it *satisfied* the modified problem specification (***S***) and was assessed for acceptability.

Reasoning step(s) observed in context 8.2/DC-IV:	
	Frame articulation specifies/satisfies (action #1a and 1b) Further development of articulated $\langle\mathcal{T}, \mathcal{S}\rangle$ (action #6a and 6b) Applying admissible components (action #5)

EXPLICIT ACHIEVEMENT 8.2/DC-V

The acceptability test did not pass 'unconditionally', because the designer made a small amendment to the frame from context DC-IV before declaring a design solution. However, as record 8.2/DC-V shows, this was not so much the failure of the acceptability test as the completeness and clarity test. Nonetheless, it triggered minor refinements of the structural primitives in the problem-solving model ***T***, and led to a refined conceptual frame in the new explicit context DC-V.

Reasoning step(s) observed in context 8.2/DC-V:	
	Solution acceptability/admissibility checks (actions #4 and #7) Further development of articulated $\langle\mathcal{T}, \mathcal{S}\rangle$ (action #6b) Applying admissible components (action #5)

The following breakdown in Table 8–2, gives a more concise account of our interpretation of the recorded and annotated design sequence. For clarity, the interpretation is conducted using the vocabulary from chapter 6. The legend associates the referential numbers of reasoning steps with the verbal descriptions. The columns correspond to the individual design episodes of the narrative as described earlier in section 8.2. The values in the individual cells reflect the 'order' in which the different actions were observed (if possible).

Table 8–2. Conceptual summary of task T11 (see design episodes 8.2a to 8.2k)

'action'	a	b	c	d	DC-2	e	f	g	DC-3	h	k	DC-4	DC-5
1a		1.			1.				1.			1.	
1b					1.			4.	1.			1.	
2	1.		1.	2.			3.	2.			1.		
3a	2.			3.			4.						
3b		1.	2.	3.				3.			2.		
4			4.	4.				1.			3.		1.
5			3.		3.	4.			3.		3.	3.	3.
6a		2.			2.	3.			2.	3.		2.	
6b		2.	2.	1.	2.				2.			2.	2.
7						1.	1.	1.		1.			1.
8						2.	2.			2.			

Legend to Table 8–2:

Reference	Verbal description of referenced action
1a	Articulate frame → *specifies(S,* $\mathfrak{DP}$*)*
1b	Articulate frame (conceptual part $\mathfrak{T}$)
2	Retrieve similar frame ($\phi = \langle\tau, \theta\rangle$)
3a	Articulate *S*, $\mathcal{S}$ similar to θ
3b	Articulate *T*, $\mathfrak{T}$ similar to τ
4	Is re-used τ and/or θ *admissible*?
5	Exists such *T* that *satisfies* given *S*
6a	Develop *S*, $\mathcal{S}$ into details (incl. alternatives)
6b	Develop *T*, $\mathfrak{T}$ into details (incl. alternatives)
7	Is candidate solution *T acceptable*?
8	Re-interpret current frame (i.e. *S* → *S*')

8.4 Annotated task T21 – paper-smoothing plant

The initial problem specification or design brief given to the designer for this particular problem reads as follows. *Paper-milling plant consists of several different chains of machinery that produce different types of paper. We are interested in one particular plant that takes semi-finished paper on a roll as an input, smoothes it, and rewinds it on another roll on output.*

Your task is to design a part of a paper mill that would perform the above-defined operation, and ensure that the paper is evenly thick, smooth, and wound tightly on the output. The raw paper at the input may have ripples, variable thickness, rough surface, or any combination of these features. In addition to the design of a plant, suggest also a robust strategy for the regulation of such a plant, particularly with regard to paper thickness and strength changing case to case.

Design episode 8.4a

Initialisation of the first frame in context DC-I

Design context: DC-I Justifying threads: J-690 to J695

As in the other experimental cases, the designer used the design brief as shown above, and conducted an initial task analysis that aimed to clarify the objectives and specify the problem in more formal language. The introduction to the problem was made in record J-692, in which the designer sketched how an input and output to/from the designed system might look like. Based on the initial drawing, he started articulating the parameters describing the problem in the language of design, and expressed their desired functionality of the designed system {J-693 to J-695}.

While the reference to the past experience is observable in threads J-694 and J-695, it is more subtle than in some other cases. The designer used a familiar vocabulary of concepts such as 'reducing paper thickness', 'polishing paper surface' to articulate the initial frame for the interpretation of the problem. The past experience is observed in J-695 when he attended to the issue of how much the paper thickness would be modified by the system. This initial frame led to the formal expression of the task objectives explicitly articulated later in context DC-II.

Design episode 8.4b
Development of the initial frame and the emergence of explicit objectives
Design context: DC-I Justifying threads: J-696 to J704

In addition to the clarification of the concepts from the design brief (see 8.4a), the designer brought forward several additional parameters and requirements. As can be seen from the records, these originated in his generic knowledge of the control theory and experience with controlling mechanical systems {J-696}. For example, he demanded additional 'restrictions' from the customer, and enquired about the typical dimensions and weight of the processed paper roll {J-697, J-699}.

Another issue that was not mentioned in the initial brief but was attended to during the refinement of the initial frame was the expected initialisation of the process {J-700}. He pointed out that it was highly probable that the system might require manual start-up, for instance in the form of feeding the paper through the assembly to the output roll {J-703}. This proposal did not appear to be very important, however. More important consequence was that a similar situation might occur at the end of the process. Thus, he articulated the need to monitor when the paper at the input roll is 'running out', so that the system could halt in time before tearing the paper from the input roll {J-704}.

EXPLICIT ACHIEVEMENT 8.4/DC-II
First 'prototypic solution' is formally recorded

Despite rather abstract definition of the desired parameters of the 'ideal solution' that appeared in the initial phase (8.4a and 8.4b), the designer made an early explicit commitment to the following goals and requirements he aimed to achieve:

– the system should be able to remove ripples from the raw paper on input {R-1}
– the system should also be able to polish the surface so that it does not contain any 'rough' debris {R-2}
– the system should be able to reduce thickness to the desired value as specified by the operator, so that paper is evenly thick throughout its length {R-3}
– and finally, the controller should be able to detect the paper on the input roll running out and stop the process before the paper is abruptly torn off the input roll {R-4}

Since the initial frame was articulated rather abstractly, it did not contain the necessary conceptual 'primitives' to construct a technologically feasible solution. Therefore, the designer used a 'black box' approach to express the solution functionally. Such approach enabled him to propose and work with an early candidate solution without committing himself to a particular technology. The system in this highly abstract 'solution' S-1 is referred to as 'processing' or 'smoothing' unit that exhibits the desired functionality. The designed system also contains an input and output rolls (see Figure 8–7).

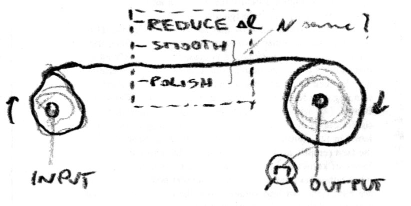


Figure 8–7. Initial abstract formulation of the design problem

Design episode 8.4c
Extension of the initial frame and details of the 'smoothing unit'

Design context: DC-II Justifying threads: J-707 to J711

As expected (and intended), the solution proposed within frame DC-II needed further refinement before any acceptability could be tested. Therefore, the designer investigated the means how the desired objectives could be achieved {J-708}. One technology that appeared in the justifying records was based on rolling and the application of the pressure by a rolling drum on the paper surface {J-709}. The designer sketched several different prototypic implementations of this technology (e.g. a single drum on a 'table' or a pair of drums with a gap in between).

Rolling was a technology that was applicable and could be re-used in this particular problem. However, the designer did not accept it 'off the shelf'. Record J-709 shows a quick acceptability check and the identification of potential issues with the application of rolling. The main concern the designer had was that the paper might be damaged when rolled and the operation might not be efficient. From the experience he recalled that rolling ('ironing') was often accompanied by some form of pre-processing the material – e.g. warming up, wetting, or melting.

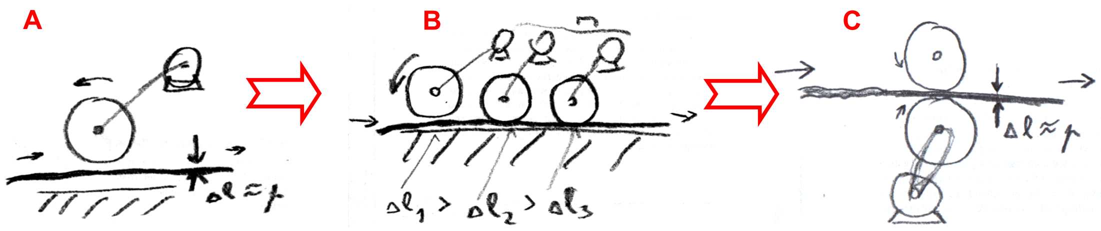


Figure 8–8. Functional alternatives addressing the explicit requirements

Therefore, in records J-709 and subsequently J-711 he introduced an additional requirement to the problem specification assuming that the paper would be treated before entering the rolling assembly. This requirement was to ensure efficiency and reduce the danger of paper damage that might have occurred if the pressure on a dry paper was increased. Simultaneously with the new assumptions, the concepts for constructing the detailed solutions emerged, and the designer proposed how such a moisturiser might look like. See this concept in Figure 8–9 that shows the system at a later stage.

<u>Design episode 8.4d</u>
Further refinements of the frame and ordering of the operations
Design context: DC-II Justifying threads: J-717 to J720

Following the proposals made in the design episode 8.4c, the designer elaborated the details of the newly articulated concepts of 'wetting' and 'rolling' {J-717}. One important concept emerging from his commitment to this processing was an order in which the operations take place and consequently a layout of the designed system. Since there was a clear causal dependency between wetting and rolling, the order was 'given' by this causality – first moisturise, then apply pressure between the rolling drums {J-718}. In order to keep the conceptual solution simple the designer chose a single pair of drums with a gap corresponding to the desired thickness {J-719}.

Nevertheless, this breakdown and explicit attendance to the details of the interactions between the operations introduced the need of another 'operation'. After leaving the rolling drums, the paper was still damp, and as such, it could not be stored on the output roll. Therefore, the designer rectified this inconsistency by bringing in a unit that would dry the damp paper. In other words, yet another requirement emerged, and the corresponding structural concepts were identified {J-720}.

<u>EXPLICIT ACHIEVEMENT 8.4/DC-III</u>
Detailed and technologically improved solution formally recorded

As already discussed in the episodes 8.4c and 8.4d, several new requirements emerged during the refinement of the over-simplified prototypic solution S-1. In addition to those explicitly noted in context DC-II, the new extended context DC-III contained the designer's explicit commitment to the following three objectives addressing the newly identified need of pre-processing:

– the paper should be treated before any manipulation by moisturising the dry paper delivered to the input {R-5}
– once the paper is wetted, it should pass the smoothing unit where ripples should be removed and thickness reduced {R-6, also addressed partially by R-1 and R-3}
– the process should ensure that dry paper is delivered at the output; therefore a mechanism removing excessive moisture should be added after the 'smoothing unit' {R-7}

The newly-articulated objectives were embodied in a much more realistic, though still rather simple solution. The designer addressed the need of a moisturiser by implementing 'shower-like' jets releasing warm damp on the paper. Rolling was confirmed as a basic principle for smoothing, and finally, drying was addressed by implementing a 'tunnel' with hot, dry air that caused the excessive moisture evaporate. Warm paper leaving the drying tunnel was immediately wound on the output so that the paper does not 'wrinkle' as an effect of drying. This solution was labelled as S-2 and the next step in its development is sketched in Figure 8–9 (see also context 8.4/DC-IV).

Design episode 8.4e
Exploratory probe addressing the issue of thickness reduction
Design context: DC-III Justifying threads: J-723 to J727

Using a similar strategy as described in the explicit achievement 8.4/DC-II and proposing a partial solution to check the feasibility of the conceptual approach, the designer addressed another outstanding issue with thickness reduction {J-723}. He identified the reason of his dissatisfaction with the simple solution S-2 as an insufficient mechanism for thickness reduction. Although the rolling unit could be re-used for thickness reduction as well, the single pair of drums imposed undesired limitations on the degree of thickness reduction; i.e. how much can be reduced by a single pair {J-725}.

The designer demanded a more robust and flexible approach that would be able to reduce thickness to a greater extent (if necessary). In order to address this refined requirement of thickness reduction, he articulated a new conceptual term featuring a sequence of rolling drums {J-726}. Another important concept emerging from assessing the acceptability of solution S-2 featured decreasing values of the gap clearance between the rolling drums {J-727}. Starting with the first pair (closest to the moisturiser) the gap clearance would decrease in steps, so that the last pair of drums features the desired thickness. This approach would reduce the thickness step-by-step thus avoiding potential dangers of clogging the rolling assembly with excessive paper 'left-overs'.

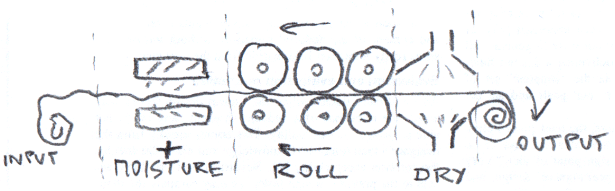


Figure 8–9. Simple solution extended and refined

EXPLICIT ACHIEVEMENT 8.4/DC-IV
Third, re-addressed solution formally recorded

After introducing new conceptual terms to address the inadequate behaviour of previous solutions (S-2) in episode 8.4e, the designer recorded the explicit commitment to the requirement of merging the 'smoothing' mechanism with a 'thickness reducing' mechanism. He implemented it within existing rolling unit. Thus, the refined specification of the problem contained the following statement:

– the desired purpose of the smoothing unit should be co-ordinated with the need of thickness reduction; for instance by multiplying the number of 'smoothing units' each having a different gap clearance {R-6.1 extending the original R-6 from DC-III}

The new frame of context DC-IV did not differ from the previous one in many aspects. The most striking difference was the introduction of a sequence of rolling drums and their linear arrangement with a decreasing gap clearance between the drums in each pair. The sequence replaced the single pair from solution S-2. Although very similar to S-2, the new iteration was conceptually distinct, and therefore was labelled as S-3. It is also depicted as a sketch in Figure 8–9.

<u>Design episode 8.4f</u>
Re-addressing the issue of thickness reduction and paper damage
Design context: DC-IV Justifying threads: J-728 to J738

With S-3 being a realistic candidate solution to the specified problem, the designer assessed its acceptability, and became aware of several drawbacks of this particular approach. As record J-731 suggested, they could have been identified after sketching the intended concept of 'sequence' in the designer's notebook. His main concern is clearly visible when comparing Figure 8–8c with the latest sketch in Figure 8–9. In J-731 the designer suggested that such a layout might be large and impractical from the controlling perspective. Moreover, the damp paper could still be torn in the rollers when pulled through, which was clearly unacceptable and contradicted the explicit requirements.

As record J-733 shows, he considered a simple remedy to the large size by 'squeezing' the drums; i.e. placing them closer to each other, thus reducing the overall length of the assembly. However, he was not satisfied with this 'fix' and explored the other opportunities. The concept of 'squeezing' was apparently useful for triggering an analogy, because the next record {J-734} shows the emergence of an entirely new conceptual term. First, the new concept emerged in the language of existing frame, and required 'squeezing the drums in two dimensions instead of one'. This cumbersome definition was followed up and re-formulated in record J-736.

Thus, a new concept of 'alternate layout' was introduced in J-736 as an alternative to the current 'linear layout'. This idea was elaborated in J-738 and proved to be promising, because it featured much larger 'effective or active surface' acting on the paper (another new concept emerging from the frame extension). Thanks to a more efficient rolling, the pressure of the drums could have been reduced and the gap clearance ceased to be a major restrictive factor.

<u>EXPLICIT ACHIEVEMENT 8.4/DC-V</u>
New solution in a re-framed context formally recorded

The newly-emerged or identified alternative for re-arranging the drums was formally recorded as a preferred requirement that replaced the old R-6.1 from context DC-IV. Its transcript is as follows:

– the functionality of smoothing should be co-ordinated with the need of thickness reduction; for instance by re-arranging the 'smoothing units' in an alternate (zigzag) manner {R-6.2}

The plant featuring the alternate layout of the rolling pairs and their control was recorded as an extended solution S-4, and it is depicted in Figure 8–10. As it was observed in achievement 8.4/DC-III, the two alternatives are functionally similar but at the same time, conceptually they are very different. They are based on totally different conceptual primitives.

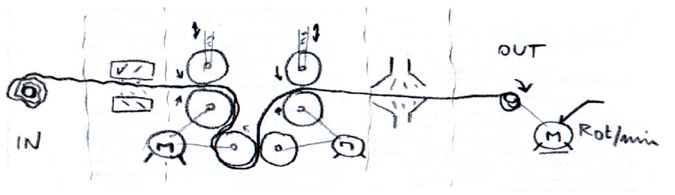


Figure 8–10. Introduction of a concept of alternately positioned rolling drums

Design episode 8.4g
Currently controlled devices (motors, rolls) incompatible with the new layout
Design context: DC-V   Justifying threads: J-754 to J762

The next step appearing in the recorded notes referred to transferring the design focus to the actual control strategy {J-754}. So far, the control was very simple; basically, adjusting all the parameters at the beginning and monitoring only the end of paper at the input roll (see also episode 8.4b). However, such strategy did not suit the latest layout of the assembly, and needed to be re-addressed. The first point that was clarified re-iterated record J-695 in terms of thickness being constant for a given input roll; i.e. it would not change during the actual smoothing operation {J-755}.

This new assumption helped to articulate the control strategy based on the monitoring the input to the smoothing unit {J-756}. The parameters measured at the input would apply throughout the process; including approximate diameter and weight of the roll. Similarly, the amount of water or damp sprayed on the paper in the moisturiser would be determined by the initial humidity of paper.

The output roll was proposed as the part of the system upon which the control action would be exerted {J-759}. It was coupled with a motor that rotated the roll with an appropriate velocity. The velocity corresponded to the actual diameter of the output roll (which was also measured), and was controlled so as to ensure certain tension of the paper throughout the smoothing assembly.

The designer decided to have a simple control strategy and drive only the output roll {J-762}. This drive was simultaneously responsible for unwinding the paper from the input roll and pulling it through the moisturising unit, rolling unit and drying unit. Another functionality required from the control was identified as maintaining the tension of paper thus avoiding the undesired 'wrinkling'.

EXPLICIT ACHIEVEMENT 8.4/DC-VI
Details of the control strategy elaborated and formally recorded

In order to address the issues emerged in design episode 8.4g that proposed further development of the previous solution S-4, the designer articulated two additional requirements. Unlike the requirements recorded in the previous frames, these were more concerned with the details of the control strategy rather than the actual operation of smoothing. Nonetheless, the new context DC-VI was articulated and contained these two requirements in addition to DC-V (and the previous contexts):

– first, a mechanism was needed for unwinding the paper from the input roll and simultaneously winding it at the output {R-10}
– second, an adjustable drive (driving force) was identified in order to rotate the output roll thus maintaining the paper tension {R-11}

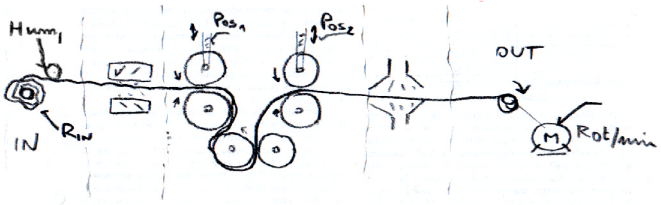


Figure 8–11. Smoothing plant with alternate rollers and control parameters

A solution articulated to address the newly formulated needs was labelled as S-5 and in principle, it was structurally based on the previous one (S-4). However, as Figure 8–11 shows, it contained several additional concepts. These included the sensors measuring roll diameter at the input (R_{IN}), humidity sensor (Hum_I), gap adjusters (Pos_1, Pos_2), controlled parameter at the output (Rot/min), and an appropriate actuator at the output (motor). These new conceptual objects were positioned to at the appropriate places of the designed plant and the paper smoothing process.

<u>Design episode 8.4h</u>
Conflict between the control strategy and damp paper
Design context: DC-VI Justifying threads: J-765 to J768

However, after sketching the proposed control strategy based on driving the output roll, the designer discovered a contradictory pair of requirements that resulted in an increased danger of damaging paper {J-766}. On one hand, the output roll torque was maximal at the beginning of the process when most of the paper was at the input. On the other hand, paper was wetted and fed into the complex rolling assembly. Driving the output roll meant transferring potentially large torque exerted at the output through the entire system, so that the input roll started unwinding. For thinner and damper paper, this could prove fatal and the paper could tear, which was unacceptable

Therefore, the designer amended the requirement of having a single adjustable drive, and proposed controlling both input and output rolls based on the measurements between them {J-767}. Thus, he modified the original concept of 'pulling' from frame DC-VI to a combined 'pulling' coming from the output roll and 'pushing/feeding' coming from the driven input roll. The proposal seemed to be applicable because it balanced potentially dangerous torque and spread it 'evenly' between both ends of the designed plant.

<u>Design episode 8.4k</u>
Refinement of the new conceptual interpretation of input/output control
Design context: DC-VI Justifying threads: J-768 to J770

Before implementing the deliberation from episode 8.4h explicitly, the designer seemed to assess the acceptability of the latest idea. As record J-768 shows, the 'pushed' paper could cause unneeded problems similar to those identified earlier in design episode 8.4e. The rolling drums could be easily clogged if the input roll pushed the paper between them. The danger was only emphasised by the fact that by now the system was feeding wet paper, which was more difficult than feeding the dry paper (e.g. wet did not keep the shape).

Therefore, he returned to the concept of 'pulling' paper from the input roll. However, this time, a 're-visited' form was proposed {J-769}. It assumed driving an additional device that would be located between the input roll and the moisturising jets. It was suggested that it consisted of two 'drums' that are driven so that the tension of the paper is maintained on a constant level. Thus, a similar pattern was re-applied as at the output, just in the reversed order and in a different location. This remedy solved the problem of clogging the rollers, because only that much paper was unwound as to maintain the tension.

<u>Design episode 8.4l</u>
Radical re-shape of the rolling assembly
Design context: DC-VI, DC-VII Justifying threads: J-768 to J770

Following the introduction of a new conceptual object - a device pulling the paper from the input and ensuring the sufficient tension was maintained, the designer proposed an interesting re-shape of the plant. This was not recorded 'verbally' but it emerged in one of the last drawings. The idea from an initial frame in the context 8.4/DC-II of using pairs of rolling drums was re-visited. Instead of pairs, the single drums arranged in a zigzag fashion 'did the trick'. As the designer mentioned on a margin, this could be sufficient and simpler than controlling the complex plant shown in Figure 8–11.

From his description of the final structure of the plant, it is possible to figure out the return to the concept of 'polishing' that already appeared in episode 8.4a and context 8.4/DC-II! The same drums that were originally applying pressure on the paper were also generating friction. Using this interesting parallel, the principle of rolling was re-interpreted from the pure 'pressure application' to a combined 'pressure and friction' (featuring a kind of 'abrasion'). The latest iteration in the solution development featuring the paper pulling from the input roll and the new shape of rollers is shown in Figure 8–12.

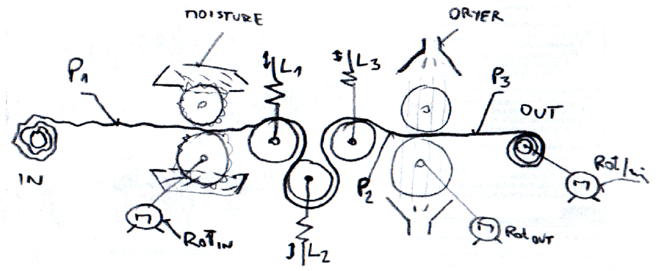


Figure 8–12. Re-shaped solution addressing the re-framed problem

<u>EXPLICIT ACHIEVEMENT 8.4/DC-VII (DC-VIII)</u>
'Final' solution and refined control strategy

As captured in the sequence of design episodes 8.4h to 8.4l, the specification of paper smoothing problem was extended so that there were two drives. This new requirement extended and refined the original R-10 appearing in the frame DC-VI. In addition to the refinement of R-10 and as its direct consequence, also a new requirement similar to R-11 from frame DC-VI was added. Here is the list of the extensions to the original frame (DC-VI) forming a new frame DC-VII:

– first, a mechanism was needed for unwinding the paper from the input roll and another (physically independent but co-ordinated) one was needed for winding paper at the output {R-10.1}
– second, an adjustable drive (driving force) was needed to unwind the paper from the input roll that would simultaneously maintain the tension at the input of the plant and prevent clogging {R-12}

As we already mentioned in episode 8.4l, the designer eventually re-shaped the layout of the plant to reflect the latest additions to the problem specification. The 're-shape' could take place thanks to the implementation of a device addressing requirement R-10.1 as well as the designer's re-interpretation of an existing concept ('polishing'). The result of the design process that was eventually proclaimed a design solution is depicted in Figure 8–12. Rough schema of the control loop with the measured and controlled parameters is depicted in Figure 8–13. Solution S-6 consists of both parts {J-770}, only it was easier to scan the structure and the control circuit separately.

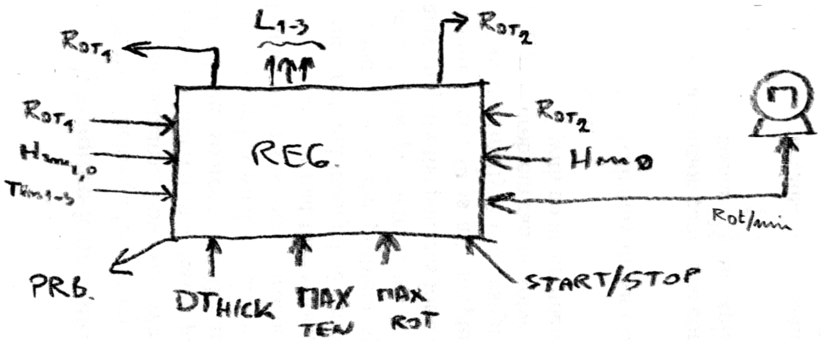


Figure 8–13. Control loop with measured/control parameters complementing Figure 8–12

DESIGN TASK ENDS BY PROPOSING A SOLUTION

8.5 RFD analysis of task T21

As we did in section 8.3, we summarise the annotated transcript of the experimental session T21 in terms of the initial RFD model as introduced in chapter 6. The annotation of the captured records revealed that task T21 was tackled in seven successive conceptual contexts. In the episodic narratives in section 8.4, these are labelled from DC-I to DC-VII. Moreover, there were ten 'self-contained' design episodes identified in section 8.4 from 8.4a to 8.4l. Thus, the first generic observation on the level of the overall appearance of the annotations reveals almost the same number of the self-contained reasoning chains as we observed in task T11 (in sections 8.2 and 8.3). However, the pattern of the overall design process looks slightly different.

First, there are more explicit articulations of the partial solutions than it was in task T11. This observation is also supported by the total number of 'explicit milestones' (8.4/DC-II to 8.4/DC-VII) – six as compared to four. Second, the distribution of the narratives and explicit achievements is also different. These two differences suggest a more structured approach in the second analysed task. They also suggest a more exploratory approach to the design of a paper-smoothing plant. In our opinion, the designer explored (and constructed) the available design space by articulating a small set of 'probes', which he immediately attempted to apply to validate their feasibility.

The pattern of structuring the design problem is particularly well observable in the regular interweaving of the explicit records and solution articulations (8.4/DC-III to DC-VI) and the 'probing extensions' to the problem interpretation (design episodes 8.4e to 8.4g). Several of these exploratory probes feature a frame amendment, which supports our assumptions from chapter 6 about a frequent mutual interaction between the problem specifications and solutions. Similarly as it has been observed in section 8.3, the shorter sequences of design episodes seem to be related to the immediate validation of the amended frame.

Design episode 8.4a

Task T21 started similarly as task T11 with the population of the initial context DC-I from the design brief, and the explicit specification of the problem $\mathfrak{DP}$. The process of framing featured a retrieval and application of the familiar past cases and the re-use of their vocabulary for the specification of the main objectives (see episode 8.4a). Only little commitment has been made in respect to the articulation of 'structural primitives', in the first narrative, the designer focused on the basic tenets of the explicit problem specification. Rather than labelling these concepts as set ***S***, they exhibited more generic and abstract features of the conceptual specification $\mathcal{S}$ (see chapter 6 for differences).

Reasoning step(s) observed in episode 8.4a:	
	Articulation of terms for problem specification (action #1a)
	Retrieval of a familiar frame (action #2)
	Articulation of S, $\mathcal{S}$ similar to a familiar frame (action #3a)

Design episode 8.4b

The identification and selection of the main requirements from set $\mathcal{S}$ defined in 8.4a (i.e. the articulation of an explicit problem specification $\boldsymbol{S} \subseteq \mathcal{S}^*$), paved the way for the identification of basic concepts ($\mathcal{T}$), as documented by episode 8.4b. These conceptual primitives included such terms as starting the smoothing process, monitoring paper running out, or measuring weight/diameter. Whether a particular familiar term was included in the initial frame of the design problem depended on its 'admissibility'. The designer performed a simple check on consistency of the newly discovered objects with existing goals of the explicit problem specification ***S***.

Reasoning step(s) observed in episode 8.4b:	
	Retrieval of a familiar frame (action #2)
	Articulation of **T**, $\mathcal{T}$ similar to a familiar frame (action #3b)
	Further development of frame $\langle \mathcal{T}, \mathcal{S} \rangle$ (actions #6a and 6b)
	Is re-usable component admissible? (action #4)

EXPLICIT ACHIEVEMENT 8.4/DC-II

At this stage, the designer presented the first abstract and principled solution, in which the basic conceptual terms were deployed. See for example, the reference to an input and output parts of the designed system or a smoothing unit in the explicit 'milestone' 8.4/DC-II. These concepts were later refined and instantiated using a more specific terminology. While the proposed solution $\boldsymbol{T} \subseteq \mathcal{T}^*$ was too abstract for the purposes of implementation, it provided a useful referential framework as proved in the subsequent iterations. In the language of the RFD model, the designer proposed a problem solving model that satisfied the explicit problem specification – i.e. *satisfies(T, S)*.

Reasoning step(s) observed in context 8.4/DC-II:	
	Frame articulation specifies/satisfies (action #1a and 1b)
	Further development of articulated $\langle \mathcal{T}, \mathcal{S} \rangle$ (action #6a and 6b)
	Applying admissible components (action #5)

Design episode 8.4c

Contrary to our theory of design frame development (chapter 6), the designer's records in episode 8.4c do not show a clear assessment of the proposed solution model for its acceptability. Instead, he continued with the development of the frame started in episode 8.4b above. This interesting pattern, however, seems to support our explanation presented at the beginning of this section about the exploratory probing. In other words, the acceptability test was incorporated in upholding of the explicit specification (***S***$\subseteq\mathcal{S}^*$) and the problem solving theory ($\mathcal{T}^*$) in the construction of the prototypic solution S-1. Thus, the designer refined the conceptual primitives for the construction of a more specific solution (e.g. he added the rolling drums and technology of rolling). Moreover, he also introduced new concepts from his experience that seemed to be applicable to the current problem $\mathfrak{DP}$ – e.g. the requirements to operate efficiently and avoid damage to the paper.

Reasoning step(s) observed in episode 8.4c:	
	Solution admissibility and acceptability checks (action #4 and #7)
	Further development of conceptual base of frame $\langle\mathcal{T}, \mathcal{S}\rangle$ (action #6b)
	Retrieval of a familiar frame (action #2)
	Articulation of **S**, $\mathcal{S}$ similar to a familiar frame (action #3a)

<u>Design episode 8.4d</u>

The vaguely identified extension of the current conceptual frame and problem specification started in episode 8.4c continued also in 8.4d with further refinements and commitments. Thus, the designer extended the current conceptual base $\mathcal{T}$ with such terms as 'wetting' and 'drying'. Another contribution made in this episode was the introduction of ordering between the identified 'primitive' operations featuring paper smoothing (i.e. unwinding → wetting → rolling → drying → winding). The order of the operations was found important, and the designer incorporated an explicit requirement for such an ordering into his explicit problem specification ***S***.

Reasoning step(s) observed in episode 8.4d:	
	Articulation of **T**, $\mathcal{T}$ similar to a familiar frame (action #3b)
	Further development of frame $\langle\mathcal{T}, \mathcal{S}\rangle$ (actions #6a and 6b)

<u>EXPLICIT ACHIEVEMENT 8.4/DC-III</u>

Since several new requirements emerged or were refined since the previous solution was constructed, the designer made another commitment to a partial solution satisfying the amended problem specification. In terms of our model, the explicit record 8.4/DC-III corresponds to the construction of a new set ***T***$\subseteq\mathcal{T}^*$ for which the predicate '*satisfies(T, S)*' would hold. Such a problem solving model was indeed found, and it was noted within the 'milestone' 8.4/DC-III in the iterative design of a paper-smoothing plant.

Reasoning step(s) observed in context 8.4/DC-III:	
	Frame articulation specifies/satisfies (action #1a and 1b)
	Further development of articulated $\langle\mathcal{T}, \mathcal{S}\rangle$ (action #6a and 6b)
	Applying admissible components (action #5)

<u>Design episode 8.4e</u>

Unlike the previous solution S-1 developed in the context 8.4/DC-II, solution S-2 was already more specific and it could be assessed for its acceptability. As documented in the episode 8.4e, the designer did not accept this particular implementation, and re-visited the requirement demanding thickness reduction. In order to articulate the 'missing bits', he brought in several new requirements about the flexibility and robustness of the thickness reduction, as well as the requirement of a co-ordination between paper smoothing and thickness reduction. In the language of our theory, design episode 8.4e corresponds to the refinement of the problem specification ($S' = S \cup \{s\}$) and definitely features a frame development.

Reasoning step(s) observed in episode 8.4e:	
	Solution acceptability assessment (action #7)
	Frame amendment (action #8)
	Refinement of S, $\mathcal{S}$ (action #6a)
	Proposing a solution (action #5)

Explicit achievement 8.4/DC-IV

The extensions proposed in design episode 8.4e were immediately tested by a proposal of another candidate solution (S-3) as documented in record 8.4/DC-IV. This context focused on the co-ordination between the two operations as demanded in the modified problem specification in 8.4e. The designer looked for yet another problem solving model ($\boldsymbol{T'} \subseteq \mathcal{T}^*$) that would *satisfy* such specification. Such model was indeed found, and it contained a few new conceptual terms that were hinted in the design episode 8.4e (e.g. a sequence of the rolling drums, decreasing gap clearance, stepwise reduction of thickness, etc.). The solution with a linear sequence of drums seemed to address the explicit problem specification, but needed to be assessed for its acceptability.

Reasoning step(s) observed in context 8.4/DC-IV:	
	Frame articulation specifies/satisfies (action #1a and 1b)
	Further development of articulated $\langle \mathcal{T}, \mathcal{S} \rangle$ (action #6a and 6b)
	Applying admissible components (action #5)

Design episode 8.4f

The acceptability assessment occurred in the design episode 8.4f, where the designer reflected on the sketch that was made as a part of solution S-3. Although as a first approximation the solution was sound, the designer became aware of several potential dangers of the proposed layout. One problem was identified as impractical length of the assembly that might pose difficulties with control and maintenance. Another issue that contradicted his explicit problem specification was concerned about tearing the wet paper when too big pressure was applied on it. Thus, the co-ordination of smoothing and thickness reduction had to be re-addressed, and the unacceptability of paper tearing was re-iterated. In addition to an extended problem specification, a new concept of 'two-dimensional' (zigzag) arrangement of drums emerged leading to an extension of conceptual base $\mathcal{T}$.

Reasoning step(s) observed in episode 8.4f:	
	Solution acceptability assessment (action #7)
	Frame amendment (action #8)
	Retrieval of a familiar frame (action #2)
	Articulation of **S**, **T**, $\mathcal{T}$ similar to a familiar frame (actions #3a and 3b)
	Refinement of **S**, **T**, $\mathcal{S}$ (actions #6a and 6b)
	Proposing a solution (action #5)

Explicit achievement 8.4/DC-V

The proposed re-formulation of one requirement and the respective solution was immediately tested by sketching it in the notebook (see 8.4/DC-V). Though functionally the proposed solution S-4 behaved similarly is S-3 or S-2, it was conceptually very different. Mainly because it first introduced the zigzag layout of the drums that 'survived' until the end of the design process. Another contribution of this explicit commitment is a clear presence of an iterative development of a requirement – for example, the requirement R-6 has been re-formulated and refined twice since its conception.

Reasoning step(s) observed in context 8.4/DC-V:	
	Frame articulation specifies/satisfies (action #1a and 1b)
	Further development of articulated $\langle \mathcal{T}, \mathcal{S} \rangle$ (action #6a and 6b)
	Applying admissible components (action #5)

Design episode 8.4g

Structurally, the proposal S-4 was found acceptable; however, the simplistic control strategy used from the beginning was not fully suitable for a more complex structure. Therefore, in 8.4g the designer re-focused his objectives to amend this strategy. He decided to control the output roll, and the influential parameters were measured mostly at the input of the plant. In order to express his preference about the control strategy, he articulated a new requirement that the output roll be responsible for winding the paper at the output, unwinding it from the input and pulling it through the plant. This new statement extended existing problem specification (***S***) and was complemented by an articulation of new conceptual objects for its implementation, e.g. sensors and motors (extending set $\mathcal{T} \rightarrow \mathcal{T}$').

Reasoning step(s) observed in episode 8.4g:	
	Solution acceptability/admissibility assessment (action #7 and #4) Refinement of existing **S**, **T**, $\mathcal{S}$ (actions #6a and 6b) Proposing a solution (action #5)

EXPLICIT ACHIEVEMENT 8.4/DC-VI

As in the previous iterative episodes, the new requirements were implemented and the designer recorded another explicit commitment to the proposed structure and control strategy, as shown in the explicit achievement 8.4/DC-VI and respective solution S-5. The focus shift was manifested by an introduction of brand new requirements to the problem specification (***S'*** = *S* ∪ *{s}*). These were formulated to address the outstanding issues with the previous solution S-4, which needed to be amended as well. The new solution model perused the newly introduced conceptual primitives (see set $\mathcal{T}$' in design episode 8.4g), and the essential contribution of this episode was their incorporation into existing structure of the previous solution S-4.

Reasoning step(s) observed in context 8.4/DC-VI:	
	Frame articulation specifies/satisfies (action #1a and 1b) Further development of articulated $\langle\mathcal{T}, \mathcal{S}\rangle$ (action #6a and 6b) Applying admissible components (action #5)

Design episode 8.4h

The exploratory probe conducted in 8.4/DC-VI revealed an issue with exerting too large torque and tension on the paper when pulling it through the entire assembly as proposed in S-5. Design episode 8.4h returned to the control strategy and modified the preference for a single controlled point (the output roll) thus re-formulating the explicit problem specification. As a quick 'fix' to the issue, the designer proposed driving and controlling both input and output rolls, thus pushing and pulling the paper. The aim of this shift was to distribute the torque and tension more evenly.

Reasoning step(s) observed in episode 8.4h:	
	Solution acceptability assessment (action #7) Frame amendment (action #8) Refinement of existing **S**, **T**, $\mathcal{S}$ (actions #6a and 6b) Proposing a solution (action #5)

Design episode 8.4k

As the subsequent design episode 8.4k documents, the tentative proposal of pushing (feeding) the paper into the rolling assembly did not pass the acceptability check. The designer resolved that it was not a feasible alternative, because in certain instances, it could make the issue of damaging the paper worse. Therefore, he proposed a new conceptual primitive to address the conflict within paper feeding.

In the language of the theory from chapter 6, he amended both sets forming a design frame – $\mathcal{S}$ and $\mathcal{T}$. However, it is difficult to say in what order did this happen.

Reasoning step(s) observed in episode 8.4k:	
	Solution acceptability assessment (action #7) Refinement of existing **S**, **T**, $\mathcal{S}$ (actions #6a and 6b) Proposing a solution (action #5)

Design episode 8.4l

Before testing these probes in an explicit solution, the designer proposed an interesting alternative to existing structure of the designed plant. Design episode 8.4l does not contain any explicit reference to an unacceptable behaviour; however, it features a significant frame shift. We would like to argue that the acceptability test was conducted internally, and the designer believed he could further improve his design although not being able to put his finger on any explicit conflict. What happened in this last occurrence of re-framing re-interpreted the functionality of existing structural primitives (drums). We think that this reasoning sequence shows very nicely the principle of the conceptual re-formulation rather than extension or refinement that were observed in the previous episodes. The conceptual base $\mathcal{T}$ remained the same as in the context DC-VI; however the 'meaning' (purpose) of the individual elements changed (see the shift from 'pressure application to 'abrasion' in 8.4l).

Reasoning step(s) observed in episode 8.4l:	
	Frame amendment (action #8) Articulation of **S**, $\mathcal{S}$, $\mathcal{T}$ similar to a familiar frame (actions #3a and 3b) Proposing a solution (action #5)

EXPLICIT ACHIEVEMENT 8.4/DC-VII

Eventually, the probes proposed in design episodes 8.4h to 8.4l were incorporated into an explicit account of a solution that was finally found reasonably complete and acceptable to address the given design problem $\mathfrak{DP}$. The set of requirements recorded in context 8.4/DC-VII sufficiently *specified* the problem $\mathfrak{DP}$, and the last proposed alternative of a solution (S-6) *satisfied* this 'sufficient' problem specification. In addition to the satisfiability, the solution S-6 was also 'tacitly' *acceptable*.

Reasoning step(s) observed in context 8.4/DC-VII:	
	Frame articulation specifies/satisfies (action #1a and 1b) Further development of articulated $\langle\mathcal{T}, \mathcal{S}\rangle$ (action #6a and 6b) Applying admissible components (action #5) Frame amendment (action #8)

As in section 8.3, the breakdown in Table 8–3, gives a concise account of our interpretation of the recorded sequence of design episodes. For clarity, the interpretation is conducted using the vocabulary from chapter 6. The legend associates the referential numbers with the verbal descriptions. The columns in the table correspond to the individual design episodes of the narrative as described earlier in section 8.4 (ranging from *a* to *l*). The values in the individual cells reflect the 'order' in which the different reasoning steps were observed – if such ordering was possible to be interpreted from the recorded threads. As we already mentioned, the narratives (columns labelled with small letter *a* to *l*) serve as a lead between the explicit contextual 'milestones' of the design process (marked with capital DC-s).

Table 8–3. Conceptual summary of task T21 (see reasoning steps 8.4a to 8.4l)

'action'	a	b	DC-2	c	d	DC-3	e	DC-4	f	DC-5	g	DC-6	h	k	l	DC-7
1a	1.		1.			1.		1.		1.		1.				1.
1b			1.			1.		1.		1.		1.				1.
2	2.	1.		3.					3.							
3a	3.			4.					4.						2.	
3b		2.			1.				4.						2.	
4		4.		1.							1.					
5			3.			3.	4.	3.	6.	3.	3.	3.	4.	3.	3.	3.
6a		2.	2.		2.	2.	3.	2.	5.	2.	2.	2.	3.	2.		2.
6b		3.	2.	2.	2.	2.		2.	5.	2.	2.	2.	3.	2.		2.
7							1.		1.		1.		1.	1.		
8							2.		2.				2.		1.	

Legend to Table 8–3:

Reference	Verbal description of referenced action
1a	Articulate frame → *specifies(S,* $\mathfrak{DP}$*)*
1b	Articulate frame (conceptual part $\mathfrak{T}$)
2	Retrieve similar frame ($\phi = \langle\tau, \theta\rangle$)
3a	Articulate *S*, $\mathscr{S}$ similar to θ
3b	Articulate *T*, $\mathfrak{T}$ similar to τ
4	Is re-used τ and/or θ *admissible*?
5	Exists such *T* that *satisfies* given *S*
6a	Develop *S*, $\mathscr{S}$ into details (incl. alternatives)
6b	Develop *T*, $\mathfrak{T}$ into details (incl. alternatives)
7	Is candidate solution *T acceptable*?
8	Re-interpret current frame (i.e. *S* → *S*')

8.6 Visualised summary of tasks T11 and T21

We conclude chapter 8 that discussed the annotation and analysis of the empirical study by summarising the analytic results. We visualise each of the annotated and analysed problems, and show them as a 'design narrative'. In other words, we put the individual design contexts (*DC-s*) into a single coherent flow of design episodes (*a*, *b*, *c*, etc.). In turn, we want to visualise also the sequence of the individual episodes; in this case, the coherence is achieved by chaining the individual reasoning steps (*1a*, *1b*, *2*, *3a*, etc.) in between the design episodes.

The visualisation is presented in the following two figures. First, Figure 8–14 shows the 'control flow' creating the 'narrative' structure for task T11 – design of an active shock absorber. Next, the diagram in Figure 8–15 depicts a similar 'narrative' constructed for the task T21 – design of a paper-smoothing plant. The symbols used in the figures are very straightforward and relate directly to the definitions and glossary of terms introduced in section 8.1. Thus, the three different shapes correspond to three different 'levels' of annotation and analysis – design contexts, design episodes and the reasoning steps.

Design contexts are shown as hexagons, and each 'line of reasoning' begins and ends with the design contexts. Each context is labelled with the same name as appears in Appendix C and sections 8.2 to 8.5. Design episodes are symbolised by rectangles, and they are also labelled in accordance with the references used in Table 8–2 and Table 8–3. Finally, the lowest level of the RFD analysis recognised the reasoning steps, these are marked as circles in the diagrams, and are labelled with their particular 'referential values' (i.e. *1a*, *4*, *6b*, etc.) that are translated in the legend above.

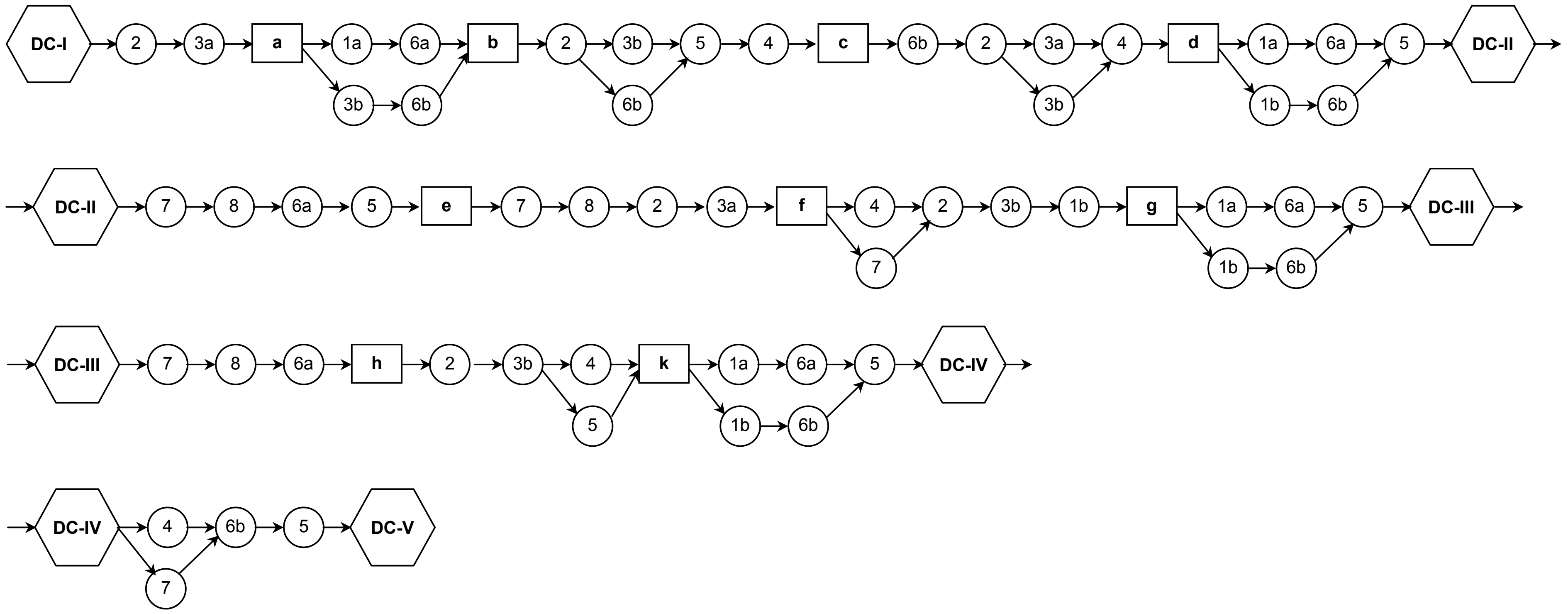


Figure 8–14. Flow of design ‘narrative’ for design problem T11 (active shock absorber)

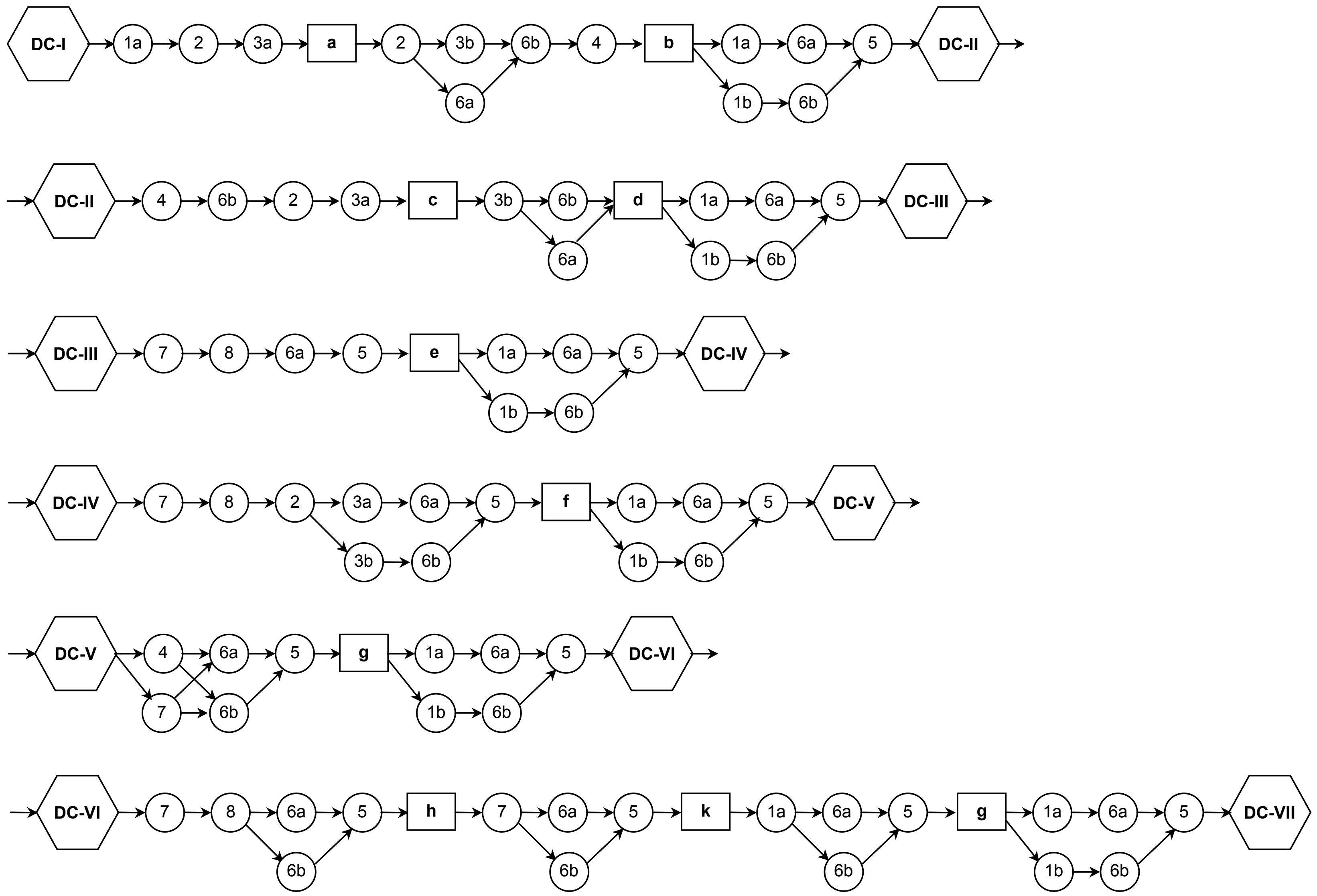

DC-I 1a 2 3a a 2 3b 6a 6b 4 b 1a 6a 1b 6b 5 DC-II
DC-II 4 6b 2 3a c 3b 6b 6a d 1a 6a 1b 6b 5 DC-III
DC-III 7 8 6a 5 e 1a 6a 1b 6b 5 DC-IV
DC-IV 7 8 2 3a 6a 3b 6b 5 f 1a 6a 1b 6b 5 DC-V
DC-V 4 7 6a 6b 5 g 1a 6a 1b 6b 5 DC-VI
DC-VI 7 8 6a 6b 5 h 7 6a 6b 5 k 1a 6a 6b 5 g 1a 6a 1b 6b 5 DC-VII

Figure 8–15. Flow of design ‘narrative’ for design problem T21 (paper-smoothing plant)

9 RESULTS AND IMPLICATIONS

As was mentioned in chapter 7, we conducted 24 experiments featuring non-trivial design situations and scenarios. The data captured during all the experiments were annotated and analysed as shown in chapter 8. We gave details of only two experimental sessions because of the spatial constraints imposed by the size of this thesis. Nevertheless, using the methodology and instructions for the annotation of raw data from section 8.1, we produced a set of tabular summaries for each scenario similar to those in Table 8–2 and Table 8–3. It is the purpose of this chapter to present these findings and summaries in a concise manner.

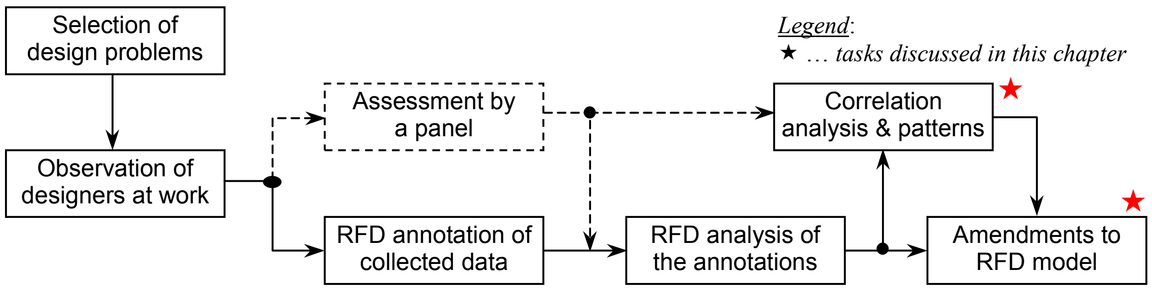

Selection of design problems
Observation of designers at work
Assessment by a panel
RFD annotation of collected data
RFD analysis of the annotations
Correlation analysis & patterns
Amendments to RFD model
Legend:
★ … tasks discussed in this chapter

Figure 9–1. Phases of the experimental study tackled in this chapter

According to Figure 9–1, which accompanies all chapters dealing with the experimental study, this chapter is mainly concerned with the identification of interesting patterns and their analysis. However, we also propose a few changes or improvements to the original RFD model, thus addressing the last ‘box’ in the flowchart. First, we present and discuss the overall summaries in section 9.1. In this discussion, we identify certain sequences typically occurring in the design process, and relate them to the individual predicates, sequences of predicates, and the characteristics of design. The significant patterns are then super-imposed on the preliminary assessment of the design sessions that was presented earlier in section 7.4.

Several interesting correspondences are presented in section 9.2. In that section, we also show how the innovative/routine distinction is performing in the conceptual terms of our RFD model. Next, we draw conclusions from these patterns and associations, and return to the original recursive model of framing in design. Although the experimental findings uphold the feasibility of the initial RFD model, we believe there are certain implications of the patterns discussed in section 9.2 in respect of the completeness of the model. Therefore, section 9.3 proposes a few extensions or refinements to the model that was originally introduced in chapter 6. These conclusions are presented very briefly and are elaborated in more detail in chapter 10.

9.1 *Summary of the experimental findings*

We concluded the previous chapter discussing the results of the experimental work with the visualisation of the sequence of reasoning actions that were observed in the two annotated and analysed experiments. Similar visual maps can be made for the remaining design scenarios. However, this form of result presentation is rather lengthy and difficult to summarise. Therefore, this section presents a simplified representation of the results in a succinct graphical form. The inspiration for the presentation came from a historical source known as the statistical graph of Napoleon's fatal march to and from Russia. It was devised by Charles Minard who graphically depicted the fate of Napoleon's army in Russia in 1816. His aim was to visualise the rapidly decreasing strength of the army over a period of six months. His graph associated the time, strength, direction of marching, and the temperatures. As a reference, see for instance (Tufte 2001) or http://www.stat.ucla.edu/history/march.htm. Our re-use of the metaphor is based on the following 'dimensions' and simple rules:

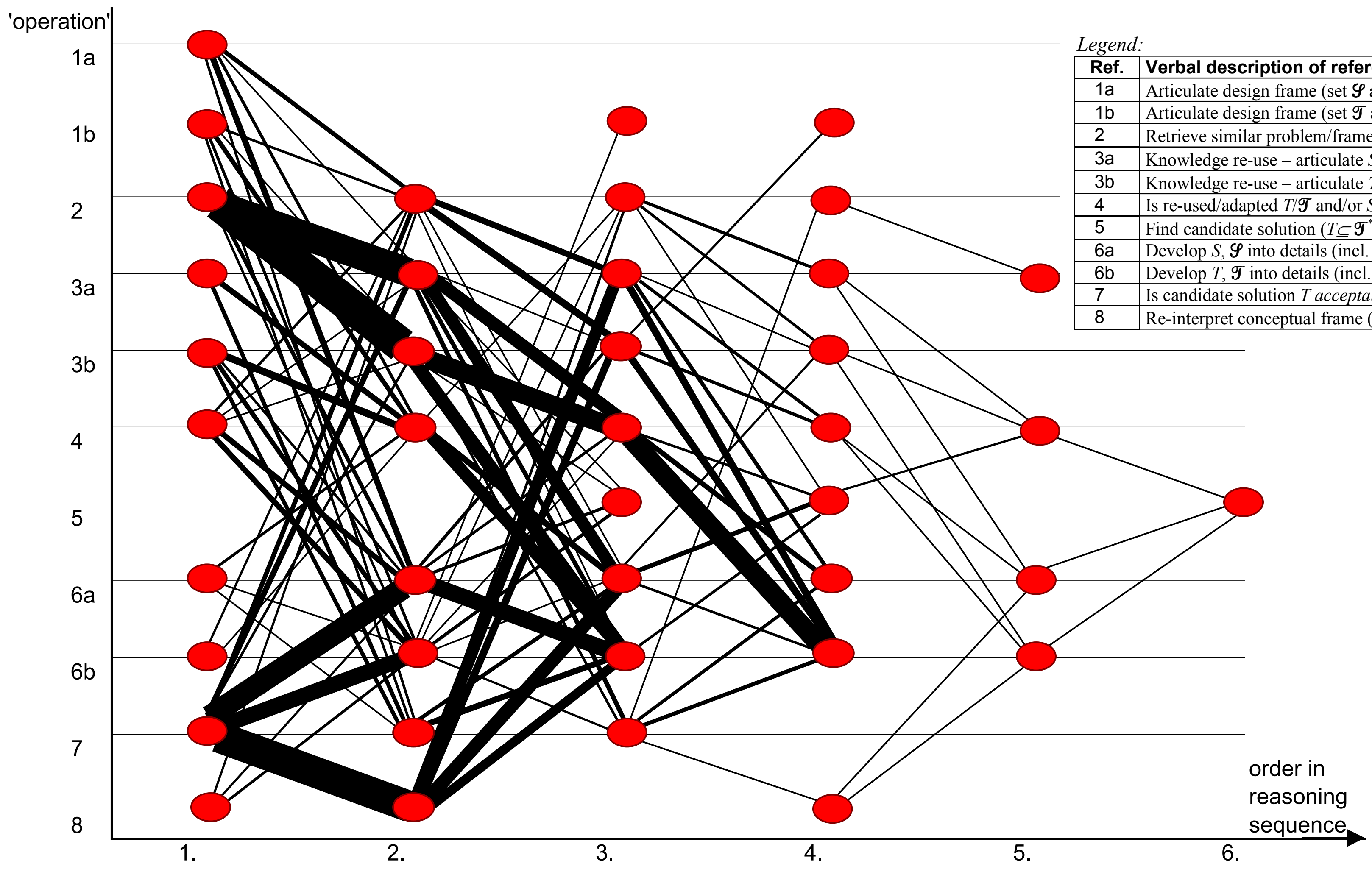
- The X-axis depicts the sequential order, in which the actions took place; the allowed values are discrete numbers (e.g. *1. step*, *2. step*, etc.); in our graph, there is a maximum of six steps;
- The Y-axis depicts the reasoning steps that were used for the RFD analysis of the design sessions (e.g. *1a* – articulation of a requirement, *7* – acceptability check, etc.);
- A bullet located in such a space of co-ordinates corresponds to an operation Y being observed in step X (e.g. *7* – acceptability check observed as *2. step*)
- A line between the bullets in the two consecutive steps corresponds to an occurrence of the particular sequence of reasoning steps (e.g. step *7* – acceptability check *immediately followed* by step *6a* – specification refinement);
- One occurrence of a particular sequence of reasoning steps (e.g. *7* → *6*) corresponds to a line that is ¼ of a point (0.25pixel) thick;
- Repeated occurrence of the same sequence of operations observed between the same two steps makes a particular line thicker; the increment step is always ¼ of a point per sequence;
- Thus, the thicker a line is, the more frequently a particular sequence of actions was observed and annotated.

Figure 9–2 shows the graph of reasoning sequences in its original, full form. As it is visible from the figure, the number of potential transitions between various nodes is very large. Altogether, there are 104 different types of transitions (links) depicted in the graph. The line intensity ranges from the thinnest ¾ of a point line to the thickest 15 ¼ of a point. In order to make the graph more legible, we also present versions that take into account various thresholds later in this section.

9.1.1 Conceptual terms and assumptions in the graph

Let us now interpret the results depicted in the summarising graph in Figure 9–2. First, we look at the characteristics of design as enumerated in chapter 4. As in section 7.4, we attend to these characteristics in four separate categories. These are as follows (the order is changed deliberately so that it is possible to present the finding in a meaningful way that would reflect the causality of the particular reasoning sequences):

a) Presence of knowledge transfer featuring re-use of the familiar 'frames';
b) Presence of the phenomenon of co-evolving design specifications and solutions;
c) Re-formulation of design problems (incl. articulation of additional requirements/constraints);
d) Reflection on the partial design solutions (and solution candidates)



Legend:

Ref.	Verbal description of referenced action
1a	Articulate design frame (set $\mathcal{S}$ and $S\subseteq\mathcal{S}$: *specifies*$(S, \mathfrak{DP})$)
1b	Articulate design frame (set $\mathcal{T}$ and/or $\mathcal{T}^*$)
2	Retrieve similar problem/frame ($\phi_{SIM} = \langle\tau, \theta\rangle$)
3a	Knowledge re-use – articulate S, $\mathcal{S}$ similar to θ
3b	Knowledge re-use – articulate T, $\mathcal{T}$ similar to τ
4	Is re-used/adapted $T/\mathcal{T}$ and/or $S/\mathcal{S}$ *admissible*?
5	Find candidate solution ($T\subseteq\mathcal{T}^*$: *satisfies*(T, S))
6a	Develop S, $\mathcal{S}$ into details (incl. alternatives, refinements)
6b	Develop T, $\mathcal{T}$ into details (incl. alternatives, refinements)
7	Is candidate solution T *acceptable*?
8	Re-interpret conceptual frame (i.e. $\Phi \rightarrow \Phi'$)

Figure 9–2. Occurrence of individual sequences of reasoning steps aggregated over all experimental sessions (each occurrence = ¼ point thick link)

a) Knowledge transfer and familiar frames

This assumption category covers both the application of the previous experience in framing and solving the design problem, as well as its various forms. Under the 'knowledge transfer' category we refer to methods and strategies such as analogy- and similarity-based reasoning, prototype and model re-use, or the application of explicit 'good practice'. As is visible from the 'pattern' selected and highlighted in Figure 9–3, the knowledge transfer sequence occurred very frequently in the observed design experiments. Thus, we may conclude that this category of assumptions was upheld in the experimental work. Moreover, it is possible to recognise a typical 'forking' pattern of knowledge re-use that is repeated not only at the beginning of the design episode, but also in later steps.

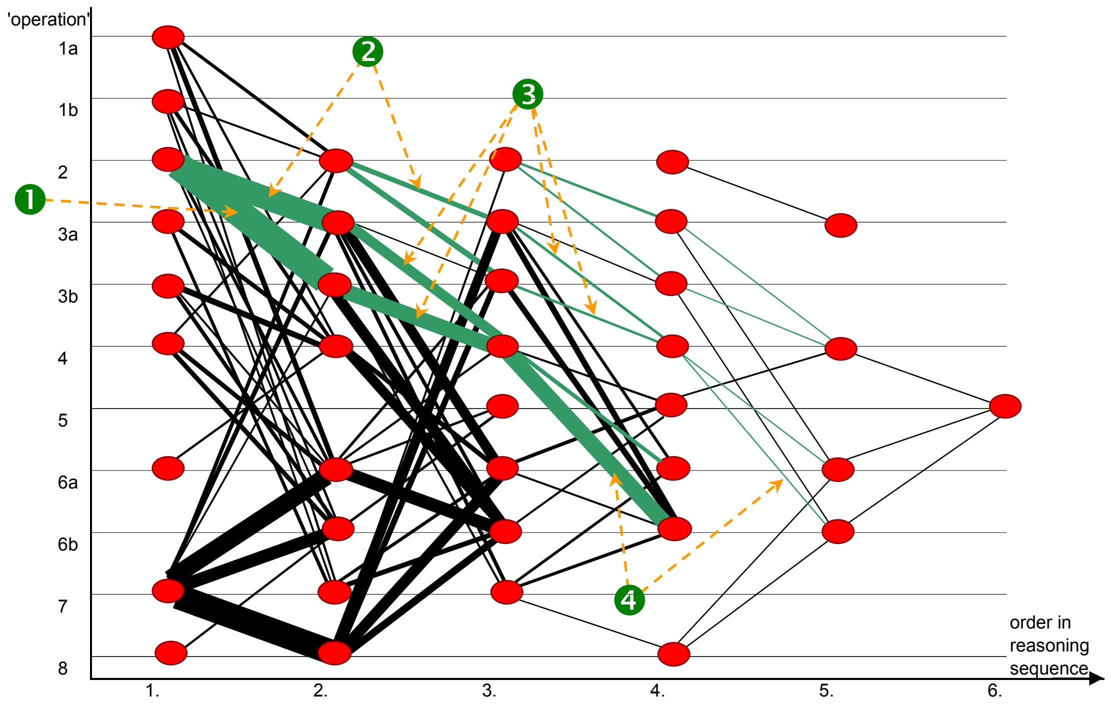


Figure 9–3. Pattern of knowledge retrieval – re-use – application

The 'knowledge transfer' pattern consists of the following reasoning actions that typically co-occurred in our annotations. The process begins with an identification of a 'familiar frame'; i.e. a past design case, a design prototype, or a model of a controller. The past frame may be then used to define the conceptual objects the designer intends to use for the construction of a solution in the current problem – i.e. the conceptual base $\mathcal{T}$. This transition is depicted in Figure 9–3 as a link between action *2* (retrieval of past frame ⟨τ, θ⟩) and action *3b* (articulation of $\boldsymbol{T}$ or $\mathcal{T}$ based on previous τ). Altogether, this transition occurred 61 times in step 1, 13 times in step 2, 4 times in step 3, and is labelled by ❶.

Another transition observable as a part of knowledge transfer connects action *2* (retrieval of past frame ⟨τ, θ⟩) and action *3a* (articulation $\boldsymbol{S}$ or $\mathcal{S}$ based on previous θ). This transition occurred slightly less frequently than the previous one (51 times in step 1, 11 times in step 2, and 8 times in later steps). In the figure, it is depicted by a link labelled ❷. This 'fork' clearly supports our opinion presented earlier in chapter 5 that the past cases may serve as models for both the problem specification and the solution construction.

Typically, these actions documenting knowledge re-use in either of the two knowledge spaces forming a design frame were followed by an assessment of the relevance of a piece of retrieved knowledge for the current problem. This was typically a check of compliance with already existing domain objects, partial solutions or constraints. As argued in section 6.4, such an explicit assessment can be performed by a formal theory featuring TMS, ATMS or constraint satisfaction techniques. In our model, it is represented by a 'merger' of two links starting in nodes *3a* and *3b*, and ending in node 4 (admissibility check). The occurrence of assessing the re-used specification (*3a* → *4*) is 10 times in step 1, 23 times in step 2, and 6 times in later steps. A similar breakdown for transition *3b* → *4* is: 13 times for step 1, 34 times for step 2, and 6 time in later steps. See also the links labelled as ❸.

However, an interesting observation from the aggregated results shows that after the admissibility assessment, typically one more step was conducted. Re-used components may have been refined or, in the terminology of CBR research, they were adapted (see also chapters 4 and 5). We believe that this adaptation is represented in Figure 9–3 by a frequent occurrence of a transition from action *4* (admissibility check) to actions *6a* or *6b* (refinements of current $\mathcal{S}$ or $\mathcal{T}$). Altogether, this 'follow-up' after the explicit assessment of the transferred knowledge was observed 35 times in step 2 and 22 times in step 3. See the link with label ❹.

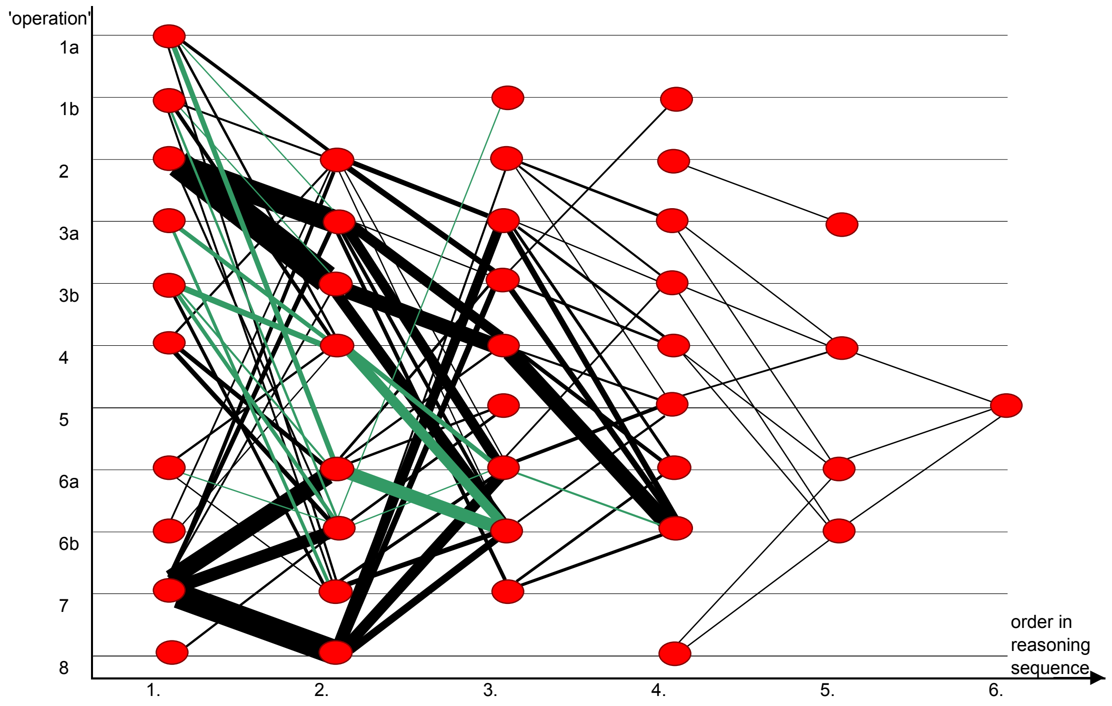


Figure 9–4. Several patterns showing co-evolution of problem specification and solution

b) Co-evolving specification and solution

It is also possible to confirm a pattern corresponding to the phenomenon that was referred to in the previous chapters as 'co-evolution of design specification and solution' from the aggregated results. Multiple 'paths' in the graph can be associated with this particular feature of a design process. They are highlighted in Figure 9–4 in green. In general, these pattern have one thing in common, they are represented by links starting in nodes *1a/3a/6a* (articulation of $\boldsymbol{S}$ or $\mathscr{S}$) and ending in nodes *1b/3b/6b* (articulation of $\boldsymbol{T}$ or $\mathscr{T}$), or vice-versa.

The variability in the location of this 'co-evolution' pattern is mainly dictated by the event that triggered the re-formulation or refinement in the first place. Thus, the most frequent is the development of a solution ($\boldsymbol{T} \subseteq \mathscr{T}^*$) based on the similarity of the retrieved problem specification and the current problem specification ($\boldsymbol{S} \subseteq \mathscr{S}^*$). This transition may have occurred with or without an admissibility check, and in the graph is represented by sequences *3a* $\rightarrow$ *6a/6b* or *3a* $\rightarrow$ *4* $\rightarrow$ *6a/6b*. A verbal account of such transitions is that after the assessment of a particular previously-known requirement, it was usually adapted for the current problem, and its implications were drawn in the space of design solutions.

The interaction also occurred in the reverse order; i.e. the transitions of the type *3b* $\rightarrow$ *6a* or with a check *3b* $\rightarrow$ *4* $\rightarrow$ *6a*. However, judging from the frequency, this pattern is less usual than the above-mentioned one. Nevertheless, this is not a shortcoming of our theory. It supports our position that this solution 'backtalk' is often conducted on an implicit or tacit level, and features some kind of acceptability assessment of the partial solution rather than the explicit admissibility.

Indeed, if we subscribe to this fundamental assumption of our RFD theory, we can document it by an unusually thick link between node *7* (acceptability check) and *6a* (refinement of $\boldsymbol{S}$ or $\mathscr{S}$). We explain this particular pattern by a difference between a *routine* refinement of the problem specification and/or solution (i.e. transitions *3a/b* $\rightarrow$ *6a/b*) and the *reflective* refinement of the designer's understanding (i.e. transitions *7* $\rightarrow$ *6a/b*). In other words, the co-evolution very often took place *between the two* (or more) consecutive design episodes, rather than *inside a single* one. More on this phenomenon is presented also in point d) further down.

c) Problem re-formulation, new requirements/constraints/concepts

Re-formulation and refinement of designer's understanding of the design problem is also supported by the aggregated results from the experimental study. First, there are the 'usual' ways of introducing new requirements and/or constraints to the current design situation. The term 'usual' in this context refers to the knowledge re-use (described in point a) earlier) that is observed early in the particular design episodes and early in the design session. In the terminology of our RFD model, it is more suitable to refer to this 'operation' as the *initial interpretation* (of requirements/constraints).

Even if we disregard the initial interpretation and commitment to the explicit knowledge about the particular design problem, the highlighted pattern in Figure 9–5 shows another strong 'backtalk'. The origin of this *re-interpretation* or amendment of the current understanding of the design problem is in the acceptability of the current solution (node *7*). It seems that in many cases, the acceptability check failed to satisfy the designer's tacit expectations or intentions, and the designer explored the reasons. To that extent he may have looked at the problem from a different angle; i.e. the transition *7* $\rightarrow$ *8* labelled as ❶ occurred very frequently as a 'first aid' to save the design.

Nevertheless, the designer did not end his exploration with a different view. After shifting or amending his current frame, he had several choices. These are indicated by labels ❷, ❸, ❹, ❺, and ❻. As we can see, the most frequent was the transition resulting in the refinement of an existing statement in the explicit problem specification (node 6a and label ❺ – 24 times). This was followed by a more radical problem re-interpretation featuring introduction of a new statement to the explicit problem specification (node *3a* and label ❸ – 21 times). An interesting link in this 'fan' pattern is the one labelled as ❷ that leads to recalling the previous knowledge (node *2*). Although, this link does not prove or document the re-interpretation directly, it supports the position of introducing totally new, less common but still familiar views to resolve the tacit unacceptability of a particular sound solution.

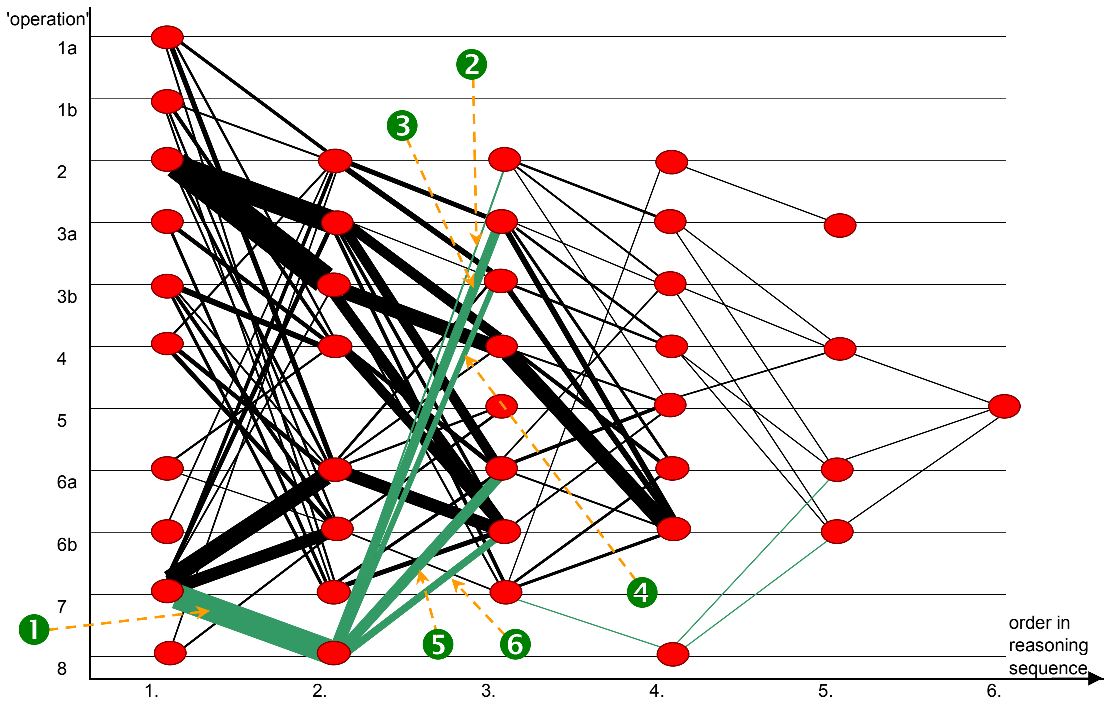


Figure 9–5. Highlighted *re-interpretations* of the problem understanding

d) Reflection on partial design solutions

Finally, let us conclude this generic interpretation of the aggregated results by pointing out the most interesting characteristics of design – reflection on partial solutions. As we already mentioned in the previous category, the presence of solution 'backtalk' or reflective appreciation of a partial solution is typically observable between two consecutive design episodes. In order to provide a better picture, we present this information as a simple bar chart in Figure 9–6.

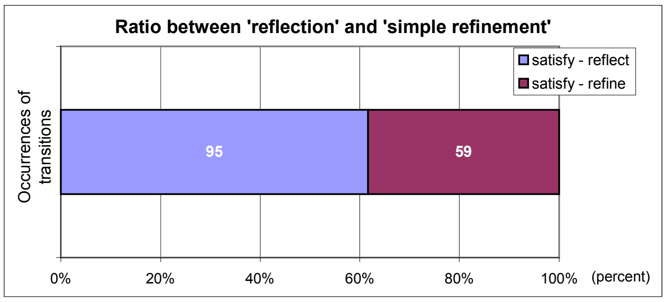


Figure 9–6. Reflection and solution 'backtalk' vs. solution refinement

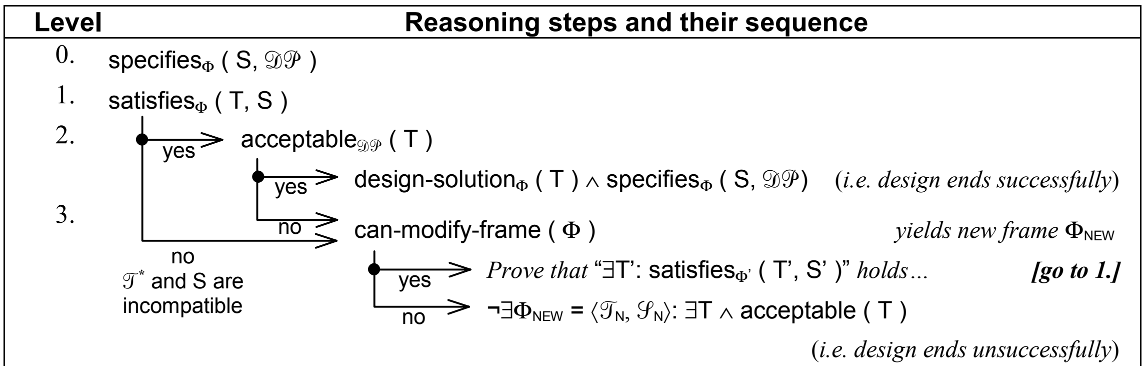
First, we went through the individual analyses of the design sessions and counted how many times the explicit commitment to a (partial) solution was followed by the appreciation of acceptability and the decision to extend/refine the current perspective. The first causal relationship corresponds to a link from node *5* to node *7*, as is depicted in Figure 8–14 or Figure 8–15; the second one denotes the direct transition from *5* to *3* or *6*. Note that we deliberately did not include these ‘inter-episodal’ transitions in the aggregated summary as shown in Figure 9–2. The essential reason is that Figure 9–2 depicts what happened *inside* any particular episode, whereas the reflective solution ‘backtalk’ (Figure 9–6) was observed *between* two episodes. Moreover, we did not look at any arbitrary episode pairs but focused on what happened after the explicit commitment to a particular context (in chapter 8, these explicit commitments are recorded as ‘DC’ entities, e.g. 8.3/DC-II and 8.5/DC-VI).

First, Figure 9–6 shows that it was possible to distinguish between the episodes in which the proposed partial solution was only elaborated into details and those episodes featuring inarticulate decisions. We think that it is appropriate to talk about ‘reflection’ as a knowledge-level representation of the acceptability assessment by a designer. Thus, transitions corresponding to the ‘satisfy → reflect’ portion from Figure 9–6 are depicted in Figure 9–2 as paths starting from node *7* (acceptability check). Any other refinement or elaboration of a partial solution is labelled as ‘satisfy → refine/extend’ in Figure 9–6, and in Figure 9–2, the source of such refinements are nodes *1b*, *3b* or *6b*.

Nevertheless, the bar graph depicting the ratio of inarticulate, tacit exploration versus explicit and straightforward refinements supports our position that design is a reflective process. Following possible paths after the check of tacit acceptability, we may confirm that the process of design framing is also reflective. Above all, regardless of whether the solution was extended, refined, or re-interpreted in a new frame, the bar graph in Figure 9–6 supports the use of the adjective *recursive* in our *Recursive* model of Framing in Design. Alternatively, we may talk about iterative design processes. The observed design sessions exhibited between four and ten such *iterations* from a partial solution towards a more complete and better-understood one.

9.1.2 RFD model supported by the aggregated graph

After discussing the aggregate results and their visual representation from a more generic and abstract perspective, we look at how these results support the RFD model of design as proposed in chapter 6. First, we present the original recursive model of framing in design in order to see its essential points next to the interpreted aggregated results. Next, the individual steps and decisions that were defined for the model are attended to step-by-step, similarly as we did in the previous section.



Level
Reasoning steps and their sequence
0.
specifies$_\Phi$ (S, $\mathscr{DP}$)
1.
satisfies$_\Phi$ (T, S)
2.
yes
acceptable$_{\mathscr{DP}}$ (T)
yes
design-solution$_\Phi$ (T) $\wedge$ specifies$_\Phi$ (S, $\mathscr{DP}$) (*i.e. design ends successfully*)
3.
no
can-modify-frame (Φ)
yields new frame Φ_{NEW}
no
$\mathscr{T}^*$ and S are incompatible
yes
Prove that “$\exists$T’: satisfies$_{\Phi'}$ (T’, S’)” *holds…*
[go to 1.]
no
$\neg\exists\Phi_{NEW} = \langle\mathscr{T}_N, \mathscr{S}_N\rangle$: $\exists$T $\wedge$ acceptable (T)
(*i.e. design ends unsuccessfully*)

Figure 9–7. Recursive model of framing in design (from chapter 6)

$$\textit{can-modify-frame}\ (\Phi) \Leftrightarrow \exists\Phi_{NEW} = \langle \mathcal{T}_N, \mathcal{S}_N \rangle : S' \subseteq \mathcal{S}_N^* \wedge \textit{specifies}_{\Phi'}\ (S', \mathfrak{DP}) \qquad \textbf{Eq. 9–1}$$

Our original proposal of a recursive model featuring framing and re-framing in design was presented as a sequence of three mutually interacting knowledge-level actions. These corresponded to three basic predicates or relational statements, namely: *specifies*, *satisfies*, and *acceptable*. The mutual interaction between these predicates was achieved by an interchange of data and control, as depicted in Figure 9–7. An auxiliary predicate '*can-modify-frame*' serves as an abbreviation of the longer version that is defined in Eq. 9–1 below the model. Let us associate the aggregated results from Figure 9–2 with the RFD model using the individual levels providing a structure to the presentation.

Level 0. s*pecifies*$_\Phi$ *(S, $\mathfrak{DP}$)*

According to the definition, level labelled as zero denoted the articulation of an initial conceptual frame for the design problem $\mathfrak{DP}$. The construction of an initial frame started with a reference to the familiar vocabulary and continued with an attempt to describe the current problem $\mathfrak{DP}$ in such vocabulary. The result of this initial *framing* was an explicit articulation and commitment to a pair of two conceptual sets $\mathcal{T}$ and $\mathcal{S}$. Set $\mathcal{T}$ contained the basic conceptual terms the designer decided would be available for solving the design problem $\mathfrak{DP}$. Set $\mathcal{S}$ contained the same for problem specification.

The initial framing of the design problem $\mathfrak{DP}$ in the familiar terms was indeed observed during the experimental sessions. Typically, a design session (e.g. T21 in section 8.4) started with a selection of terms from the design brief that expressed the customer's requirement(s); in the terminology of our model, this was action *1a* (articulation of explicit set $\boldsymbol{S} \subseteq \mathcal{S}^*$). The next or the alternative step referred to a familiar frame, from which the designer extracted further details to extend the design brief, translate it into engineering language, and define main conceptual blocks for a solution construction. In other words, the sessions could also start with the action *2* (knowledge retrieval), and typically continued with an adaptation of the previous specification or conceptualisation (*3a* and *3b*).

Thus, level 0 is confirmed in the aggregate results, and since it was already discussed earlier in section 9.1.1, part a) we are not going to repeat the same arguments and respective figures.

Level 1. *satisfies*$_\Phi$ *(T, S)*

According to the definition of predicate '*satisfies*', the main purpose of this reasoning step was to find an admissible solution. A solution should be admissible in respect to a particular explicit problem specification $\boldsymbol{S}$ that was decided on the previous level 0 as sufficiently specifying the problem $\mathfrak{DP}$. The complete or more often partial solution $\boldsymbol{T}$ was sought in the applicable problem solving theory $\mathcal{T}^*$. We did not investigate the methods how this solution construction happened in the particular sessions, what we were interested in was the explicit commitment to something that could be viewed as a candidate solution.

In the annotated sessions, the occurrence of this particular 'milestone' was labelled as an explicit commitment to a particular problem specification ($\boldsymbol{S}$) and initially contained between two and four key requirements. In a majority of cases, this early specification was accompanied by an abstract drawing depicting the intended approach to the problem – a simple solution model ($\boldsymbol{T}$). The second explicit commitment contained a solution model in all design sessions. For instance, session T21 (section 8.4) produced the first abstract model in the explicit context 8.4/DC-II as a result of choosing admissible concepts during the preceding *framing* operations (see episodes 8.4a and 8.4b).

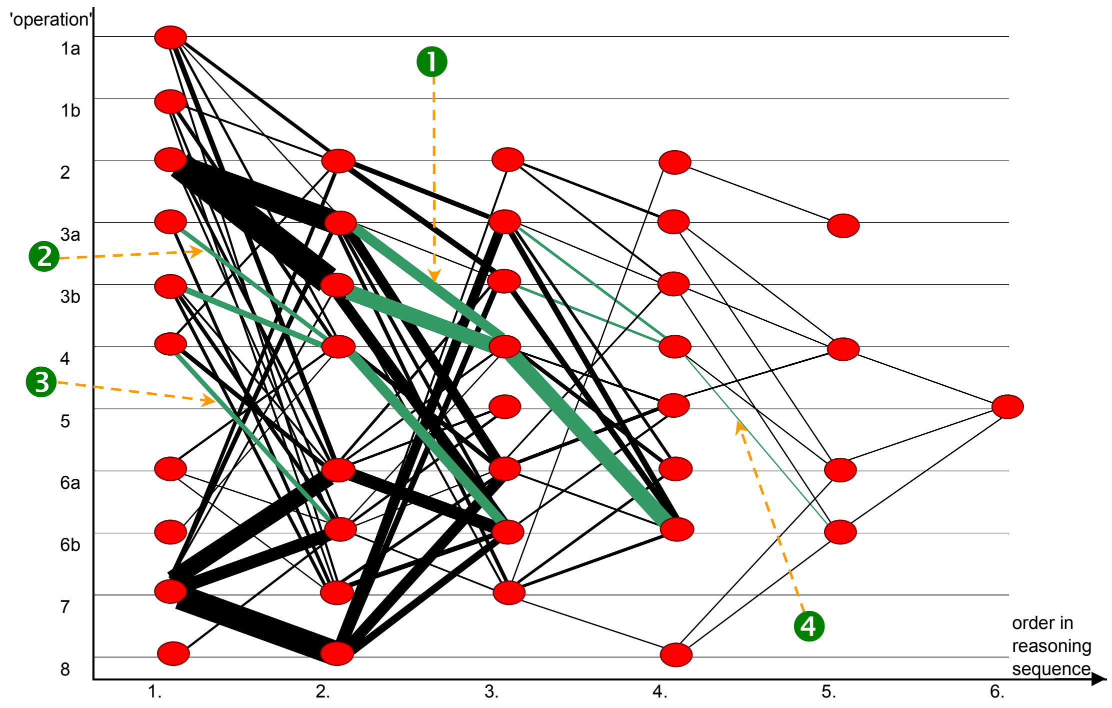


Figure 9–8. Pattern for a construction of initial admissible solution

Taking into account the aggregate results, the pattern depicted in Figure 9–8 and labelled as ❶ shows a representative sequence that can be associated with the predicate '*satisfies(T, S)*' in the early phases of design. Since this predicate was allowed to occur also in the later phases, the alternative representative patterns are also shown in Figure 9–8 with labels ❷, ❸ and ❹. Nevertheless, the aggregate results seem to confirm the sequential order and the position of predicate '*satisfies*' in respect to '*specifies*' or '*modify-frame*' (this will be discussed further down).

Level 2. *acceptable*$_{\mathfrak{DP}}$ *(T)* | Φ

The next level according to the definition of RFD model is concerned with the appreciation of the solution tentatively proposed as a result of the '*satisfies*' phase. We argued that a designer may work within a particular conceptual frame and try several alternative solutions, each of them being explicitly sound and admissible in respect to the explicit constraints. However, once the commitment was made to a particular candidate solution, its compliance with the inarticulate and not-articulated expectations needed to be assessed. We defined this alignment of the explicit conceptual frame and the tacit expectations as a solution *acceptability* check. We also argued that here the designer assessed the solution ***T*** against the 'real' problem $\mathfrak{DP}$, rather than the explicit requirements and constraints ***S***.

As we already mentioned in section 9.1.1, part c) and d), the acceptability check was an important decision the designer was making regularly during a design session. For instance, session T21 (section 8.4 is a typical representative showing five occurrences of some kind of acceptability assessment. In the aggregated graph, the acceptability check can be associated with the paths starting from node *7* (e.g. a typical sequence is *7* → *8* → *6a*). See also highlighted patterns in Figure 9–9.

Figure 9–9 on one hand confirms the correct location of the solution acceptability assessment in the design process. However, the next important assertion by the RFD model argues that if the solution acceptability check fails, it points to an incomplete conceptual frame ***Φ***. Our initial model suggested that a potential remedy of an unacceptable solution might lie in the articulation of a new conceptual frame ***Φ'***. In the aggregated graph (Figure 9–9), this causal relationship between the acceptability check (node *7*) and a reference to new conceptual terms of an amended design frame (node *8*) is clearly visible. This is the thick transition labelled as ❶, and highlighted in light blue.

After the amendment of a conceptual frame, the designer could make appropriate modifications or refinements in the current explicit problem specification (sets ***S*** and $\mathcal{S}$) or problem solving theories and models ($\mathcal{T}$ and ***T***). We leave these amendments for the discussion on the next level. Nevertheless, we want to show one very important finding that seems not to be entirely covered by the proposed RFD model. Namely, the pattern of transitions starting in node *7* (acceptability check) and continuing toward nodes *6a* or *3a* (refinement or extension of an explicit problem specification), and in parallel the link between node *7* and *6b* (solution refinement). See labels ❷ and ❸ in Figure 9–9.

These transitions were observed frequently, and they differ from the scenario dictated by RFD model in the fact that no particular frame amendment accompanied these extensions/refinements. Thus, it seems that there are several choices for a designer to choose from after assessing the acceptability of a solution. First, he may commit to a new or modified frame. Alternatively, he may refine the problem specification based on the tacit expectations. We return to this particular pattern later in sections 9.2 and 9.3, where the original RFD model is re-visited and extended. Nevertheless, the essence of the original model – a presence of a tacit acceptability check was confirmed by the experiments, too.

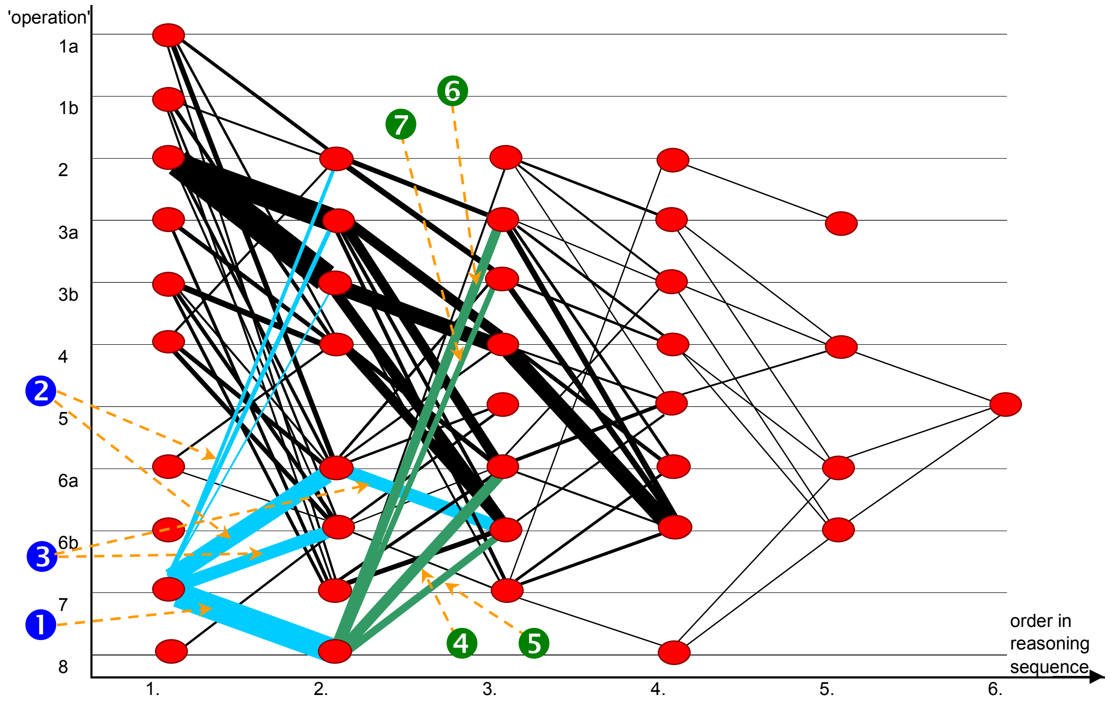


Figure 9–9. Patterns from design episodes showing *acceptability* check

Level 3. ***can-modify-frame (Φ)***

As mentioned earlier, this knowledge-level action directly follows the acceptability assessment that was described in the previous paragraphs. We also mentioned that there were several paths leading from the node *8* in the aggregated graph; see the green links in Figure 9–9 for reference. Some of these follow-up actions aimed to <u>refine</u> the designer's current understanding of the problem (e.g. link ❹ pointing to step *6a* and ❺ pointing to *6b*). Alternatively, the designer could commit himself to more significant <u>extension</u> and/or <u>addition</u> to the explicit problem specification or solution (see the link labelled as ❻ pointing to step *3a* or ❼ pointing to *3b*, respectively). All these different refinements and extensions originated in a modified conceptual frame; i.e. in the different perspective the designer chose for the investigation of the design problem $\mathfrak{DP}$.

This observation supports the correctness of the RFD model depicted earlier in Figure 9–7 in respect to the definition of a re-framing operation. Nevertheless, our original definition of *re-framing* (see) also Eq. 9–1 does not account for the observed variability. In principle, the predicate defining the operation of re-framing is correct but we believe it can be further refined in response to the empirical observations. Note that this newly emerged pattern is already incorporated into our RFD model; only it is defined rather abstractly and superficially. Therefore, in sections 9.2 and 9.3, and later in chapter 10 we make a clearer distinction between the *refinement* and *extension* of the designer's understanding.

Recursion and iteration (level 3. → level 1.)

Clearly, the amendment of a conceptual frame was not conducted self-purposely. Its objective was to bring in something new into the explicit knowledge about the design problem. This 'something new' corresponded to a construction of an exploratory probe whose aim was to restore or achieve the acceptability of a design solution. Therefore typically, the designer decided to validate the tentative proposal following the acceptability checks or frame amendments, and the typical way of how to do it was to construct a new (partial) solution.

The recursion and iteration is observable in a fact that each design session featured more than one design context (see DC-II to DC-VII in section 8.4). In the analysed records, the iteration is also visible as a repeated sequence of certain design episodes leading from the old to the new explicit achievements (see Figure 8–14 and Figure 8–15 in section 8.6). Nevertheless, the same remark must be noted as in the analyses of the last two levels – the recursion did not always require the amendment of a design frame. We believe that this is a minor issue, which can be easily resolved by a more detailed definition of the predicate '*can-modify-frame*'. However, before defining the extensions to the RFD model formally, let us first show the emerging patterns that were discussed above in a broader context.

9.2 Interesting patterns

The purpose of this section is to provide further details about the frequency of the patterns that were found interesting in the previous section. We show in the following paragraphs how the ratio between the 'reflective' development and the explicit ('simple') refinement of design understanding changes for the different types of design problems. This distribution reflects the occurrences of the transitions *5* →

7 → * (i.e. satisfaction followed by an acceptability assessment) compared with transitions *5* → *3/6* (i.e. satisfy partially and refine without any acceptability checking).

Another interesting distribution that was identified on the levels 2 and 3 in the previous section would be also detailed for the different types of design problems. Thus, we present how frequently each of the following transitions occurred in a particular class of problems:

- *4* → *3/6* (admissibility check followed by an extension/refinement),
- *7* → *3/6* (acceptability check followed by an extension/refinement), and finally
- *7* → *8* → *3/6* (problem extension/refinement following in the amended frame)

In order to give a certain structure to these comparative analyses, we categorised the experimental design sessions into several complementary categories. These categories and association of a particular session with a particular category were not chosen randomly. We used the auxiliary assessment provided by a panel that was elicited *before* the RFD analysis, thus being sufficiently independent of any conclusions that were summarised in section 9.1. These findings also support the proposals for extending the RFD model that were hinted at in the previous section. The categories selected for the comparative analysis include the following:

1. Overall distributions (all experimental sessions);
2. Distributions for 'innovative' and 'routine' sessions;
3. Distributions for superficial solutions and more elaborated, complex ones;
4. Distributions for conceptually detailed and abstract solutions

Note that the classification of the individual sessions into particular categories was conducted for each particular design session based on the explicit records and drawings from that session. It is not the intention of this classification to argue that for instance, all designs of shock absorbers must always be innovative. Simply, this particular design session was found innovative or routine, simpler or more complex by the panel, and we accept this subjective decision for our detailed analyses.

Let us present first the ratios between the reasoning sequences for all the design problems that were annotated and analysed. The distribution showing the ratio between the occurrences of 'reflective' development and simpler, 'explicit' problem refinement/extension is shown in Figure 9–10. The numbers in data labels correspond to the absolute occurrences of respective reasoning sequences. In our experiment, the ratio of these occurrences was ***reflective : explicit = 61.7% : 38.3%***.

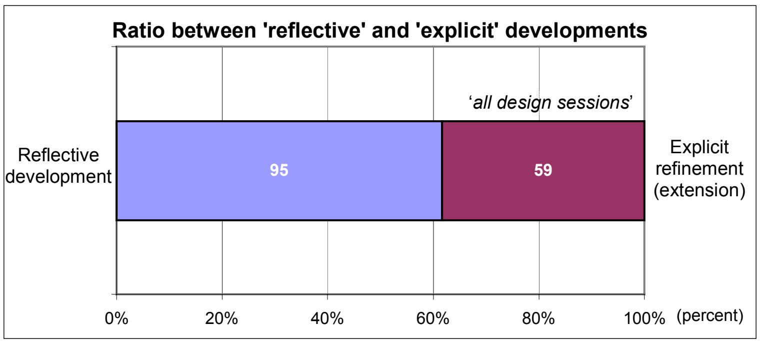


Figure 9–10. Ratio 'reflective development' vs. 'explicit' refinement/extension of design

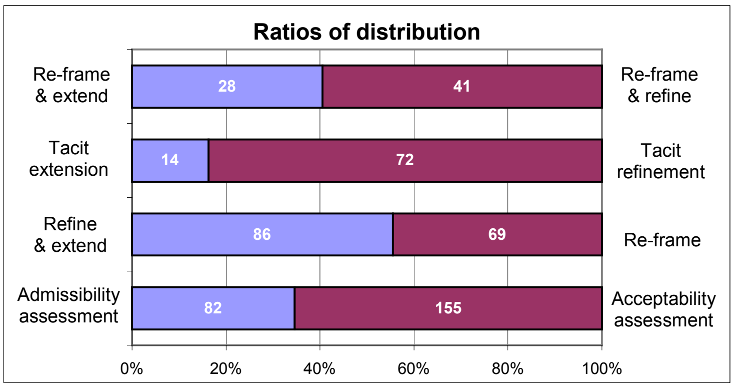


Figure 9–11. Ratios between the consequences following different assessments

The bar chart in Figure 9–11 shows a similar distribution for the operations that followed the assessment of an explicit admissibility and tacit acceptability. The ratios and percentages presented below are later compared with those for the particular classes of problems thus revealing interesting distribution patterns for the specific classes. Nonetheless, the respective overall ratios are as follows:

- Tacit assessment (acceptability) vs. explicit assessment (admissibility) = ***65.4% : 34.6%***;
- Design refinement/extension vs. re-framing following acceptability check = ***55.5% : 44.5%***;
- Tacit refinement of design vs. tacit extension of design = ***83.7% : 16.3%***;
- Re-frame & refine design vs. re-frame & extend design = ***59.4% : 40.6%***

9.2.1 Distributions between 'innovative' vs. 'routine' tasks

The classification of the individual design sessions into 'more routine' and 'more innovative' ones is based upon the overall impression provided by the panel, which was presented in section 7.4 (specifically in Table 7–1). Most sessions exhibited features that were considered by the panel as 'innovative' (16 tasks) compared to 11 tasks showing 'more routine' features. Figure 9–12 shows the distributions between the 'explicit' and 'reflective' problem development for the two classes. The classes are compared using percentage ratios rather than absolute frequencies, and for a comparison, the respective ratios are:

- More '*innovative*' problems: ***reflective : explicit = 69.5% : 30.5%*** (data series at the bottom);
- More '*routine*' problems: ***reflective : explicit = 46.7% : 53.3%*** (data series at the top)

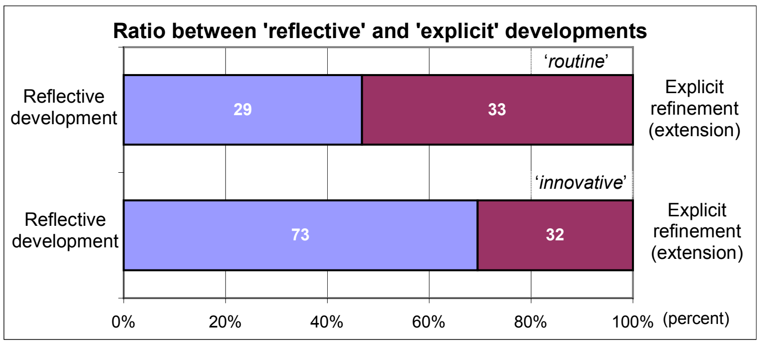


Figure 9–12. Ratio 'reflective development' vs. 'explicit' refinement/extension of design

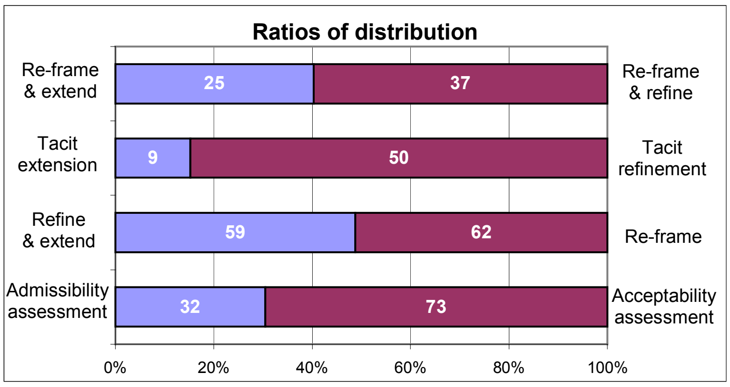


Figure 9–13. Ratios between consequences of tacit vs. explicit assessments (innovative class)

Figure 9–13 and Figure 9–14 show the distributions of the individual reasoning sequences for the 'innovative' and 'routine' problems, respectively. In order to make a comparison with the 'overall' scores easier; the percentage ratios are presented below:

Innovative problems:

- Tacit assessment (acceptability) vs. explicit assessment (admissibility) = ***69.5% : 30.5%***;
- Design refinement/extension vs. re-framing following acceptability check = ***48.8% : 51.2%***;
- Tacit refinement of design vs. tacit extension of design = ***84.7% : 15.3%***;
- Re-frame & refine design vs. re-frame & extend design = ***40.3% : 59.7%***

Routine problems:

- Tacit assessment (acceptability) vs. explicit assessment (admissibility) = ***46.8% : 53.2%***;
- Design refinement/extension vs. re-framing following acceptability check = ***70.2% : 29.8%***;
- Tacit refinement of design vs. tacit extension of design = ***81.8% : 18.2%***;
- Re-frame & refine design vs. re-frame & extend design = ***50.0% : 50.0%***

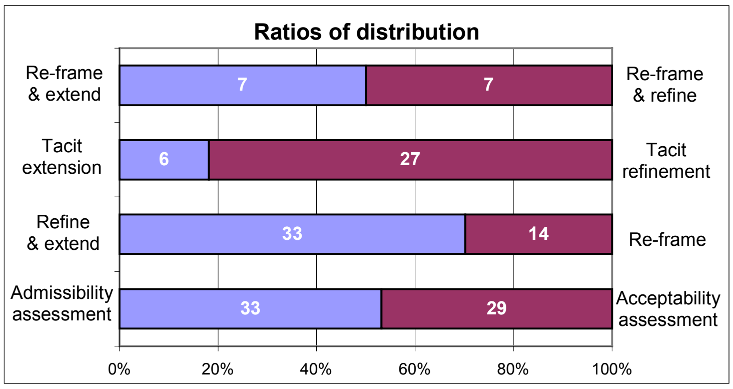


Figure 9–14. Ratios between consequences of tacit vs. explicit assessments (routine class)

The first significant difference between the two classes is in the ratio between the 'explicit' and 'reflective' problem developments. The 'innovative' problems show a ratio significantly in favour of reflection, whereas with 'routine' problems, the ratio is in favour of simpler, explicit extensions and refinements (see Figure 9–12). The 'innovative' problems also show higher percentage of reflective sequences (69.5%) than the 'routine' problems (46.7%). This is precisely reversed for the 'explicit' extensions and refinements – 30.5% for the 'innovative' class and 53.3% for the 'routine' class.

A similar pattern is observed in the distribution between the methods of assessing partial solutions (Figure 9–13 and Figure 9–14 compared). For the innovative problems, the reflective and inarticulate assessment of acceptability (69.5%) is more typical than the assessment of explicit admissibility (30.5%). On the contrary, the 'routine' tasks feature a more balanced assessment, and they slightly tend to prefer the explicit checks of admissibility (53.2%) to the acceptability (46.8%).

Following from these ratios, the next feature is also not surprising. More 'innovative' problems show almost even distribution between problem refining/extending and re-framing (48.8% and 51.2%, respectively). The same distribution for the 'routine' class is shifted more significantly towards refinements (70.2%), giving less space to the problem re-framing (29.8%).

The prevalence of reflective reasoning for the innovative tasks and the lack of problem re-framing for the routine ones both seem to uphold the attributes typically associated with the respective classes. 'Innovative' problems tend to be vaguer and under-specified, thus requiring more reflection; also the amendments to problem understanding are more significant (including re-framing). On the contrary, 'routine' tasks are better defined; they may need only minor clarifications and refinements rather than significant perspective shifts. 'Routine' tasks are usually not ill-structured problems, therefore explicit criteria for the assessment of their admissibility are typically defined in advance. With a large number of explicit criteria available, there seems to be less need for tacit assessments. More vaguely defined or missing criteria of admissibility in the 'innovative' class seem to force the designer to rely more on the tacit assessments of acceptability.

In addition to these mutual ratios, the 'innovative' problems show scores that are *well above* the average scores calculated for all the experimental sessions. Subsequently, 'routine' problems seem to be *below* the average scores presented in Figure 9–10 and Figure 9–11. This observation seems to comply with the perception that 'innovative' problems are usually outperforming the familiar and 'average' solutions.

9.2.2 Distributions for 'superficial' and more elaborated, complex solutions

The classification of the individual design sessions in respect to producing 'superficial' solutions (often only control algorithms) and 'more elaborated' ones is based upon Table 7–2 in section 7.4.2. The number of sessions exhibiting features of these two classes is almost evenly distributed – 14 'superficial' and 13 'elaborated' solutions/alternatives. The classes are compared using percentage ratios rather than absolute frequencies. Figure 9–15 shows distributions between 'explicit' and 'reflective' problem development in the two classes. For a comparison, the respective ratios are:

- More '*elaborated*' design solutions: ***reflective : explicit = 68.5% : 31.5%***;
- More '*superficial*' design solutions: ***reflective : explicit = 55.0% : 45.0%***

As we can see from Figure 9–15 and the respective percentages, the sessions producing more 'elaborated' (complete) solutions tended to rely more on reflective reasoning (68.5%) rather than direct refinements and extensions (31.5%). On the other hand, the gap between reflective development and explicit refinements/extensions was much less significant for the simpler, more superficial solutions – 55% reflective and 45% explicit. In comparison with the previous section, we can see remarkably similar distributions. Only in this case, more elaborated and complete solutions tended to force the designer to reflect more frequently (68.5%) than the straightforward, rather superficial solutions (55%). However, the gap is narrower than it was between 'routine' and 'innovative' tasks, which suggests that innovation does not necessarily come with more elaborated (complex) solutions.

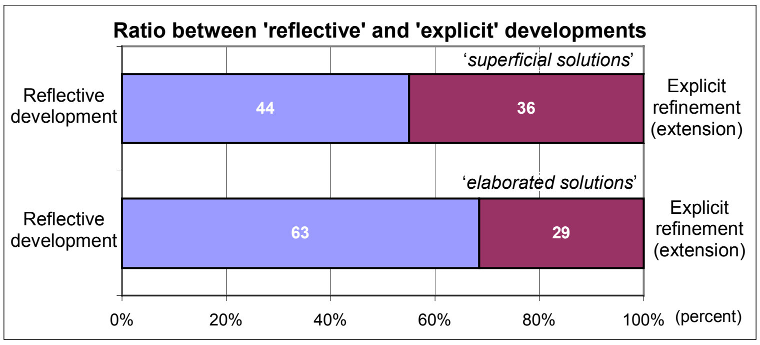


Figure 9–15. Ratio 'reflective development' vs. 'explicit' refinement/extension of design

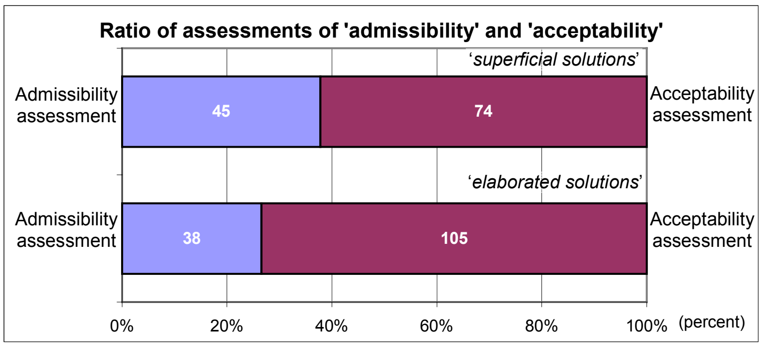


Figure 9–16. Ratio 'admissibility assessments' vs. 'acceptability assessments' of partial solutions

From Figure 9–16 it is possible to conclude that more elaborated, complete solutions may have required more tacit assessments of the acceptability than simpler, superficial solutions. Moreover, solutions that were more complex featured more inarticulate assessments of acceptability (73.4%) than the explicit checks of admissibility (26.6%). The ratio is somewhat similar for the superficial solutions – showing a similar trend. The acceptability checks were more frequent (62.2%) than checks of the explicit admissibility (37.8%). The justification would be similar to the one made earlier – complexity does not necessarily mean the same as 'poor structure' and the need for tacit knowledge.

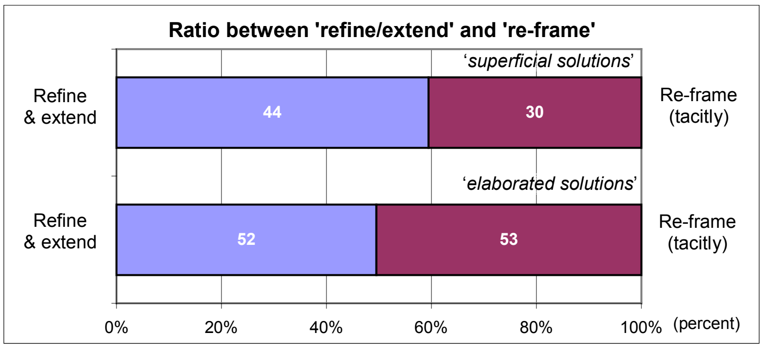


Figure 9–17. Ratio between 'solution refinements/extensions' and 're-framing'

The observation from the previous figures is confirmed also in Figure 9–17. According to the last chart, the 'elaborated' solutions seemed to require more effort from the designer in understanding the particular details he wanted to elaborate. A re-framing sequence in reasoning occurred in 50.5% cases, compared to only 40.5% for the 'superficial' design solutions. The superficial solutions tended to exhibit more refinements and extensions (59.5%) than re-framing (40.5%). This suggests that the designer needed to introduce less conceptual terms and/or requirements for superficially handled problems than for those that yielded more elaborated, complete solutions.

9.2.3 Distributions for conceptually detailed and abstract solutions

A similar pattern in the ratio between reflective development of a design problem and the direct, explicit refinements and extensions is also observable when comparing conceptually detailed and conceptually more abstract solutions. Conceptually detailed solutions often included some details of the implementation, whereas conceptually 'abstract' ones usually corresponded to block diagrams and strategies rather than implementations. Figure 9–18 shows distributions between 'explicit' and 'reflective' problem development in the two classes. For a comparison, the respective ratios are:

- Conceptually '*abstract*' design solutions: ***reflective : explicit = 63.1% : 36.9%***;
- Conceptually '*detailed*' design solutions: ***reflective : explicit = 57.0% : 43.0%***

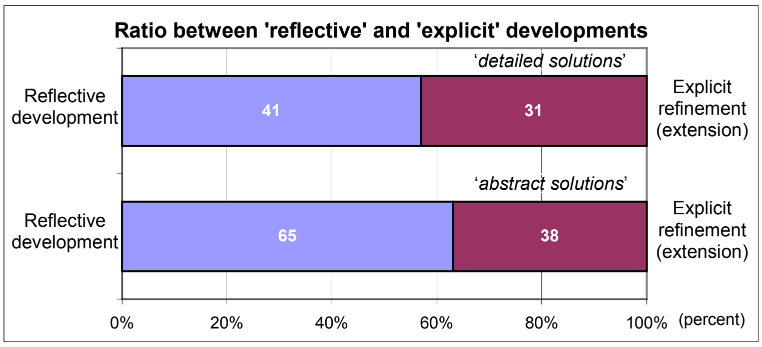


Figure 9–18. Ratio 'reflective development' vs. 'explicit' refinement/extension of design

As we can see from Figure 9–18, there is no significant correlation between the level of conceptual detail and the frequency of reflection. Conceptually 'abstract' solutions tend to provoke more reflection (63.1%) than direct refinement or extension (36.9%). The same holds for the conceptually detailed solutions – reflection observed in 57% cases, direct refinement/extension only in 43%. There is a marginal difference in the number of direct refinements/extensions, which goes slightly in favour of conceptually detailed solutions. This seems to be understandable – if the solutions are conceptually high-level or abstract, it may be difficult to refine or extend them without any exploration or probing.

Nonetheless, the conclusion from this particular visual representation is that there is little correlation between the conceptual abstraction and tacit, reflective reasoning strategies. In certain cases, it was possible to produce an abstract solution from a re-used template – directly refining or extending it. In other sessions, even abstract block diagrams may have required more effort.

9.3 Proposed extensions to RFD model

As we argued throughout this chapter, the empirical results and the aggregated graph confirmed the correctness of the RFD model. Nonetheless, we also identified a few interesting patterns that are not sufficiently accounted for by the original model proposed in chapter 6. As we mentioned, we do not see this as a major shortcoming of our theory or a model. It rather supports the fact that this research is concerned with 'ill structured' design problems, and in the same time, it is 'ill structured' itself. Therefore, the refinements to the first proposal are understandable.

In the next chapter, we look at these refinements in more detail. However, let us conclude this chapter presenting the results and their implications with a brief enumeration of the patterns that need further elaboration:

1. Designers seem to resolve their acceptability assessments either by a *direct* refinement or extension of the current frame, alternatively, they may articulate a new design *frame*.
2. Acceptability assessment need not always lead to a frame amendment, there seems to be another 'operation' of frame *refinement*.
3. The actual frame amendment (or *re-framing*) seems also to exhibit two distinct types. First, it is *re-framing* as a result of having an admissible solution but not accepting it. Let us refer to this type as *problem specification refinement* or *extension via re-framing*.
4. Second, there is re-framing 'proper' or a *conceptual re-framing* that is triggered by the non-existence of an admissible problem solving theory, i.e. there are contradictions in the theory that cannot be overcome by a refinement of problem specification.

These patterns mentioned in the list above require definitions of a few additional predicates that would enable recursion both with and without frame amendment. However, as we shall see in chapter 10, the extended RFD model is only a more complex (or perhaps more complete) version of the original RFD model from chapter 6.

10 EXTENSIONS TO THE RFD MODEL

As we have shown in chapter 9, the RFD model is generally a reasonable approximation of a complex process of design. Nevertheless, the results also seem to show that the original model, as it was first introduced in chapter 6, is not complete and can be further improved. The major extension is needed to account especially for the situations when the explicit problem specification and/or solution may be refined (or extended) irrespective of shifting the conceptual frame. Thus, we were talking in chapter 9 about sequences *7 → 3/6* and clearly distinguished them from simpler, explicit refinements/extensions that can be produced e.g. by deduction. We refer to the specific form of reasoning that aims to refine the current conceptual frame rather than replace it as ***design frame refinement***. This part of the improved model will however, cover both the refinement and the extension of a problem specification/solution. Its chief attribute is that all the improvements occur ***without*** frame amendment.

Another pattern that emerged from the experimental study that was reported in chapter 9, showed that there were at least two different types of ***re-framing***. One type of amendments to the conceptual frame aimed to refine the designer's commitment to a particular problem specification or solution by borrowing the vocabulary of different conceptual frames. The current conceptual frame could be ***extended*** in terms of new explicit problem specification; i.e. new statements could be added into it to specify a new objective that could not be originally expressed. We believe that this kind of ***re-framing*** is less complex than the one that was defined originally in chapter 6. In the aggregated graphs it corresponds to the sequences *7 → 8 → 3/6a*.

Next, we re-visit the original definition of ***re-framing***, and present it as another specific form of frame shifting. In order to distinguish it from the frame extension that was mentioned earlier, we say that the design frame is amended ***conceptually***. In other words, an entirely new conceptual apparatus is needed in order to construct a sound design frame. Unlike in the previous case, this frame shift is focused directly on the conceptual primitives, terms and objects for the solution construction. Since the objects are changed in this operation, the applicable domain theory may change as well!

Let us define the newly emerged patterns similarly as we defined the original RFD model. We do not repeat the definitions of terms and simple relations because these were upheld in the experiments and remain unchanged. Section 10.1 defines a few additional predicates that are needed to implement the above-mentioned improvements. Section 10.2 presents the examples of the newly defined *schemes of reasoning*. For a brief summary of the extended RFD, see also (Dzbor and Zdrahal 2002).

10.1 Recursive model of framing in design (2.)

In the following paragraphs, we propose a refinement or extension of the recursive model of conceptual framing in design. This model extends and supersedes the claims made in section 6.2. The building blocks of the extended RFD model are the same entities as identified in chapter 6:

a) a problem solving *theory* $\mathfrak{T}^*$,
b) an explicit problem *specification* $\boldsymbol{S}$,
c) a problem solving *model* $\boldsymbol{T}$, and
d) a conceptual design *frame* $\boldsymbol{\Phi}$

The extended RFD model is again defined as a sequence of interactions between two predicates '*satisfies*' and '*specifies*', as they were defined in chapter 6. To refresh the memory, the '*specifies*' predicate explicitly articulates the problem specification, and '*satisfies*' attempts to solve the explicitly specified problem. These two 'actions' have a potential to generate and change the explicit design knowledge. They are complemented by the third predicate '*acceptable*' that acts as a switch deciding on whether design process continues or whether it may be successfully concluded by an articulation of a (acceptable) design solution.

Let us define the new auxiliary predicates that are used in the design model in order to decide about the further steps. Similarly as in chapter 6, they are useful for the interpretation of the RFD model. Then, the model can be read more intuitively as a sequence of decisions followed by actions. According to the experimental findings, the simplest modification to the current interpretation of the design problem is to explicate a statement that is believed to refine the current specification. If such a statement can be articulated using the *current design frame* $\boldsymbol{\Phi}$ and the vocabulary of the currently chosen conceptualisation $\mathcal{T}$ we may continue with a decision as shown in Figure 10–1, level 3. The predicate for *frame refinement* is defined by Eq. 10–1.

$$\textit{can-refine-frame}_{\Phi}\ (S) \Leftrightarrow \exists s \in \mathcal{S}^{*}, S \subseteq \mathcal{S}^{*}: S' = S \cup \{s\} \wedge \textit{specifies}_{\Phi}\ (S', \mathcal{DP}) \qquad \textbf{Eq. 10–1}$$

As we mentioned at the beginning of this chapter, this predicate does not modify the actual conceptual frame. It only shifts the designer's attention to a statement that already existed in the problem interpretation; now there is an *explicit commitment* to that statement. The fact that the design frame remains untouched is emphasised by regarding $\boldsymbol{\Phi}$ as a parameter circumscribing the space for the possible refinements. The predicate in Eq. 10–1 corresponds to transitions *reflect – refine* (*extend*).

Another auxiliary definition regards a more complex decision that attempts to re-interpret the explicit problem specification in a new frame. Thus, there is still a refinement or an extension to the current problem specification; however, the new statements are introduced from a new design frame rather than the existing one. The difference from the predicate in Eq. 10–1 is in the fact that this is a sequence of *reflect – re-frame – refine* or *reflect – re-frame – extend*. None of these more complex extensions can be done without changing the conceptual frame; i.e. the currently used conceptual categories from set $\mathcal{S}$. Eq. 10–2 contains a formal definition of the decision to resolve the 'tacit' non-acceptance by committing to the terms borrowed from a new design frame (new view on the problem).

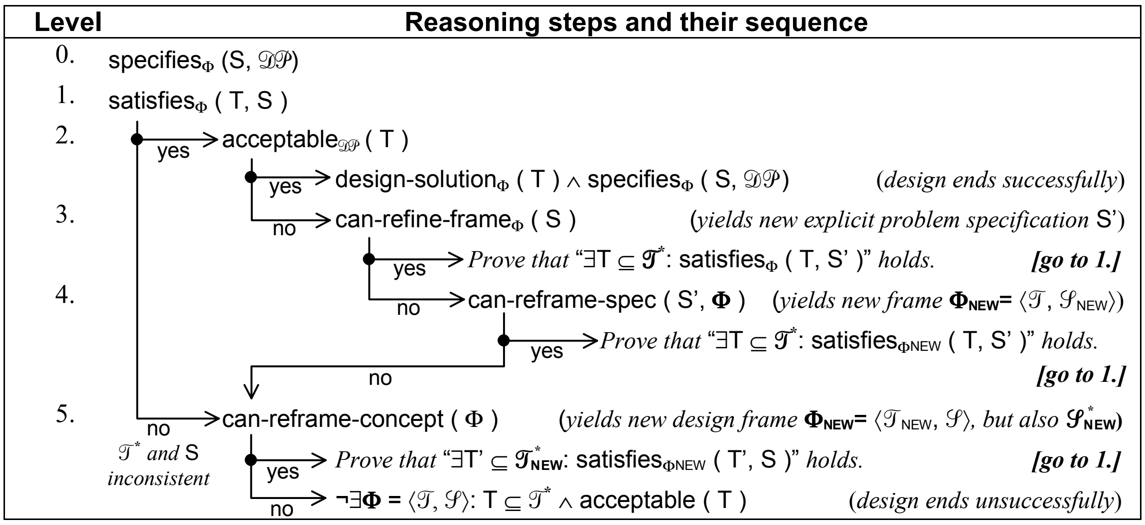
$$\textit{can-reframe-spec}\ (S, \Phi) \Leftrightarrow \exists \boldsymbol{\Phi}_{\mathbf{NEW}} = \langle \mathcal{T}, \mathcal{S}_{\text{NEW}} \rangle: S' \subseteq \mathcal{S}^{*}_{\text{NEW}} \wedge \textit{specifies}_{\Phi\text{NEW}}\ (S', \mathcal{DP}) \qquad \textbf{Eq. 10–2}$$

And finally, a decision that can be most disrupting to the current design frame and the interpretation of the problem is defined in Eq. 10–3. As we see, what is changed in this particular situation, are the conceptual means/primitives of the current design frame for solving the problem. The problem conceptualisation $\mathcal{T}$ that formed the basis of the conceptual frame $\boldsymbol{\Phi}$ and the current approach is given up and replaced. A new frame is articulated so that the current explicit specification of the problem (i.e. $\boldsymbol{S \subseteq \mathcal{S}^{*}}$) becomes a consistent and admissible specification of the design problem (albeit in a modified design frame, modified context, modified conceptual vocabulary).

$$\begin{aligned} &\textit{can-reframe-concept}\ (\Phi) \Leftrightarrow \\ &\quad \Leftrightarrow \exists \boldsymbol{\Phi}_{\mathbf{NEW}} = \langle \mathcal{T}_{\text{NEW}}, \mathcal{S} \rangle: T' \subseteq \mathcal{T}^{*}_{\text{NEW}} \wedge S \subseteq \mathcal{S}^{*}_{\text{NEW}} \wedge \textit{specifies}_{\Phi\text{NEW}}\ (S, \mathcal{DP}) \end{aligned} \qquad \textbf{Eq. 10–3}$$

Note a very interesting pattern in the definition of the last method of re-framing in Eq. 10–3. Unlike any previous definitions, this one is amending the set $\mathcal{T}$ (problem conceptualisation). The newly articulated conceptual foundation $\mathcal{T}_{\text{NEW}}$ is inevitably leading to a re-consideration of a generic domain theory ***DT***, and eventually a definition of a new problem solving theory $\mathcal{T}^*_{\text{NEW}}$. However, because of the change in the conceptual vocabulary for problem solving, the set $\mathcal{S}^*$ must be re-considered, too. With the new objects, different specifications can be made – and this is the reason of having the subset relation $\boldsymbol{S} \subseteq \mathcal{S}^*_{\textbf{NEW}}$ in the definition. Nevertheless, the designer still wants to pursue the same goals ***S***, as in the previous frame; there is no reason to change both sets 'in parallel'.

Each of these three auxiliary predicates represents a decision that can be made in a certain specific phase of tackling the design problem $\mathfrak{DP}$. As in chapter 6, all these decisions share one important feature. They are decisions and at the same time, they are generative operations that improve the designer's understanding of the design problem. All the decision predicates deal with the relation '*specifies*' and aim to find a new problem specification or a new frame in which the original specification could be interpreted. The decisions defined above are followed by the implementations proving their correctness. These are represented by a relation '*satisfies*' that manipulates the problem solving theories, models and solutions. Thus, the mutual interaction of two knowledge sources remains.



Level
Reasoning steps and their sequence
0. specifies$_\Phi$ (S, $\mathfrak{DP}$)
1. satisfies$_\Phi$ (T, S)
2. yes → acceptable$_{\mathfrak{DP}}$ (T)
yes → design-solution$_\Phi$ (T) ∧ specifies$_\Phi$ (S, $\mathfrak{DP}$) (*design ends successfully*)
3. no → can-refine-frame$_\Phi$ (S) (*yields new explicit problem specification* S')
yes → *Prove that* "$\exists$T $\subseteq \mathcal{T}^*$: satisfies$_\Phi$ (T, S')" *holds.* **[go to 1.]**
4. no → can-reframe-spec (S', **Φ**) (*yields new frame* **Φ**$_{\textbf{NEW}}$= ⟨$\mathcal{T}$, $\mathcal{S}_{\text{NEW}}$⟩)
yes → *Prove that* "$\exists$T $\subseteq \mathcal{T}^*$: satisfies$_{\Phi\text{NEW}}$ (T, S')" *holds.* **[go to 1.]**
no
5. no → can-reframe-concept (Φ) (*yields new design frame* **Φ**$_{\textbf{NEW}}$= ⟨$\mathcal{T}_{\text{NEW}}$, $\mathcal{S}$⟩, *but also* $\mathcal{S}^*_{\textbf{NEW}}$)
$\mathcal{T}^*$ *and* S *inconsistent*
yes → *Prove that* "$\exists$T' $\subseteq \mathcal{T}^*_{\textbf{NEW}}$: satisfies$_{\Phi\text{NEW}}$ (T', S)" *holds.* **[go to 1.]**
no → ¬∃**Φ** = ⟨$\mathcal{T}$, $\mathcal{S}$⟩: T $\subseteq \mathcal{T}^*$ ∧ acceptable (T) (*design ends unsuccessfully*)

Figure 10–1. Recursive model of framing in design (extended version)

The breakdown depicted in Figure 10–1 follows similar levels and obeys exactly the same simple rules that were first time introduced in chapter 6. They are repeated below:

1) When using terms 'requirements' and 'constraints', we always mean hard, strict demands that *must not be relaxed* – see e.g. (Fox 1994) for the constraint relaxation techniques.
2) We *do not consider* the design problems, where a problem specification can be simplified or otherwise relaxed; these issues are covered by other research, e.g. (Zweben and Fox 1994).
3) However, problem specification may be changed by a *monotonic extension* of an initial set; i.e. new, refining statements about a specification can be made under specific circumstances.
4) The monotonic extension of a problem specification corresponds to a designer's attempt to 'fine tune' a problem solving model, to *narrow down* the number of derivable alternatives.

5) Problem specification can be '*fine-tuned*', only if an admissible problem-solving model exists for the current conceptual frame, and it needs to be *refined*.
6) Sentence in the form '*Prove that (...) holds…*' represent <u>*a recursive step*</u> starting always at the level 1. of the recursive model.
7) The recursive step represents a designer's attempt to address given design problem by some modification to the knowledge sources available for the design. It can be also understood as an order to an agent to '*evaluate*' a particular predicate '*with the new arguments provided*'.

Let us explain the proposed recursive model by identifying the schemas of reasoning in design that are explainable from the extended RFD model, and how they correspond to the suggestions made at the beginning of this chapter. These are conceptual schemas; we propose them as abstract models of different patterns of reasoning that were observed empirically. They are intended to depict the mutual dependence and sequence of different knowledge-level reasoning steps, as well as the interplay of explicit reasoning strategies with those using designer's inarticulate knowledge. Schema 2 features only a monotonic refinement of the design frame, whereas schemas 3 and 4 feature two forms of the non-monotonic introduction of new knowledge about design problem. Predicate chains in the schema definition (arrows) refer to the paths between the respective actions/decisions in Figure 10–1.

<u>Schema 1</u>

$$\begin{tikzcd} \text{satisfies}_{\Phi}(T,S) \arrow[r, "\text{yes}"'] & \text{acceptable}_{\mathscr{DP}}(T) \arrow[r, "\text{yes}"'] & \textbf{design-solution}_{\boldsymbol{\Phi}}\textbf{(T)} \end{tikzcd}$$

This schema remains the most trivial in the design reasoning from the perspective of interplay between articulate and inarticulate reasoning. It was described earlier, in chapter 6, and we are not going to repeat it here. Nonetheless, we should associate it with the aggregate results of the experiments. From that point of view, this schema usually features transitions *1a,b* → *5* → *4* → *6b* (i.e. satisfy the explicit specification, check that the solution is admissible and refine it if needed). In this case, no 'tacit' knowledge is needed, and the problem can be solved using a formal algorithm. In our opinion, the essence of this schema can be expressed as follows: "*I know how to define the problem, and I have all necessary data/objects available.*"

<u>Schema 2</u>

$$\begin{tikzcd} \text{satisfies}_{\Phi}(T,S) \arrow[r, "\text{yes}"'] & \text{acceptable}_{\mathscr{DP}}(T) \arrow[r, "\text{no}"'] & \text{can-refine-frame}_{\Phi}(S) \arrow[r, "\text{yes}"'] & \text{satisfies}_{\Phi}(T,S') \\ & & \textit{yields refined or extended problem specification } S' & \textit{iterate with new explicit specification} \end{tikzcd}$$

This is a situation when, for example, there are multiple solutions available, and all of them are admissible in the given design problem. Although being 'equally suitable', the designer seems to distinguish between them on the level of their *acceptability* as design solutions. As we defined in chapter 6, an admissible solution must be also in accordance with the designer's implicit and tacit expectations from the product of the design. Since these expectations are not explicitly articulated, there are no criteria that can be used during the admissibility checks. Nonetheless, the designer can complement the admissibility checks by this tacit assessment of acceptability.

If there are multiple alternative solutions, for example, the designer does not accept such an ambiguous situation, and looks for a discriminatory condition (requirement, constraint) to choose from the alternatives, and sort ‘more acceptable’ from the ‘unacceptable’ ones. The discriminatory condition is taken from the same ontological frame, and its role is to refine the current understanding of the design problem, especially its explicit specification. No external factors are needed to formulate it. We think that this schema corresponds to a situation when a designer exclaims: “*I have forgotten to say specifically that (... I want the values of parameter X be from range ⟨Y, Z⟩.)*” In other words, the designer has a conceptual apparatus available in the current frame to express the missing statement.

In the terminology of the aggregated results shown earlier in chapter 9, this schema is addressing the sequence of transitions *(5 →) 7 → 3a/6a*; i.e. a sound solution exists, but is not acceptable, and the problem specification needs to be refined/extended. After a clarification has been made in the specification space, the designer can in parallel improve the product. This refinement of the solution corresponds to a sequence *(5 →) 7 → 3a/6b*. Nonetheless, any changes occur inside a single, sound conceptual frame, no new concepts are required.

<u>Schema 3</u>

$$\text{satisfies}_{\Phi}(T, S) \xrightarrow[\text{yes}]{} \text{acceptable}_{\mathscr{DP}}(T) \xrightarrow[\text{no}]{} \text{can-refine-frame}_{\Phi}(S) \xrightarrow[\text{no}]{}$$

$$\xrightarrow[\text{no}]{} \underset{\substack{\textit{yields new frame}\\ \Phi_{\text{NEW}} = \langle \mathscr{T}, \mathscr{S}_{\text{NEW}} \rangle}}{\text{can-reframe-spec}(S, \Phi)} \xrightarrow[\text{yes}]{} \underset{\substack{\textit{iterate w/new frame}\\ \textit{and new specification}}}{\text{satisfies}_{\Phi\text{NEW}}(T, S')}$$

Similarly as in the second schema, this is a situation when there is at least one sound solution available but the designer does not accept it. The reasons are also similar as earlier. There seems to be a difference between an explicit frame $\Phi = \langle \mathscr{T}, \mathscr{S} \rangle$ that was used for the problem specification and the solution construction, and the designer’s tacit expectations. A logically admissible solution is subjected to an assessment of its acceptability, and as a result a need for ‘synchronisation’ of the explicit frame and tacit expectations may emerge. Unlike the schema number 2, it is not possible to synchronise the tacit expectations using the conceptual vocabulary of the current frame. In other words, the specification set $\mathscr{S}$ does not contain a vocabulary for expressing the new expectations explicitly.

Moreover, the designer may not be entirely sure how to restore the acceptability, and tries to borrow concepts from a different frame to express his or her expectations in some explicit form. Such a frame synchronisation is equivalent to a formulation of a new conceptual frame ($\exists \boldsymbol{\Phi}_{\text{NEW}} = \langle \mathscr{T}, \mathscr{S}_{\text{NEW}} \rangle$). A new design frame $\boldsymbol{\Phi}_{\text{NEW}}$ contains a different vocabulary, which may help to reveal what might have been missing in the original explicit specification of the problem. Thus, re-framing amends first the explicit problem specification primitives ($\boldsymbol{S'} \subseteq \mathscr{S}^{*}_{\text{NEW}}$), and subsequently, may have an impact also on the process of satisfying the particular, amended specification.

This case is clearly distinct from schema 2 in the fact that a new requirement can be formulated only after shifting of the design frame. Without such a shift the requirement remains ‘hidden’; i.e. in frame $\boldsymbol{\Phi}$ there are no conceptual means to express it explicitly. We believe that the following sentence expresses this reasoning. *“I cannot solve the problem specified in this way, something is definitely missing. What if I looked from a different angle and assumed XY?*” The vocabulary for expressing the missing statement may not be available, but the designer knows how to acquire a new vocabulary.

In the aggregated graphs, this schema shows the transitions *(5 $\rightarrow$) 7 $\rightarrow$ 8 $\rightarrow$3a, 6a*; i.e. between the acceptability check and refinement is inserted a frame modification. Provided it is accomplished successfully, i.e. a new frame is found, the designer has to confirm the correctness of the tacit need for a new conceptual frame by implementing the conjectured exploratory probe. A new solution needs to be constructed that would *satisfy* the amended specification ***S'***. In the aggregated graphs, this would be typically depicted as a transition *7 $\rightarrow$ 8 $\rightarrow$ 6b*.

Schema 4

$$\begin{tikzcd}
\text{satisfies}_{\Phi}(T, S) \arrow[r, "\text{no}"'] & \text{can-reframe-concept} (\Phi) \arrow[r, "\text{yes}"'] & \text{satisfies}_{\Phi NEW}(T', S)
\end{tikzcd}$$

yields new design frame   *iterate w/new frame & new problem*

$\Phi_{NEW} = \langle \mathcal{T}_{NEW}, \mathcal{S} \rangle$ *conceptualisation* $\mathcal{T}_{NEW}$ *(also* $\mathcal{S}^{*}_{NEW}$*!)*

Unlike the previous two schemas that could occur after an acceptability check, this situation may occur as a result of both, the admissibility or acceptability check. The reason is that the original frame $\boldsymbol{\Phi}$ contained mutually incompatible and inconsistent sets. The mutual inconsistency manifested itself as an explicit contradiction between the requirements or constraints in $\boldsymbol{S} \subseteq \mathcal{S}^{*}$. However, the source of the contradiction was found to be in the conceptual foundation of the problem interpretation – set $\mathcal{T}$. Using that particular conceptualisation and a chosen domain theory ***DT*** instantiated by the conceptual set $\mathcal{T}$, caused an emergence of a 'deep contradiction' that could not be resolved by any refinements to the problem specification whether with or without re-framing.

In other words, if the previous schemas addressed the issue of not having an acceptable problem specification, this schema is looking for a new frame in which the problem could be *satisfied*. The conceptual vocabulary of frame $\boldsymbol{\Phi}$ for the solution construction is incomplete and must be refined, extended or otherwise modified. To achieve this goal, a new design frame $\boldsymbol{\Phi}_{\mathbf{NEW}}$ is articulated. It contains new or different conceptual terms than the old one, and consequently, new, different problem solving theories $\mathcal{T}^{*}_{\mathbf{NEW}}$ can be formulated. The novelty of the problem solving theories is emphasised by the opportunity to choose a different generic domain theory $\boldsymbol{DT}_{\mathbf{NEW}}$ that fits a new frame better than any previously used domain theories did.

New conceptual interpretation of a design problem $\mathfrak{DP}$ is clearly a non-monotonic and ill-defined operation. At the same time, this situation is rather well structured; i.e. the designer has an explicit knowledge of a contradiction. There is a (typically constraining) condition available that can be in focus of a designer's attention whilst searching for a new frame – something that can be assessed for compliance with any new conceptualisation and a conceptual design frame. The following statement expresses this situation: "*I know what is wrong, and I have an idea about how to fix it.*" Similarly as in schema 3, the designer does not have the expressive means in the current frame to formulate those missing concepts; he seeks a new conceptualisation.

In the aggregated graphs, this schema corresponds to a direct commitment to the following transition *(7 $\rightarrow$) 8 $\rightarrow$ 3b, 6b*; i.e. after identifying an explicit conflict, define a new conceptual frame and propose new components to form an admissible solution. The reasoning chain may start with an acceptability check; this only emphasises the fact that the discovery of a contradiction may start as an inarticulate feeling that is step by step narrowed to an explicit conflict.

10.2 New design schemas exemplified

Having proposed a formal definition of conceptual design frames thus forming the foundations of the recursive model of framing in design, we illustrate the claims with examples. The identified schemas from the previous section are expanded in the subsequent paragraphs, and 'instantiated' using the empirical findings we acquired during our research. The accounts given below are abbreviated and focus on the relevant phase that illustrates a particular schema. Additional details and annotations of the experimental design sessions can be found elsewhere in this document; see chapter 8.

Before discussing the details of the individual schemas, we would like to note that any proposed schema might characterise an entire design task. However, in most cases, several of the schemas appear during the process of designing. This must not be viewed as a shortcoming of the theory describing the use of conceptual frames in the design! In our opinion, the appearance of multiple reasoning schemas in a single design case underlines the ill-structured nature of design problems. Simultaneously, this feature supports our supposition that designers apply a variety of reasoning techniques, when they are tackling the ill-structured design problems. The point of 'multiple paths to a design solution' was mentioned in several articles on the ongoing research (Dzbor 1999; Dzbor and Zdrahal 2001b; Dzbor and Zdrahal 2002), as well as in chapters 4 and 5.

Rather than labelling a design task with a unique schema, and following a procedure that may be typically associated with such a schema, design is by nature exploratory. Several paths may lead to an acceptable solution – some yielding solutions that are more acceptable than others may, but it cannot be said in advance, which one will guarantee a results. We shall argue that this is the basis of the 'famous' ill structure of Herbert Simon (1973). In addition, we argue that the application of several different schemas is also in accordance with the theory of reflection in action authored by Donald Schön (1983), and mentioned several times in the review (chapters 3 to 5).

From a theoretical point of view, we confirmed the intuitively expected coincidence of certain schemas in a particular design task (see chapter 9). For instance, the less structured the design problem is, the higher the frequency of observing schemas number 2 or 3, rather than number 1. Inventive problems may be perhaps exhibiting schema 4 more frequently than other schemas. On the other hand, more routine and trivial tasks typically comply with schema 1 and possibly 2. The presented analyses confirmed such intuitive expectations rigorously and over a larger number of design sessions.

In the following sub-sections, we present the exemplar design scenarios in terms introduced earlier; i.e. we state what was the original conceptualisation, domain theory, problem solving theory, problem solving model, design specification, constraining conditions, etc.

10.2.1 Schema 2 – design refined by implicit specification

This reasoning schema is commonly observable in design tasks. We believe that a reason for this is that the designers work with many implicit expectations, in addition to the explicit specification of a design problem. These expectations are not expressed formally or in the same language, as the explicit requirements or constraints. For instance, Nakakoji, Sumner *et al.* (1994), and Nakakoji, Yamamoto *et al.* (1998) refer to a similar phenomenon from a cognitive perspective as 'design intentions', and argue that many intentions remain tacit. This is a worthwhile assertion but we believe it includes (mainly) the implicit expectations (Dienes and Fahey 1998), as well as tacit ones (Cook and Brown 1999).

We argue that there is a significant amount of confusion in the research community in using terms 'tacit', 'implicit', and perhaps 'not explicit'. As Cook and Brown (1999) correctly argue, tacitness and explicitness are two different types, *two dimensions* of knowledge; one can be used to acquire another one but it is not possible to turn tacit into explicit. Thus, in line with such an epistemological perspective, we claim that the process of 'surfacing tacit intentions and mapping them into explicit representations' as Nakakoji, Sumner *et al.* (1994) see it, is mainly about an explicit formulation of implicit, 'hidden' intentions! Designers often take certain things for granted, and they do not explicate them until necessary. Their tacit knowledge, experience may surely help to formulate the 'forgotten', important expectations, but the result is only an explicit formulation of a statement in the language of a particular conceptualisation. In other words, such a statement must be (and usually is) *expressible*, but has simply not been *expressed explicitly*!

Consider, for instance, the following design situation that we observed as a part of our experimental work. In section 8.4, a designer was asked to design a strategy for smoothing raw paper of variable thickness. The problem was initially framed so that the operation of smoothing was associated with the application of pressure, and consequently a pair of rolling drums was suggested as a problem solving model. This approach satisfied the desired requirements on the smooth paper and the even thickness. The paper was smoothed as intended, however in some cases, there was an imminent danger of its tearing, which was not desirable. Despite the fact that the prevention of paper tearing was not explicitly articulated in the initial problem specification, it seemed reasonable to add this as another *explicit* requirement. This explicit commitment reminded the designer that he had to be aware of this situation in the future design decisions. In a discussion, the designer justified this extension by pointing to the fact that such a requirement is 'so obvious that it is rarely emphasised'.

In this articulation of a previously implicit requirement, the current conceptual objects remained unchanged, but the explicit problem specification was refined. This monotonic extension occurred within the original frame that worked with terms as 'rolling', 'pressure', 'drums', etc. However, this simple explication of an implicit requirement had strong implications on the otherwise non-monotonic problem solving theory. Because of such refinement, the designer needed to find a new problem solving model to comply with the refined problem specification. In the particular design case, this extension led to the introduction of new conceptual terms – pre- and post-processing units to the assembly in form of a moisturiser and a dryer.

These new conceptual units softened the paper before rolling, so that lower pressure was required, and the danger of tearing was reduced. However, the actual addition of the new components is already another issue that is a *consequence* of schema 2 rather than its cause. Thus, in this particular case, schema 2 amended the problem specification (transition *7* → *6a*), and was followed by an attempt to satisfy the new specification (i.e. transition *7* → *3b, 6b* → *5*). Nonetheless, the whole episode worked within a single design frame. Thus, we may illustrate the episode as follows:

a) *Explicit problem specification (requirements)*:

- 'Make paper smooth, debris-free';

- 'Ensure paper has even thickness as specified'

b) Explicit problem specification (constraints):

- 'Minimal/maximal pressure for a non-destructive deformation of paper is given by a chart in catalogue XY'

c) Domain theory:

- 'Theory of mechanics, incl. equations for friction calculation, pressure distribution, material deformation/shaping, etc.';
- 'If a maximal strain of the material is exceeded, this may lead to the permanent (non-elastic) destruction of the material'

d) Conceptual design structures:

- 'Rolling drum pair' described by the dimensions of the drums, gap between them, pressure exerted on the material in the gap, etc.' (see also Figure 10–2 for illustration)

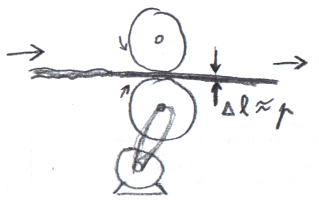


Figure 10–2. Paper-smoothing assembly *before* the implicit refinement

e) Reason of unacceptability of solution *(see Figure 10–2):*

- 'If we worked with a thinner paper, we could easily exceed the maximal strain threshold'

f) Newly articulated requirements *(previously implicit)*:

- 'We must avoid the ambiguity, and strictly ensure that paper is not damaged';
- 'Thus, it seems to be better to reduce the pressure of the drums significantly below the maximal threshold';
- 'However, we somehow need to compensate the reduced pressure.'

g) Newly articulated design solution:

- 'Lack of pressure can be compensated by loosening the bonds in the material (similarly as with steam ironing or metal sheet rolling)';
- 'A solution taking these issues into account may look like Figure 10–3.'

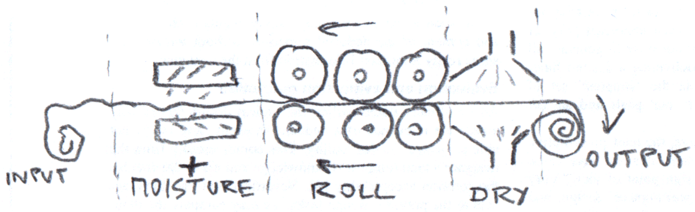


Figure 10–3. Paper-smoothing assembly *after* the implicit refinement

In the newly articulated requirement, there is no reference to a new concept or a new structural object. Not even the properties mentioned are new to the current conceptual frame. What is new however, is an explicit commitment to a particular feature of the paper that was known in the current conceptual frame. Paper continuity and ‘no structural damage’ were present in the original conceptual frame, now the designer explicitly articulated, what are the desired values for these properties. Therefore, we see schema 2 as a refinement of the problem specification rather than any amendment to the conceptual design frame.

10.2.2 Schema 3 – refinement of problem specification via re-framing

On a superficial glance, this schema is very similar to number 2 discussed in section 10.2.1. The main difference between these two schemas modifying the problem specification is in the design frame in which a missing statement is formulated. Whereas in schema 2, the formulation can be achieved in the language and concepts currently available in the chosen conceptual frame, this is not possible in schema 3. In other words, schema 2 is more about an explicit commitment to an implicitly present statement, i.e. a problem refinement. Schema 3 requires more ‘effort’ in order to discover what is actually missing. In our recursive model, we propose to express this ‘extra effort’ by an operation of *re-framing*, and consequently a formulation of new explicit problem specification of design problem $\mathfrak{DP}$.

However, this is a specific form of frame amendment, therefore we called it *refinement via re-framing*. The actual conceptualisation of the problem, ‘a vocabulary’ (i.e. entity $\mathcal{T}$ from section 6.1) does not change. What is changed, is the explicit problem specification. As with schema 2, this is also a kind of synchronisation of an explicit frame with tacit expectations and intentions. The refinements modelled by schema 3 were not only forgotten in the original specification, but there was no vocabulary to express them in the first place. This new vocabulary must be brought in from a different frame so that the designer may modify the interpretation of the design problem. In practice, new statements with new semantic meanings may be articulated for the ‘re-framed and refined’ problem specification.

As an illustration, consider the following episode from the design of an active shock absorber (section 8.2). Designer framed the problem with the help of such concepts as ‘spring’ and ‘rigidity’. Very early in his design, he discovered that the rigidity of ‘a spring’ must be modifiable to reflect the uneven road. Rigidity of a shock absorber would depend on the toughness of a spring, and in order to modify the rigidity, this parameter would need to be changed. However, this was not so easy for a common spring, it could only be achieved by changing its material or structural properties (such as diameter or length). Therefore, he proposed a functional alternative – a pneumatic absorber, and devised a set of rules for amending the pressure inside the piston (see also Figure 10–4).

It seemed to be a straightforward design from this point onward. However, when reflecting on the proposed solution and the respective controller, the designer became aware of an unexpected behaviour. According to his control strategy, if a disturbance was recorded on the chassis, this was interpreted as a car entering a ‘rough road’. The control action immediately softened the absorber; i.e. decreased the pressure inside it. The stepwise reduction of pressure would stop as soon as the oscillations ceased to be measurable. Then, the designer recalled the original frame from which he borrowed the pneumatic piston, and realised that his control strategy had serious gaps.

Namely, he became aware of ‘one-way-only’ control; i.e. he was only able to decrease the pressure when an uneven road was hit. However, this was not a correct interpretation of the term ‘active suspension' because it could not perform a *reversed effect*. Thus, the control needed to be 'more active' in terms of reacting to the road surface. To that extent, he articulated a need for a reversible pressure control that included a simple 'prediction' of the future states of the road. Thus, a concept of optimal shock absorption emerged together with the basic assumptions enabling the prediction.

Eventually, new terms extended and refined the explicit problem specification to reflect the designer's awareness of the inadequate understanding of the original problem. However, these terms, requirements and assumptions were not expressible in the original frame. They required the designer to allow for 'historical' data that can be used for the prediction. Furthermore, the designer had to introduce *hypotheses* and *assumptions* for the prediction, which were neither requirements nor constraints. In any case, they helped to articulate a more acceptable and more complete specification of the problem of designing an *active* shock absorber. They refined designer's understanding of the problem by referring to new relevant concepts from similar frames (cases).

a) *Explicit problem specification (requirements)*:

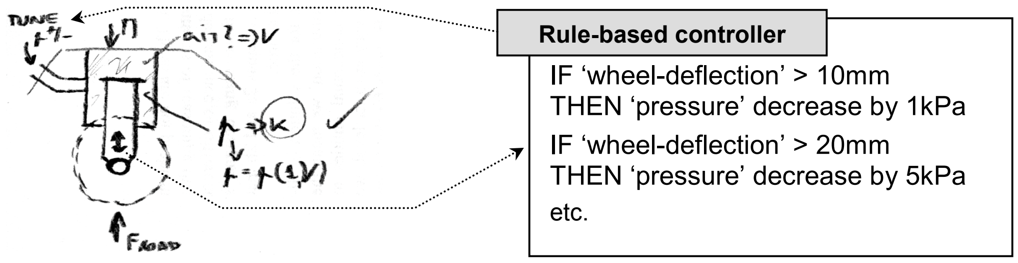
- 'Adjust pressure of an absorber in response to the uneven road surface';
- 'Pressure adjustment must be easy to realise and fast in response'

b) *Domain theory*:

- 'Mechanics, pneumatic systems, pressure equations, IF-THEN productive rules, etc.'

c) *Conceptual design structures*:

- 'A simple shock absorber consisting of a pneumatic piston, measuring and control unit as depicted in Figure 10–4 (on the left)';
- 'IF the wheel changes its position more than X mm, THEN decrease the pressure of absorber to eliminate the disturbance as quickly as possible';
- 'IF the chassis oscillation is eliminated quickly, THEN passenger has higher comfort'



Rule-based controller

IF 'wheel-deflection' > 10mm
THEN 'pressure' decrease by 1kPa
IF 'wheel-deflection' > 20mm
THEN 'pressure' decrease by 5kPa
etc.

Figure 10–4. A simple shock absorber with a 'rule-based' control strategy

d) *Missing features typically occurring in similar cases*:

- 'The adjustment must be controlled **reversibly**, i.e. able to increase and decrease pressure';

⇒ 'The activity must be more visible and more intelligent'

e) *Tentative proposals*:

- 'If we had some kind of predictive mechanism that would be able of adaptation, that might deliver more active and intelligent control';
- 'What about a new rule along the lines: IF nothing is happening with the car, THEN try increasing the pressure → a stiffer absorber';

- 'Such heuristic rules may bring in better adaptation and a kind of probing how far the controller can go in adjusting the pressure → optimality of suspension'

In the brief extract from the episode, we may see the tentative question 'What if?' referring to the specification of the problem but using the terminology of the familiar cases/frames tackled in the past. Similarly as schema number 2 in section 10.2.1, this schema is modelling a refinement or an extension of the explicit problem specification. However, this is a refinement that draws on the tacit discovery of the potentially useful features in the past problems and their re-use in the current problem understanding. This reference to conceptually new or different frames is the feature distinguishing schemas 2 and 3. Unlike schema 4, this schema amends the specification rather than generates a new solution to the given problem. Anyway, the illustrative scenario corresponds to a sequence of reasoning steps *7 → 8 → 6a* but also *7 → 8 → 3a*. It is refining the notion of 'activity' and simultaneously, extending the specification with the concepts 'prediction' and 'adaptation'.

10.2.3 Schema 4 – re-framing of a contradictory theory

This schema is typically invoked when the current perspective is unable to propose any suitable problem solving models that would be consistent with the required and specified features. The reason may be a fact that some of the specified features are mutually contradictory, and cannot co-exist in a form that is dictated by the current problem solving theory $\mathcal{T}^*$. The reader remembers that a problem solving theory was an instantiation of a generic domain theory ***DT*** using the chosen conceptualisation $\mathcal{T}$. This is not to say that the domain theory itself must have been flawed, but clearly, its interpretation in terms of the chosen conceptualisation led to contradictions.

A similar argument addressing the inventive technological designs and their origins in an explicit contradiction was already argued in section 4.1.4. A contradiction goes usually deeper than the plain absence of a desired feature in the current state (Coyne, Rosenman *et al.* 1990). The innovative character of a solution appears in response to challenging these deeply rooted familiar interpretations rather than tackling the absence of a desired superficial feature. Thus, problems must be seen in a conceptually innovative manner, in order to have innovative or inventive solutions for them!

Let us illustrate this scenario on an episode from design of a paper-smoothing plant (section 8.4). At a certain stage, the designer identified two contradictory requirements that could not co-exist and be satisfied in a single design solution. On one hand, he demanded quality of paper smoothing, as expressed by an even thickness and smooth surface. In order to achieve better quality he added more pairs of rolling drums, and extended the plant to form a sequence of drums. This 'extensive' approach to satisfying a particular requirement was logically correct. Nevertheless, he could not continue adding drums infinitely. On the other hand, the design was bound by an implicit constraint on a maximum length of the plant that was justified by the difficulty of maintenance and control.

It was clear that improving the quality by enlarging the assembly only worsens the maintainability and control-related constraints. The increase in pressure had also its limits, because the higher the pressure, the greater was the danger of tearing the paper. A torn, damaged paper was (obviously) not admissible at this stage (see also section 10.2.1). The designer tried to 'squeeze' the rolling drums closer together so that he could gain some 'reserve' in the dimensions; however, this was of little use in simplifying the control of the plant. Thus, the designer found himself in a ridiculous, never-ending circle of mutually contradicting requirements.

He resolved the threatening deadlock rather nicely. Instead of squeezing or expanding the layout of the rolling assembly in 'one dimension' (i.e. linearly laid-out pairs of drums), he amended his conceptual frame. This amendment occurred, when he became aware of and articulated the concept '*two-dimensional layout*' and '*two-dimensional squeezing*'. When it was impossible to go beyond the constraining limits in one dimension, he brought in another dimension, and tried the same in two dimensions. The result of this conceptual extension of his understanding of the design situation was an introduction of the alternately (zigzag) laid-out drums. These featured higher effective surface acting on the paper; thus, smaller rolling section was needed, and both size- and pressure-related constraints could be managed at the same time.

a) *Explicit problem specification (requirements)*:

- 'Make paper smooth, debris-free';
- 'Ensure paper has even thickness as specified';
- 'We want to produce paper having thickness with a precision of 0.01 mm'

b) *Explicit problem specification (constraints)*:

- 'Minimal/maximal pressure for a non-destructive deformation of paper is given by …';
- 'The rolling section should not have more than three drums to keep it simple'

c) *Domain theory*:

- 'Theory of mechanics, incl. equations for friction calculation, pressure distribution, material deformation/shaping, etc.';
- 'If a maximal strain of the material is exceeded, this may lead to the permanent (non-elastic) destruction of the material'

d) *Conceptual design structures*:

- A sequence of 'rolling drum pairs' with a decreasing gap between them through which the material was passing.' (see also for illustration)

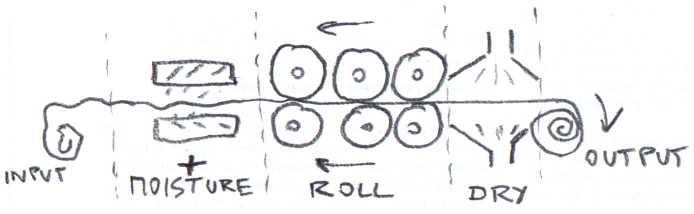


Figure 10–5. Paper-smoothing assembly *before* the discovery of a contradiction

e) *Explicit contradiction*:

- 'In order to improve quality of rolling, we add more drums';
- 'More drums may cause problems with maintainability/control';
- 'Squeezing of rolling drums does not help…'

⇒ 'It is **not possible** to extend the rolling section and maintain simplicity and vice-versa, it is not possible to shrink it and maintain the quality'

f) <u>Conceptual resolution</u>:

- ‘What about squeezing the drums in **two dimensions** instead of one?’;
- ‘This would increase the surface the drums are interacting with the paper, thus requiring a smaller rolling section and lower pressure’

g) <u>Implementation of a conceptual resolution (new conceptual primitive)</u>:

- ‘**Zigzag layout** instead of linear layout is actually a squeezing in two dimensions’
- ‘The new product is proposed in Figure 10–6.’

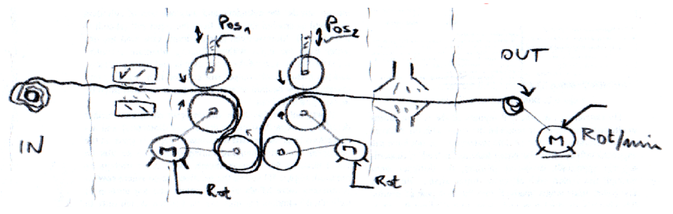


Figure 10–6. Paper-smoothing assembly *after* the resolution of a contradiction

The heralded *re-framing* of the design problem is observable in the articulation of a move from ‘one-dimensional’ design solutions towards ‘two-dimensional’ products; from a linear layout to the alternate one. This is a change of the conceptual frame; a different design frame is introduced to tackle the issue, rather than proposing an alternative or refining the problem specification. We justify our opinion by pointing to the fact that a different conceptual ‘vocabulary’ is needed in order to articulate objects and relations in multidimensional spaces.

Nevertheless, the problem specification remained unchanged; designer still wanted the smooth paper of an even thickness. Only the contradictory conceptual terms were omitted from the new conceptualisation, and were replaced with new, *functionally alternative* but sound concepts. Unlike schema 3 that only refined the existing problem specification using a new frame, this schema is introducing a new vocabulary for the construction of problem solving models and design solutions. Thus, in this particular case, schema 4 was responsible for a tacit rectification of a contradiction in the problem solving theory, and clearly featured the transition *7 → 8 → 3b*.

10.3 RFD extensions concluded

This chapter presented a few interesting extensions and refinements to the original RFD model. These particular schemas and patterns of reasoning in design emerged as the directly explainable results of our experimental study. We think that the extended RFD model is a more accurate description of how the process of design framing and re-framing can be explained. Nevertheless, this does not mean that this is the ‘final’ version. In the terminology of our own RFD theory, we rather say that the extended version of RFD model (see Figure 10–1 earlier in this chapter) is an *acceptable* solution to our ill-structured research problem.

However, it is acceptable from the perspective of an abstract model of a design process. It is far from complete or acceptable in respect to the practical implementation. There is still a lot of work needed to turn the highly abstract claims and conclusions from this research into a ‘practice-ready’ design support system. Nonetheless, it was not the intention of this research and this thesis to deliver a fully-fledged, off-the-shelf practical solution to a problem that is very poorly structured. We sought to acquire a deeper insight and understanding of the process of design (re-)framing on the knowledge level. To conclude the extension of the RFD model, we may say that this is the essential contribution of the presented thesis, and we believe that our initial objectives were achieved.

11 SUMMARY

As reviewed in chapters 4 and 5, much effort has been put into modelling engineering design as a problem solving task. However, many existing models of design task focus solely on finding a suitable solution to a given problem specification. We showed in the review that the issue with the interpretation of problems was often left aside, although the empirical studies reported various intuitive decisions made by the designers at this stage (Dominowski and Dallob 1995). Let us summarise the essential claims presented in the preceding chapters of this document. As a whole, this thesis presented design as an interaction of two knowledge-level actions – *problem solving* and *problem specification*.

We said that a task of *design* might occur whenever an agent decides to change the status of a surrounding world (Smithers, Conkie *et al.* 1990). Usually, the task starts with initial, often incomplete objectives, and leads to an artefact (a solution) realising the change. It has been accepted that design is an ill-structured task (Simon 1973), and requires a significant effort to understand the 'structure' of the problem. What does it mean to 'give a problem its structure'? Is it possible to model this 'structuring' (or *framing*) using formal means instead of referring solely to designer's 'intuition' and 'insight'?

Empirically, it is known that designers are rarely presented with a detailed specification of design problems (Schön 1983). In line with the empirical observations, we argued that a problem specification was subject to a similar evolution as a design solution. Moreover, a set of statements expressing a desired state may be proclaimed '*a design problem specification*' only at the end of design, once the designer *accepts* the proposed artefact as a design solution. The idea of co-evolving design solution and specification is not new (Swartout and Balzer 1982; Nidamarthi, Chakrabarti *et al.* 1997). However, the empirical findings lack a formalised theory. Schön (1983) observed that the practitioners often 'know' what to do to achieve their goals. They use their experience when tackling new ill-structured problems. In our work, we extended his research on experience-based *problem framing* as a design phase.

11.1 Problem interpretation and frames

We already mentioned that problem framing is an important phase of the ill-structured design problems (see chapter 4). Nakakoji, Sumner *et al.* (1994) acknowledge the existence of something like 'problem framing', and talk about 'design perspectives'. Accordingly, a perspective "*is a point of view, which implies that certain design goals exist, certain bodies of design knowledge are relevant, and certain solution forms are preferred*." The term 'design perspective' is used in the context of expressing the designer's intentions. We compared such a perspective to a vocabulary of concepts used during the problem solving phase, and based our working definition of 'design frame' upon a similar position.

In summary to the review in chapter 4, we subscribed to an assumption that framing was an important action that typically preceded and complemented the problem solving phase (Schön 1983; Fischer 1992). The review was concluded with a formulation of several challenges we have undertaken to tackle in this thesis. What are the implications of 'framing' on the knowledge level? What is actually happening with the designer's knowledge during the problem framing? Can a problem re-interpretation or framing be expressed in a formal language? These were the essential research questions we sought an answer for in this thesis.

11.1.1 Concepts introduced in the thesis

We have assumed that a designer tackled an ill-structured design problem, and denoted this design problem as $\mathfrak{DP}$. In order to solve design problem $\mathfrak{DP}$, the designer made an explicit commitment to the conceptual terms that delineated a hypothetical space of problem specifications $\mathcal{S}$. Such ***explicit problem specification***, denoted as $\boldsymbol{S} \sqsubseteq \mathcal{S}^*$, provided a context for applicable design methods and domain theories. These methods and theories were identified as the tools enabling the designer to ***satisfy*** the explicit specification. However, since the problem specification is only an interpretation of $\mathfrak{DP}$, the solution satisfying $\boldsymbol{S}$ must not necessarily be a solution to $\mathfrak{DP}$. This is the argument that distinguishes this thesis from other work. Repeating from chapter 6, a problem specification is a set of statements describing the desired states, expressed in a suitable language; e.g. first-order logic. From the perspective of formal theories, we mentioned that the designer *circumscribed*[15] $\mathfrak{DP}$ by declaring that only the statements from the explicit specification $\boldsymbol{S}$ were needed for interpreting (and later solving) the problem. This circumscriptive step was expressed by assertion in Eq. 11–1.

$$\exists S \sqsubseteq \mathcal{S}^*: \mathit{specifies}_{\Phi}\,(S, \mathfrak{DP}) \qquad \textbf{Eq. 11–1}$$

We argued that such an assertion could be made only inside certain conceptual boundaries that were named a ***conceptual design frame***. We defined design frame $\boldsymbol{\Phi}$ as a pair of two circumscribed knowledge spaces that were constructed from the relevant problem specification primitives $\mathcal{S}$ and the relevant problem conceptualisation $\mathcal{T}$. Thus, 'framing a design problem' meant articulating a set of conceptual objects $\mathcal{T}$ that might be used for doing the design as specifiable by the concepts from $\mathcal{S}$ (relevant problem specification primitives). Formally, design frame $\boldsymbol{\Phi}$ was defined as $\boldsymbol{\Phi} = \langle \mathcal{T}, \mathcal{S} \rangle$.

We also introduced symbols $\mathcal{S}^*$ and $\mathcal{T}^*$ to express the above-mentioned circumscribed knowledge spaces formally. We defined $\mathcal{T}^*$ as 'a closure' constituted by the selected conceptualisation $\mathcal{T}$ and an appropriate domain theory $\boldsymbol{DT}$. Domain theory $\boldsymbol{DT}$ was understood as problem-independent knowledge, possibly applicable to different problems. A generic domain theory $\boldsymbol{DT}$ is 'instantiated' for each particular conceptual base $\mathcal{T}$, in order to obtain a usable theory for solving a particular problem. We referred to closure $\mathcal{T}^*$ as a ***problem solving theory***. Similarly, we defined $\mathcal{S}^*$ as a hypothetical instantiation of the potentially relevant problem specification statements ($\mathcal{S}$) in the chosen conceptual world $\mathcal{T}$ and selected domain theory $\boldsymbol{DT}$.

a) Conceptualisation $\mathcal{T}$ was likened to an ontology, a vocabulary of basic concepts, for which the designer decides they are available for expressing statements about a particular design problem. This vocabulary may include a terminology defining functional and structural objects, mappings between them, e.g. in form of behaviours, etc. (Bylander and Chandrasekaran 1985; Gero 1990; Choueiry, McIlraith *et al.* 1998).

b) Domain theory $\boldsymbol{DT}$ was likened to a shared, generic ontology defining a background vocabulary (Wielinga, Akkermans *et al.* 1995), in which any conceptualisation is applied. This generic theory had to be interpreted for a specific conceptual base to derive a problem solving theory.

[15] This operation indeed circumscribes designer's understanding of an incompletely defined (i.e. ill-structured) problem (McCarthy 1980; 1986).

c) A hypothetical space of problem specifications $\mathcal{S}^*$ within a particular frame $\boldsymbol{\Phi}$ provided an ontology of the desires or intentions of a designer expressible in the context of a particular design frame (Nakakoji, Sumner *et al.* 1994). It was seen as a set of relations articulated using the conceptual elements $\mathcal{T}$ and a particular problem solving theory $\mathcal{T}^*$.

Design frames are constructed dynamically during the design process from the information available to the designer as an initial design brief. Typically, a designer uses this initial brief to decide on the explicit problem specification ($\boldsymbol{S}$), and identify similar design situations giving him a vocabulary of the familiar terms. An explicit commitment to particular familiar concepts helped to circumscribe the current problem in those terms. This was an important corollary of our theory, because it corresponded with Schön's (1983) empirical observations that "*designer sees one problem as another*".

Conceptual design started with a formulation of an explicit design frame $\boldsymbol{\Phi}$. In this frame, the ill-structured problem can be interpreted, specified and solved. Next, the designer tries to formulate a minimal sub-set $\boldsymbol{T} \subseteq \mathcal{T}^*$ that *satisfies* given problem specification. 'Given' specification corresponded to the explicit set $\boldsymbol{S} \subseteq \mathcal{S}^*$, to which designer made a particular commitment. Thus, a designer tried to shrink the space defined by a problem solving theory $\mathcal{T}^*$ into a chunk that could be manipulated with and explored. This chunk was called a ***problem solving model*** (Altshuller 1984; Gero 1990), and defined as a minimal sub-set of the problem solving theory that sufficiently *satisfied* the explicit problem specification. Relation '*satisfies*' was a binary association of a problem solving model $\boldsymbol{T}$ and current explicit problem specification $\boldsymbol{S}$ (with design frame $\boldsymbol{\Phi}$ as a parameter); see Eq. 11–2.

$$\exists T \subseteq \mathcal{T}^*: satisfies_{\Phi}\,(T, S) \wedge (\neg\exists Y \subset T: satisfies_{\Phi}\,(Y, S)) \qquad \textbf{Eq. 11–2}$$

From an operational point, we mentioned benefits of distinguishing design requirements $\boldsymbol{R}$ from constraints $\boldsymbol{C}$; the explicit problem specification could be a union of the two; i.e. $\boldsymbol{S} = \boldsymbol{R} \cup \boldsymbol{C}$ (Wielinga, Akkermans *et al.* 1995). Requirements were statements demanding the explicit presence of a particular feature, and constraints were conditions that must not be explicitly violated by any candidate solution. Subscribing to such classification, we proposed a simple model defining the 'insides' of relation '*satisfies*' (Eq. 11–3). This topic was discussed as marginally relevant to the main argument of this thesis in section 6.4. Other classifications may lead to other operational definitions.

$$satisfies_{\Phi}\,(T, S) \Leftrightarrow \{(S = R \cup C) \Rightarrow T \models R \wedge \neg(T, C \vdash \perp)\} \qquad \textbf{Eq. 11–3}$$

However, let us re-iterate the fundamental point. The explicit problem specification $\boldsymbol{S}$ is only an interpretation of a design problem $\mathfrak{DP}$ that can be used for problem solving. It is only a 'model' of the actual problem $\mathfrak{DP}$. We argued that the existence of a problem solving model $\boldsymbol{T}$ '*satisfying*' the explicit problem specification $\boldsymbol{S}$ was a <u>necessary but not sufficient</u> condition for $\boldsymbol{T}$ being a 'design solution'. In addition to satisfying the explicit specification, the model $\boldsymbol{T}$ had to be '*acceptable*' as a design solution addressing the original problem $\mathfrak{DP}$. However, we mentioned that the '*solution acceptability*' was often appreciated subjectively and tacitly, and might not be expressible in the explicit terms in advance.

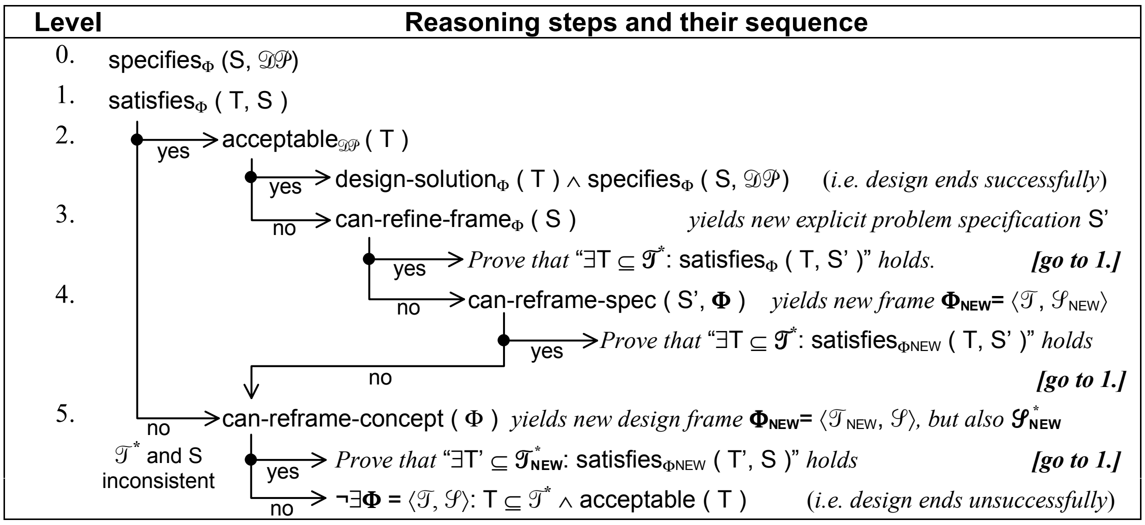
Nevertheless, solution acceptability was introduced as a residual category (see Eq. 11–4). We argued that the 'residual nature' of solution acceptability had its epistemological roots in the position given by Cook and Brown (1999), who claimed that tacit decisions could not be stripped of their contextual background. It could be difficult to define the explicit criteria of ‘acceptability’, but a designer may proclaim a certain problem solving model acceptable or not, when s/he reflects on it.

$$\textit{satisfies}_{\Phi}\,(T, S) \wedge \neg\textit{acceptable}_{\mathfrak{DP}}\,(T) \mid \mathbf{\Phi} \Rightarrow \neg\textit{specifies}_{\Phi}\,(S, \mathscr{DP}) \qquad \textbf{Eq. 11–4}$$

Formula in Eq. 11–4 was interpreted as an important consequence of working with the concept of design frames. It asserted that if an otherwise sound and admissible problem solving model was not accepted by a designer as a design solution, it might point to an incorrect interpretation of the actual design problem $\mathfrak{DP}$. The explicit interpretation in terms of design frame $\boldsymbol{\Phi}$ did not sufficiently reflect the ‘real’ design problem $\mathfrak{DP}$, because there were some unexpected or unintended behaviours that were unacceptable. An incorrect or incomplete design frame had to be amended or refined so that the acceptability was restored (chapter 6).

11.1.2 Recursive model of framing in design

To show the interaction of the introduced formal concepts, we proposed a recursive model of framing in chapter 6 and extended it in chapter 10. The model (code-named RFD) presented design as a series of interactions of predicates ‘*specifies*’ and ‘*satisfies*’. The first action worked with a problem specification, the second one tried to satisfy such specification. The RFD model as the main contribution of the thesis is shown in Figure 11–1. To help read the model easily an intuitively as a sequence of decisions and actions, we defined several auxiliary predicates representing the earlier-mentioned refinements and amendments of the design frame(s). The respective definitions are listed below the figure, and explained in detail in chapter 10.



Level
Reasoning steps and their sequence
0. specifies$_{\Phi}$ (S, $\mathscr{DP}$)
1. satisfies$_{\Phi}$ (T, S)
2. yes acceptable$_{\mathscr{DP}}$ (T)
yes design-solution$_{\Phi}$ (T) ∧ specifies$_{\Phi}$ (S, $\mathscr{DP}$) (*i.e. design ends successfully*)
3. no can-refine-frame$_{\Phi}$ (S) *yields new explicit problem specification* S’
yes *Prove that* “∃T ⊆ $\mathscr{T}^*$: satisfies$_{\Phi}$ (T, S’)” *holds.* ***[go to 1.]***
4. no can-reframe-spec (S’, **Φ**) *yields new frame* **Φ**$_{\mathbf{NEW}}$= ⟨$\mathscr{T}$, $\mathscr{S}_{\text{NEW}}$⟩
yes *Prove that* “∃T ⊆ $\mathscr{T}^*$: satisfies$_{\Phi\text{NEW}}$ (T, S’)” *holds*
no ***[go to 1.]***
5. no can-reframe-concept (Φ) *yields new design frame* **Φ**$_{\mathbf{NEW}}$= ⟨$\mathscr{T}_{\text{NEW}}$, $\mathscr{S}$⟩, *but also* $\mathscr{S}^*_{\mathbf{NEW}}$
$\mathscr{T}^*$ and S inconsistent
yes *Prove that* “∃T’ ⊆ $\mathscr{T}^*_{\mathbf{NEW}}$: satisfies$_{\Phi\text{NEW}}$ (T’, S)” *holds* ***[go to 1.]***
no ¬∃**Φ** = ⟨$\mathscr{T}$, $\mathscr{S}$⟩: T ⊆ $\mathscr{T}^*$ ∧ acceptable (T) (*i.e. design ends unsuccessfully*)

Figure 11–1. RFD model – design as interactions of problem framing and problem solving

$$\textit{can-refine-frame}_{\Phi}\,(S) \Leftrightarrow \exists s \in \mathscr{S}^*: S \subseteq \mathscr{S}^* \wedge S' = S \cup \{s\} \wedge \textit{specifies}_{\Phi}\,(S', \mathscr{DP}) \qquad \textbf{Eq. 11–5}$$

$$\textit{can-reframe-spec}\,(S, \mathbf{\Phi}) \Leftrightarrow \exists \mathbf{\Phi}_{\mathbf{NEW}} = \langle \mathscr{T}, \mathscr{S}_{\text{NEW}} \rangle: S' \subseteq \mathscr{S}^*_{\text{NEW}} \wedge \textit{specifies}_{\Phi\text{NEW}}\,(S', \mathscr{DP}) \qquad \textbf{Eq. 11–6}$$

$$\textit{can-reframe-concept}\,(\mathbf{\Phi}) \Leftrightarrow$$
$$\Leftrightarrow \exists \mathbf{\Phi}_{\mathbf{NEW}} = \langle \mathscr{T}_{\text{NEW}}, \mathscr{S} \rangle: T' \subseteq \mathscr{T}^*_{\text{NEW}} \wedge S \subseteq \mathscr{S}^*_{\text{NEW}} \wedge \textit{specifies}_{\Phi\text{NEW}}\,(S, \mathscr{DP}) \qquad \textbf{Eq. 11–7}$$

The predicate defined in Eq. 11–5 represented the simplest ‘model’ for the modification of a design frame. In this particular case, the problem was not re-framed, it was still attended to in the original design frame. Only its explicit specification was refined by an explicit commitment to other features or properties already known in the original frame. Eq. 11–6 represents a similar model that also aims to refine the explicit problem specification. However, in this case, the specification can be refined only when the designer re-frames the design problem. In other words, the refinements could not be expressed in the vocabulary of the original frame, and a new frame with new concepts was needed.

Lastly, the definition in Eq. 11–7 is the most complex one. First, it forces the designer to re-frame the problem, and second, it focuses not on the explicit specification but rather on the conceptualisation of the problem. In other words, the designer has to amend a design frame to acquire new conceptual objects from which a solution may be constructed. Unlike the other predicates manipulating the frame the *conceptual re-framing* could occur after the failure of the tacit acceptability check, as well as of the explicit admissibility. The individual patterns and schemas are summarised later in section 11.2.

The interplay of the RFD model in Figure 11–1 is achieved through a recursive step back to the top-most level of the model after each frame refinement or amendment. The labels '*go to 1.*' means that the control flow returns to the attempt to satisfy the refined or extended problem specification in the same or in the new conceptual frame. Alternatively, the satisfaction algorithm may work with the original specification but it may deploy different conceptual primitives for the construction of the solution.

The RFD model and its visual depiction in Figure 11–1 is the main contribution of the thesis. It is a formal representation of an empirically observed interplay of two knowledge-level actions. Patterns, which occurred in the design experiments we conducted, can be directly associated with the extended recursive model. We noted that these patterns were *reasoning schemas*, rather than *problem solving methods*. They were discussed in depth in section 6.3 and 10.2, and are summarised in 11.2.

11.2 Schemas of re-framing and frame refinement

Each schema was represented as a path leading to one of the 'leafs' in the decision sequence shown in Figure 11–1. The simplest and conceptually most straightforward path led directly to a design solution – thus generating a problem solving model that was both admissible and acceptable. A set of problem solving methods and theories that were able to implement the relation '*satisfies (T, S)*' operationally was discussed in section 6.4. This simplest[16] case did not feature any references or manipulations to the design frames. Therefore, we did not analyse it from the perspective of re-framing. Nonetheless, it was described as a 'schema' in section 6.3.1.

The three techniques featuring frame refinement or problem re-framing were elaborated after the empirical study, and are detailed in section 10.2. Below, we shortly summarise each of them.

11.2.1 Design frame refined by implicit problem specification

We suggested that a reason for the observation of this schema is the fact that the designers work with many implicit expectations, in addition to the explicit specification of a design problem. These expectations are not expressed formally or in the same language as the explicit requirements or constraints. For instance, Nakakoji, Sumner *et al.* (1994), and Nakakoji, Yamamoto *et al.* (1998) refer to a similar phenomenon from a cognitive perspective as 'design intentions', and argue that many

[16] This is a conceptual simplicity; the problem solving method may not be computationally simple.

intentions remain tacit. A similar assertion may include also implicit expectations (Dienes and Fahey 1998), as well as tacit ones (Cook and Brown 1999).

We claimed that the process of 'surfacing tacit intentions and mapping them into explicit representations' as Nakakoji, Sumner *et al.* (1994) see it, is mainly about an explicit formulation of implicit, 'hidden' intentions. Designers often take certain things for granted, and they do not explicate them until necessary. Their tacit knowledge and experience may help to formulate the important though 'forgotten' expectations, but the result is only an explicit formulation of a statement in the language of a particular conceptualisation. In other words, such a statement must be (and usually is) *expressible*, but simply has not been *expressed explicitly*.

From a perspective of conceptual framing, we said that the design frame remained unchanged in this schema. Thus, no new objects or concepts are added to sets $\mathcal{T}$ and $\mathcal{S}$; only the explicit problem specification is monotonically extended by one or more explicit commitments to the features that were implicit and unattended ($\boldsymbol{S} \subseteq \mathcal{S}^* \rightarrow \boldsymbol{S'} \subseteq \mathcal{S}^*$). The refinement of problem specification is followed by a new attempt to find a satisfactory solution, and the whole cycle may repeat.

From a knowledge-level point of view, the new statement was seen as *synchronisation* of the explicit conceptual frame with the implicit expectations. These implicit expectations tacitly influenced the designer's decision on solution acceptability. Such expectations usually remain 'hidden'; however, when an admissible candidate solution was found tacitly unacceptable, they were surfaced. Reflecting on the 'hidden' (perhaps empirical) expectations led to an explicit articulation of a new statement that was originally forgotten. This schema was defined in section 10.2.1.

11.2.2 Refinement of problem specification via re-framing

Situation from section 11.2.1 required only a simple refinement of the existing problem specification that was modelled with the same conceptual primitives. In some experimental sessions, we observed also other types of refinement. We mentioned that this refinement had similar aims as the schema described in the previous section. While the previous schema was expressed as '*I have forgotten to say that λ must have value y*', this new pattern was typically justified in a more 'tentative language' of conjecturing possible causes of the solution unacceptability. Its verbal description could be. "*I cannot solve the problem specified in this way. What if I looked from a different side and assumed XY?*"

This 'what if' exploration introduced new requirements or assumptions that refined the explicit problem specification. Nonetheless, this introduction required more effort than the simple refinement from section 11.2.1. We modelled this 'extra effort' in terms of articulating a new design frame; i.e. by *re-framing* the design problem. The refinements were not possible in the original frame; the designer needed a new vocabulary allowing him or her to articulate new specifications in explicit terms.

From the conceptual viewpoint, the articulation of such a speculative assumption may separate the different design contexts for problem interpretation and specification. However, this refinement draws upon the tacit discovery or awareness of the potentially useful requirements/constraints. A source of such potentially relevant, tacitly discovered requirements/constraints/assumptions may be in the base of previously tackled cases (as observed in the experiments). Unlike the previous refinement, this schema is more speculative, and it often starts with the tentatively assumed conditions rather than immediate explicit commitments. This tacit perception of relevance needs to be upheld – therefore, a designer typically returns back to the construction of a solution satisfying the conjectured specification (see label '*go to 1.*' in Figure 11–1). This schema was defined in section 10.2.2.

11.2.3 Conceptual re-framing in contradictory theory

This schema is typically invoked when the current frame is unable to propose any suitable problem solving models that would be consistent with the required and specified features. The reason may be that some of the specified features are mutually contradictory, and cannot co-exist in a form that is dictated by the current problem solving theory $\mathfrak{T}^*$. This conflict has its roots in the problem solving theory being an instantiation of a generic domain theory $\boldsymbol{DT}$ using the chosen conceptualisation $\mathfrak{T}$. The generic domain theory may be sound, but its interpretation, in terms of the chosen conceptualisation, may lead to the contradictions.

Unlike in sections 11.2.1 and 11.2.2, where only the problem specification was refined or extended, a similar amendment of the specification does not resolve a conflict of this type. This contradiction goes conceptually deeper than an absence of a desired feature. Often, the conflict can be traced to two or more mutually incompatible requirements. Nonetheless, this incompatibility is always judged in respect to a particular design frame (see predicate '*satisfies*' in section 11.1.1). In other words, the conflict between the requirements is present because of the designer's commitment to particular conceptual objects (and a particular domain theory).

Since according to our initial assertions, none of the violated requirements can be 'retracted', the designer is forced to re-visit the underlying domain theory $\boldsymbol{DT}$, interpreted in conceptual terms $\mathfrak{T}$. By defining new conceptual primitives, the designer actually changes his conceptual vocabulary for interpreting and solving the ill-structured design problem $\mathfrak{DP}$. New concepts and possibly a different domain theory are chosen so as to remove the conflict. We supported this conceptual schema by referring to a related work on 'domain contradictions' that drew upon an extensive empirical study of inventive problems (Altshuller 1984). The presented model extended the empirical findings by relating such 'domain contradictions' to a relativistic concept of 'design frame'. This schema is described twice – in section 6.3.3 it refers to Altshuller's study, and in 10.2.3 to our experiments.

11.3 Summary of empirical study

The research in this thesis focused mainly on the definition of the theoretical apparatus suitable for the investigation of the problem domain. Consequently, definitions were used in the development of a conceptual model of framing in design. The experimental background of this thesis consisted of a set of 24 experiments with two design practitioners. They solved problems from a domain of controllers for large-scale systems, and during the design sessions they were asked to record both informal justifications and ideas, as well as formal 'milestones' or explicit commitments to particular requirements and solutions.

These records were subsequently annotated and analysed in terms of the initial version of the RFD model. The examples of two annotated design sessions and their translations into the terminology introduced in the theoretical part of the thesis are available in chapter 8. The results of the analysis were presented in the graphical form in chapter 9. Let us briefly summarise how the experiments were annotated, analysed and evaluated.

We started with splitting the design session into several explicitly recorded *design contexts*. Context was defined by a (sub-)set of the explicit requirements and constraints, which the designer identified. Each context was typically accompanied by a formulation of a solution that addressed the explicit requirements. These design contexts represent 'milestones' in the process of problem shaping (i.e. specifying and solving). Between the design contexts, we identified one or usually more *design episodes*. An episode was a coherent reasoning chain focusing on a particular goal or elaborating a particular idea more in depth. Each episode consisted of several *reasoning steps* that corresponded directly to the conceptual terms we defined for the initial RFD model. Thus, the annotation of experimental sessions included the recognition of the particular reasoning steps in the individual episodes. Such annotation was conducted for all observed sessions.

The analysis started with the aggregation of the annotated sessions and visual representation of the *sequences* of the reasoning steps. The sequences from every single design episode and context were fed into a graph as links between the nodes representing the individual reasoning steps (actions). The more frequently a particular sequence occurred, the thicker particular link was drawn. Eventually, this simple method of aggregating the multidimensional data produced a graph with several interesting 'patterns' symbolising frequently occurring reasoning steps and coinciding decisions.

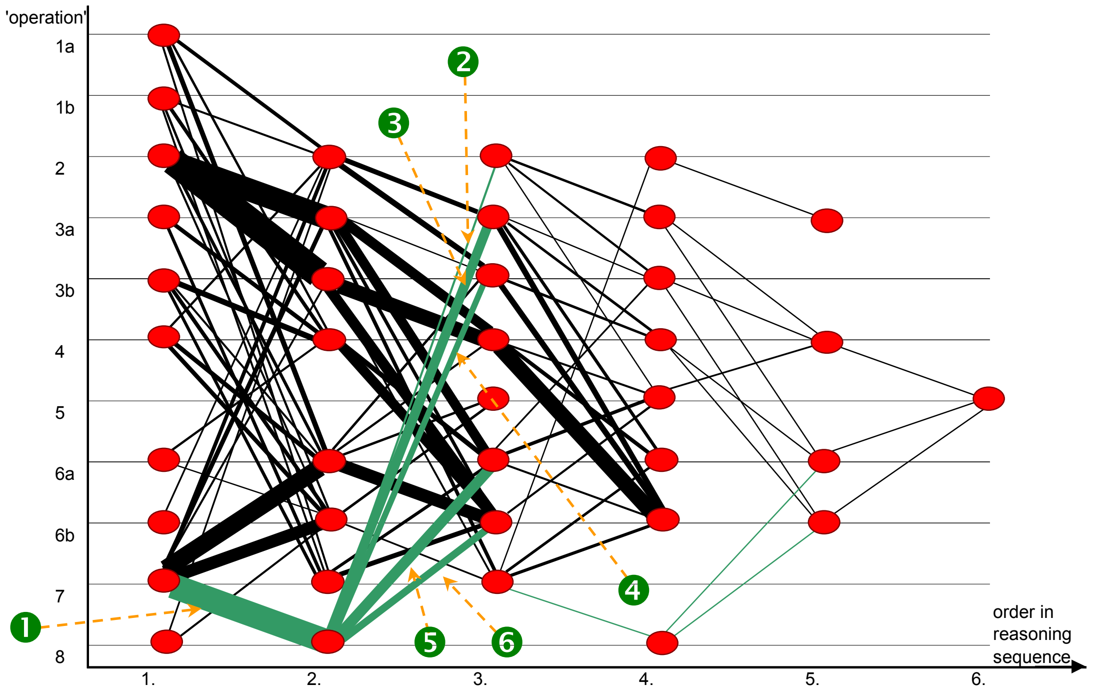


Figure 11–2. Aggregated results of the experimental analysis

For instance, Figure 11–2 shows how such aggregation looked like. This specific 'pattern' refers to the different types of frame amendment that were observed in the design sessions. The actual frame shift is labelled as ❶; operations ❷, ❸, and ❹ feature significant extensions of the explicit problem specification or solution. Operations ❺ and ❻ depict what we later defined as refinements to the problem specification or solution via re-framing. More details can be found in chapter 9.

The identified patterns were later evaluated, and served as a basis for the extension of the initial RFD model. The extensions and their implementation into the RFD framework were already summarised in section 11.2. Nevertheless, further details can be found also in chapters 9 (correlations) and 10 (schemas of reasoning, extended model).

11.4 Position of the conducted research

The research reported in this thesis is relevant to several areas and disciplines. Its multidisciplinary character is reflected in chapters 4 and 5, where theoretical and methodological foundations are reviewed and evaluated. Let us briefly summarise the relevance to other disciplines. The proposed RFD model is in accordance with the empirical findings, as reported by Schön (1983). Schön observed design practitioners and reported an oscillation between the solution development and reflection on it. He also suggested that the reason for such an oscillation might be in the 'surprising behaviours' of the proposed solution. We referred to this phenomenon as inconsistency or a lack of synchronisation between the explicit design frame and various implicit and tacit expectations of the designer. The purpose of reflection was to allow the designer to perceive their solution from a different perspective, so that they might become aware of new features, improvements or extensions to the current design.

We argued that 'surprises' and 'unexpected results' occurred in the design because of the designer's incomplete knowledge about the design problem in a particular frame. The incompleteness was due to the designer's *interpretation* of the problem in explicit terms, in order to apply various problem solving methods and techniques. Inevitably, any result or solution produced by such problem solving methods addresses only the interpreted variant of the original problem – its explicit specification. This reflection on tacit/implicit knowledge was expressed in our model as an assessment of solution *acceptability*, and it was strictly distinguished from an assessment of the explicit *admissibility* during the problem solving. If a solution was found unacceptable, the inadequate frame needed to be refined or shifted.

We observed that one form of problem *re-framing* was inspired by Altshuller's (1984) work on the contradictions that were typically triggering the inventive problems. His inventive algorithm TRIZ was based on the premise that each such contradiction could be resolved by an application of a suitable 'solution model' – an abstracted, principled approach that was closely related to the particular pair of conflicting requirements/properties. This approach basically involved the introduction of unusual concepts and components into the problem. In our terminology, the resolution of a contradiction required a new conceptual frame, in which the explicit problem specification is made consistent, and in which solutions may be found. Our schema 4 (section 10.2.3) reflects the principles of TRIZ.

We also mentioned that a design frame is typically expressed in the familiar terms, terms acquired from the past exposures to the similar design problems. Therefore, we argued that reasoning by analogy (Maher 1990; Maher, Poon *et al.* 1996) and case-based reasoning (Bhatta and Goel 1994; Kolodner and Wills 1996; Watson and Perera 1997) were also closely related to the presented research. However, unlike the existing systems and models, we focused more on the problem framing and re-framing rather than retrieval, adaptation or learning. We believe that our model complements various implementations of the design support systems that were developed in many different paradigms. The existing tools and models can be seamlessly incorporated into our model to implement the individual predicates similarly as we showed for the logical abduction, deduction and truth maintenance in section 6.4.

We also touched on the research interests of cognitive science. As mentioned earlier, the formal model of framing incorporates similar interactions typically associated with a solution talkback (Schön 1983; Nakakoji, Sumner *et al.* 1994). This talkback or a dialogue with a design situation is considered an essential characteristic of many non-trivial, ill-structured problems. This phenomenon appears very frequently in the cognitive science literature (Candy and Edmonds 1996; Cross 1997). However, it has not been formally described and analysed. This is what we tried to rectify in this thesis by defining various models of such a talkback and visualising them in the aggregated results.

An important research area that shares an interest with the presented research is the incompleteness of problem specification. We argued that the premise that can be found in many problem solvers and assuming that a problem is given, was partly incorrect. While it is possible to use various problem solving methods within certain closed boundaries, these boundaries or *frames* may be re-visited and amended during the problem solving exercise. Problem framing was presented as a kind of 'temporary' circumscription of an incomplete knowledge space for a particular design problem. By committing to a set of explicitly selected conceptual primitives for a problem specification, designers deliberately circumscribed (McCarthy 1980) the world for their problem solving. Nevertheless, they still could re-open that closure, and circumscribe the problem space in an entirely different manner. To that extent, we proposed several schemas of how and why the 'closed frame' can be re-opened.

Formally, the research in this thesis belongs to the category of non-monotonous reasoning methods (McDermott and Doyle 1980). This is particularly visible in the schemas of *re-framing*, where we allow the introduction of new conceptual objects and primitives to the problem solving theory. The reason of such non-monotonous extension was to amend the problem solving theory that originally produced a sound (*admissible*) albeit *unacceptable* solution. The re-framing schemas featured in some instances rather speculative assertions, which the designer believed would resolve the unacceptability. However, they had to be validated, and the validation typically happened in an attempt to generate a solution in such an amended frame. If an admissible and acceptable solution was found, the non-monotonous conjecture was upheld, otherwise the problem could be re-framed repeatedly.

This validation through solution appears in the cognitive literature. Our empirical study (Dzbor and Zdrahal 2001b, 2002), and those of other authors confirm an open-ended nature of both design specifications and solutions (Candy and Edmonds 1996; Nidamarthi, Chakrabarti *et al.* 1997). The essence of our argument was that the design problems might be conceptually *interpreted* by solving them. In other words, a solution is developed to address a particular problem specification, and vice-versa; a specification develops in response to the solution behaviour. In this way, the proposed RFD model expressed Schön's empirical findings regarding *reflection* on action, and related it to non-monotonous reasoning and reasoning by analogy. The 'problem interpretation through solving' and 'solution generation through re-interpretation' feature a knowledge-level interaction in design. Through this interaction, our model addresses and formally expresses the exploratory nature typical of inherently open-ended design problems (Smithers, Conkie *et al.* 1990). Interactive exploration is even more visible and important for creative and innovative problems (Gero 1996; Goel 1997).

12 CONCLUDING REMARKS

In this thesis, we have presented, illustrated and empirically validated a novel approach to modelling and explaining the design process. As promised in section 3.3, this approach has been developed on several mutually related knowledge levels. The main outcome of this work is in the formulation of a recursive model of framing in a design process in chapter 6. The model, code-named RFD, represents an interesting formalisation of a grey area in the science of design. This model was applied to the knowledge-level description of an experimental study that was described, annotated and analysed in chapters 7 and 8. Eventually, several schemas implied by the model were identified and elaborated in chapter 10. Hence, in accordance with the promises in section 3.3, this thesis contributes to the research in the areas listed below. Each of the 'contribution classes' is detailed in the subsequent sub-sections.

- Contributions to the epistemological foundations and design theory;
- Contributions to modelling the inarticulate reasoning processes in 'design(ing)';
- Contributions to the empirical and methodological research

The proposed approach towards modelling design processes as an interaction of problem framing and problem solving is not entirely new. As we mentioned in chapters 4 and 5, this phenomenon was introduced in cognitive science several years ago. Nevertheless, so far there were no formalisations of the interaction between these complex reasoning operations. The recursive model is thus a rare attempt to express the existing knowledge in formal and structured manner. In spite of rather abstract, knowledge level on which the model is defined, it is a firm step in the clarification of design process. Although, this thesis did not present any operational or computational models suggesting the implementation of the RFD schemas, the schemas may be used to inform design – methodological guidelines. However, as a part of further work, we discuss some of the ideas towards implementing the RFD theory.

12.1 Epistemology of design

In chapter 6, a recursive model of framing in a design process was proposed as an alternative approach to modelling design. From an epistemological perspective, one of the essential contributions of this model is the formal account of those phases of the design process that are, in most other models, rather vague. Especially, we emphasise the formal position of '*problem interpretation*' in terms of articulating and amending a '*conceptual design frame*' for an explicit specification of the desired states as well as satisfaction of such a specification. This phase of designing, which is referred to as '*problem framing*', received only limited attention in the existing research activities of the community.

Problem interpretation has been embodied in the mutual interaction of predicates '*specifies*' and '*satisfies*'. The former one asserted that a certain set of explicit statements was sufficient for expressing *relevant* desired states the design was aiming to achieve. The purpose of the second predicate emerged with the realisation that the validity of predicate '*specifies*' might not be provable directly in any problem solving theory. A particular specification can be upheld or rejected only drawing upon the validity of a complementary predicate '*satisfies*' and the (un-)*acceptability* of the considered candidate solution (e.g. technological artefact, product). It was the role of the predicate '*satisfies*' to find and derive such a candidate solution.

Another important corollary of the proposed recursive model of framing acknowledged the fact that the articulation of a conceptual design frame semantically corresponded to a kind of 'temporary' circumscription. The temporary nature has been explained in terms of tentativeness of the conceptual vocabularies and their potential for amendment or extension at any stage of the design. As shown in section 6.2, the amendment of conceptual frames (i.e. design problem re-interpretation) was not a random search but exhibited certain patterns. These patterns or schemas were further elaborated and clarified in section 10.2.

The reasoning patterns and schemas for the purposes of amending conceptual frames give certain structure to the problem re-interpretation. Furthermore, they serve as models of another important phenomenon that is often referred to in the cognitive literature; albeit often in abstract terms. The schemas implied by the framework of recursive framing are, to the certain extent, models of the phenomenon called '*solution talkback*' (Nakakoji, Sumner *et al.* 1994) and '*reflection on action*' (Schön 1983). As we already stated in the summary (chapter 11), this is an important contribution of this thesis, because it provides a theoretical foundation to the empirical observations and conclusions. Moreover, our RFD model is a formalised account of the above-mentioned cognitive actions that often appear very fuzzily in the existing case studies, and it provides insight into control and knowledge flow for these phenomena.

Our theoretical base complies with the above-mentioned empirical findings, and the argument that it is the purpose of reflection to articulate the designer's intentions in respect to a particular design problem was upheld in our research. Nonetheless, as the recursive model of framing implies, any consequence derived in certain design frame may be 'intended' or 'surprising' only *within* that frame. Thus, as a corollary, any 'surprise' is necessarily a manifestation of an inadequate interpretation of the design problem. Such surprises were modelled in our RFD theory by defining and positioning a specific, 'residual' predicate '*acceptable*' that drew upon designer's tacit knowledge and expectations.

12.2 Modelling the inarticulate reasoning processes

Following from the epistemological level described in section 12.1, the concept of dual knowledge-level actions is fundamental for the contributions in the area of design as a problem solving exercise. Throughout the thesis, but mainly in chapters 6 and 10, it was argued that design is as much about interpreting and specifying the problems as it is about generating solutions. Generally speaking, there has been rather limited intersection between the research on problem interpretation and problem solving. Few notable exceptions who advocated the equality of both knowledge-level actions include (Swartout and Balzer 1982), (Fischer 1992), or (Nuseibeh 2001). Despite the gap between the publication dates of these papers, there are very few works taking this 'intertwined duality' into account.

The RFD model presented in this thesis reflects such a duality and interaction of the two knowledge-level actions – predicates '*specifies*' and '*satisfies*'. The first aspect of this advocated duality was manifested in the introduction of an explicit *completeness* and *admissibility* of a candidate solution in terms of delivering all the desired features and complying with the restricting conditions. The second aspect emerged, when the concept of explicit admissibility has been contrasted with and strictly distinguished from the *acceptability* of the candidate solution. It was argued that if an admissible candidate solution was considered unacceptable, this was pointing to an incorrect or incomplete conceptual frame that was used for the interpretation of the design problem. Thus, performing one of the two dual knowledge-level actions (the problem satisfaction), the designer may learn more about the completeness and acceptability of the problem specification.

In addition to expressing two complementary knowledge-level actions occurring in a design process, we showed that both actions (and the respective predicates) are contextually biased. In other words, they both refer to a certain explicit conceptual frame, and their completeness, validity or admissibility can be judged only within that frame. Cognitively, the design frame has been compared to the 'perspective that designers may have for understanding the problem' (Nakakoji, Sumner *et al.* 1994). Conceptually, it was suggested that design frames might be represented in terms of knowledge-level vocabularies of concepts, to which a designer makes an explicit commitment in a particular phase of the design process. Such vocabularies have been associated with the existing ontological research.

Moreover, these ontological vocabularies typically consisted of familiar terms acquired and applied in past design situations. Our experiments supported the fact that past experience is instrumental for doing and practising design. Nevertheless, it was often difficult to articulate it formally and explicitly in one step. We have predicted in the model and observed empirically, that the process of framing in the familiar vocabularies was repetitive. In this aspect, we also (correctly) subscribed to the position advocated by Cook and Brown (1999), who argued that the tacit knowledge and knowing might be beneficial for the generation of explicit knowledge.

The essence of the relation of similarity between the current and past problem emerges only in the context of a particular candidate solution, in a particular frame. Examination of consequences of an explicit commitment to a specific conceptual frame may raise designer's awareness of points and features overlooked in the current approach. From the empirical study, such was the origin of the extension schemas from section 10.2.1. For instance, the knowledge of pre-heating metal slabs before rolling was contrasted with a lack of any similar operation in the analogous problem of paper smoothing. The designer explored the reasons of pre-processing in the past case, and became aware of how a similar refinement may be translated into existing problem. Thus, an 'intuitive' articulation of a new requirement loses its mystery and suddenness in the light of the transfer of a 'familiar' experience.

Similarly, conceptual amendments in sections 10.2.2 and 10.2.3 may look confusing and sudden. An external observer may be even tempted to refer to some kind of 'enlightenment'. These reasoning turnarounds surely are sudden and hard-to-explain within a particular conceptual frame. However, as soon as we allow stepping out of the current conceptual world and re-framing it, the situation becomes less extraordinary. Thus, we believe that the references to a designer's 'deep immersion' and 'insight' into a problem (Dominowski and Dallob 1995) seem to be isomorphic to the conceptual amendments of the explicit conceptual frames and the respective problem specifications. Once we allow two complementary knowledge spaces (specifications and solutions) to interact in both directions, the essence of a poor structure of the design problems becomes clearer. Not only that, the proposed model of framing provides a more holistic view of a design process than do the other, so-to-say 'solution-centred' models do.

We have not attended to the details of the operational models and implementations in this thesis. Nevertheless, it follows from the abstract conceptual model and the definitions of the terms and predicates of the model that case-based reasoning has a crucial role. We stated that in addition to the re-use of the past solution for the new problems, one shall consider the experience- and memory-based reasoning in a broader perspective (Gomez de Silva Garza and Maher 1996). In our formalism, the conceptual design frame was defined as an explicit subscription to a particular (sub-)set of familiar concepts, relations, functions, behaviours or structures.

Thus, the reasoning by analogy may be deployed in design in two different contexts. First, it is the usual application bringing in the benefits of the *case-based reasoning* (e.g. re-use of successful solutions). The other context follows from the *model-based reasoning* paradigm. For instance, framing is a typical representative of this form of reasoning; the operation of framing serves as a skeleton or a model of understanding the incomplete problem. We believe that the combination and intertwining of these two forms of reasoning is another manifestation of the advocated duality and poor structure of design problems. Why should a designer be restricted to a single problem-solving or reasoning strategy, when there are obvious benefits in having several strategies to choose from?

The 'hybrid' reasoning by analogy found a justification in the conducted experimental studies (see chapters 7 and 8). Despite a common problem domain, the designers applied several different techniques for the problems. Moreover, this diversity has been observed between not only the individual design sessions and participants, which would be rather expectable. It emerged also *within* a single design session. Designers seemed to possess and use a variety of methods and techniques in the different stages of the design problem solving.

Taking into account the ill-structured nature of the conceptual design, which was our primary subject of investigation, this is not surprising. Drawing on the conceptually different strategies and tactics may increase the designer's chances of successfully tackling an under-specified, ill-structured problem. A variety of methods provides a better flexibility and robustness to the reasoning in design. In line with the conclusions from the experiments, we may even claim that this reflection-based flexibility and variability is a vehicle of significant importance for the generation of novel, innovative or creative solutions or interpretations of a particular design problem.

One of the important contributions of this thesis is in the formalisation of the process that is known in cognitive science as 'solution talkback' (Nakakoji, Sumner *et al.* 1994; Schön 1995; Nuseibeh 2001). We showed how problem solving might be conceptually amended in response to the 'tacit assessment' of the acceptability of the candidate design solution. We also showed that the process of formulating new statements about the problem specification is essentially very similar to the process of formulating new conceptual terms and primitives. Only, each of these amendments addresses a conceptually different knowledge space.

The research described in this thesis shows a direction towards bridging an existing gap. On one hand side and in principle, the duality of design problems is widely acknowledged in terms of ill-structured specifications that need to be refined, and respective solutions need to be developed continuously rather than ‘in one go’. Many tools and models investigate and support the reasoning *from the specifications to the solutions*. On the other hand, the phenomenon of ‘solution talkback’ (Nakakoji, Sumner *et al.* 1994; Schön 1995; Nuseibeh 2001) is often blurred and insufficiently formalised. We believe that the presented recursive model of framing the design problems addresses at least some of the many aspects of this interesting (and important) phenomenon. The concept of *framing* seems to be suitable for the explanation of the oscillations *between the problem solution and the specification*.

12.3 Methodological implications

The research reported in this thesis focused on the development of a realistic model of a design process. Nonetheless, the nature of the developed model and our experiences drawn from validating the empirical study also has methodological implications. Let us mention in this section a few aspects describing how this research may benefit the methodologies for design support. First, we provided a principled and rigorous approach to modelling a complex design process as an interaction of two knowledge-level actions. The details of these principles were summed up in the previous sections.

Since our model accounts for both aspects of design – the explicit specification of a problem and the generation of a solution, it also guides designers along certain patterns that may emerge during a design process. The model leverages the major weakness of the other models; namely, their reliance on the designer’s intuition and/or insight for the recognition of a need to change a perspective = *re-frame* the problem. Unlike these models, we attempted not to treat the designer as an indescribable ‘black box’. Although it might be difficult to model a human designer on the symbol level, the presented model was developed on the knowledge level. On this level, the model explains such issues as for example, contradictory problem specifications. We also proposed an explanation of what does it mean when a designer ‘is stuck’ with an apparently sound solution in his or her hands.

Second, we proposed a set of principled schemas on the above-mentioned conceptual (knowledge) level with an aim to make the interactive patterns of the design process explicit. These conceptual schemas were elicited from the rigorous experiment that deployed such a structured approach to design. The experimental sessions were analysed in terms of reasoning steps, episodes and contextual ‘milestones’. The analysis confirmed the viability of the recursive and interactive model of framing and the underlying assumptions and conceptual definitions.

According to the participating designers and design experts, the main advantages of the methodological framework became obvious in the situations that required a team collaboration or return to the past design session or design episodes. The structured records of the co-evolving and mutually intertwined design specifications and design solutions were very helpful for a quick familiarisation with the past case. They provided a fertile ground for picking up the important and essential details from the past cases without distracting the designers from their main problem. We may merge this observation with the conclusion from the previous section that argued that a ‘problem framing’ could be seen as an explicit interpretation of the current problem in the familiar vocabularies. Thus, our structured and principled approach to recording and intertwining the specification with the solutions and informal justifications may be used as an abstract guideline for problem framing.

Since the conceptual schemas reflect the sequence of the 'typical' decisions the designer may make during the design process, there is no single, symbol-level method for the implementation of these conceptual patterns. Thus, when one decides to follow the abstract patterns and schemas, this abstract model alone can foster a principled design on the knowledge level. We acknowledge that for the purpose of computer-based support, these abstract schemas need to be turned into operational models and consequently suitable methods. However, such an operational perspective was beyond the time and resource constraints placed on this research, and we shall mention it only as a potential further work, see the discussion in section 12.4.

Taking into account the nature of the contributed operational framework, this research clearly subscribed to the category advocating the knowledgeable design assistants (see also chapter 5). It was not an intention of this work to remove designers from the design loop by degrading their 'insight' into a rigid set of rules. It was not our objective to develop an automated design system, which would be able to deliver creative or innovative results to any vaguely specified problem. On the contrary, the schemas presented in chapter 6 and particularly in chapter 10 have a potential to provide a more informed advice and support than the pure model-based or case-based techniques from the literature. However, as it was emphasised in this thesis, any such advice is heuristic by definition. It may accelerate and clarify the development of an 'insight' but does not guarantee or predict anything.

12.4 Open issues

It is very difficult to judge the ratio between how much has been done and how much could be done in any particular activity. Scholarly research is not an exception. Fortunately, there is always enough of the 'uncharted territory', where smaller or greater discoveries may be accomplished. Such is also the situation with the research described in this thesis. It seemed to be a straightforward modelling exercise at the beginning. Though, as the work progressed, new challenges emerged and new open issues were identified. Some of them were left aside as interesting though not directly relevant to the pursued direction. Others showed to be important and were incorporated into our recursive model of framing.

Thus, using the terminology coined in this thesis, we may say that the explicit 'frame' of this research is still not completely aligned and synchronised with all the tacit ideas and thoughts of the author, his supervisors and other consulted experts. We believe that we achieved an 'acceptable' state in this work, nonetheless, we are aware of its potential extensions. Let us therefore, conclude this thesis with a short vision of the remaining challenges. Again, in line with the proposed theory, some of these 'visions' represent the 'deliberately overlooked' features. Others are more 'speculative ideas' towards the extension and further elaboration of the conducted research. We group the potential extensions and ideas about the future research into two broad categories. First, we mention the pragmatic or practical issues that were left open in sections 12.4.1 to 12.4.3. In sections 12.4.4 and 12.4.5 we look at the scholarly issues that can also become a foundation of a future research.

12.4.1 'Pragmatic' issue 1: RFD model implementation

The primary issue that could be labelled as pragmatic involves a construction of the computational, knowledgeable design assistant that would behave in accordance with the findings expressed in the theory and model of framing on the knowledge level. This is not only an issue of a successful implementation and programming skills, but also an issue of acceptability of such an assisting tool – this topic shall be touched also in section 12.4.3. Let us discuss briefly how the presented RFD model can be implemented and what technologies may be applicable.

When defining the term 'conceptual design frame' in chapter 6, we mentioned that it was an explicit commitment to a particular, usually familiar vocabulary. Familiarity immediately suggests the application of various techniques for the reasoning by analogy. Analogy or familiarity can be recognised between a *base case* (familiar, well-defined previously tackled problem) on one hand, and a *target case* (current, ill-defined problem) on the other hand. According to Gentner and Forbus (1991), there are several different types of familiarities that can be recognised, such as *abstracted analogy*, *literal similarity* or *mere appearance*. Let us look at how these types can be used and how they can help with implementing the RFD model; for additional details of an engine for such reasoning, see (Falkenhainer, Forbus *et al.* 1989).

Abstracted analogy looks for matches between the higher-level relations among the objects in the similar situations (e.g. functions or behaviours), without paying much attention to the low-level details. Since this method recognises familiarity on an abstract level of functions or behaviours, it may discover rather unusual mappings between the two cases. In session T11 for example (section 8.2), the designer changed a mechanical spring into a pneumatic piston. Although the two devices are from different domains, they behave similarly – on a sufficiently abstract level. They both tend to counter the external forces acting on them; these counter-actions increase monotonically with the increasing forces. Thus, a pneumatic piston is in a certain abstract sense, a *functionally alternative* component to the spring. A new structural concept may consequently lead to a frame shift (see sections 6.3.3 and 10.2.3).

The literal similarity is usually justified by the idea that objects 'looking alike may as well act alike' (Falkenhainer, Forbus *et al.* 1989). Thus, we may recognise a literal similarity, for instance between the above-mentioned pneumatic piston (target case) and say, hydraulic press (base case). The analogous descriptive parameters of the two cases may lead for instance, to introducing new constraints or assumptions to the explicit specification of the target case. Such a modification of the designer's knowledge was in our RFD theory modelled as a specification refinement within a single frame (section 10.2.1) or with re-framing (section 10.2.2). Thus, the two methods for the discovery of similarity may complement each other and re-use different knowledge from the base case (structural components for a solution construction as well as functional constraints for a problem specification).

12.4.2 'Pragmatic' issue 2: management of 'inarticulate' knowledge

Another 'pragmatic' issue that has not been sufficiently covered by this thesis concerns the knowledge management in addition to knowledge utilisation. While the agenda for utilising the available knowledge was examined in details, the issue of acquiring and managing knowledge was left aside. We applied a simple method for extracting the relevant knowledge from the formal and informal records entered by a designer during a design session. However, we left aside the issues such as management of such a repository of design cases, 'automated' acquisition, indexing and cross-referencing of the records. All these issue are important for a successful implementation of a design support system that draws so frequently on the informal and 'tacit'

The author has been involved in a research project where similar issues were investigated in the domain of knowledge management in geographically distributed organisations (Dzbor, Paralic *et al.* 2000). The management was easy as long as the document repository was searched using keywords, and those were explicitly present in the document(s). However, the focus of the project was on retrieving ontologically relevant documents; i.e. the 'keywords' may not be present explicitly but some similar concepts may be there instead. This was obviously a 'worse-structured' problem. The system developed during the project tackled this case by a manual and semi-automated annotations of the documents. The latter method was based on various templates designed for 'document types'. Similar approach may be applicable to design – *template-based annotation* of the captured records may be based on the different types of design problems or reasoning patterns emerging in the partially recorded current session. Such annotations then may be deployed in recognising the similarities or re-using the 'ontologically relevant' parts of a past case (see also section 12.4.1 above).

Nevertheless, this issue was tackled in parallel to the research reported in this thesis. The findings reported in this thesis were used as a somewhat theoretical justification or explanation in a research project running at the Knowledge Media Institute funded by the European Commission[17]. Although the author was not directly involved in any of the project work packages, there was a significant exchange of the ideas, especially between more theoretical level of this research and more practical orientation of the project partners. Nonetheless, from pragmatic reasons, it would be very exciting to see how the entire theory presented in this document could be applied to a realistic design support tool.

12.4.3 'Pragmatic' issue 3: HCI and psychology

Another issue emerges from a psychological and HCI perspective. This was already mentioned earlier as a relevant point of view, but we did not address it any deeper. Nevertheless, the issues of providing a 'seamless' assistance to the designer are important for the acceptability of the theory, whether in a form of methodology or tool. In our experiments, we used very simple tools that recorded both formal and informal reasoning. However, much was left aside – for instance, the sketches had to be drawn on paper and then 'inserted' into a flow of justifications by a verbal remark. This required extra effort from the designer. Therefore, it would be very interesting to have all the interactions as modelled by the RFD theory integrated more naturally into a designer's work (incl. sketching on various levels of complexity). The role of such a rich and seamless integration of several different design support tools, electronic notebooks and communication facilities was investigated during the development of the Tadzebao system at KMi (Domingue 1998). A similar extension of the work presented in this thesis may constitute an interesting and fruitful topic for a computer-supported collaborative work (CSCW).

Similarly, we did not investigate the 'psychological' issues of knowledge, such as spreading activation, knowledge reinforcing or forgetting. It should serve as our justification that these issues are quite remote from the main objectives of this research. Nonetheless, they are important for the sake of acceptability of the proposed methodology and model. Furthermore, this perspective may lead to

[17] For further particulars, see the project website on http://clockwork.open.ac.uk.

interesting research, e.g. on user modelling. A truly 'intelligent' support system should be able to learn what is 'typically' acceptable for the particular user/designer. This knowledge may be used to estimate the user's decisions, and pre-filter the choices or sort them accordingly before the designer articulates the need in the explicit terms of a 'command' typed in or spoken.

12.4.4 'Scholarly' issue 1: Broader validation, other domains

One of the main scholarly issues would address the validation of the proposed framework. The model and methodology have been tested with a specific group of participants tackling a specific set of design tasks. It would be interesting to see how the interactive model performs in other design disciplines. A little self-reflection at this point suggests looking at this thesis/research as a problem of designing (ill structure, co-evolving problem specification and solution, many reflective broodings, frame formulations and re-formulations). Thus, we have another 'design case', this time from an academic and scholarly discourse rather than control engineering.

Although there may be some 'art' in doing scholarly research, it is still a relatively well-engineered exercise with one main stakeholder (author) and one underlying theme (PhD thesis). An interesting validation of the proposed model may be achieved by comparing more 'technical' disciplines with the softer, 'artistic' ones. We believe that the principled approach of the RFD model would also apply to the tasks from the 'softer' domains (Schön 1983). By large, problems of architectural or interior design, curriculum design, or problems aiming to support the communities may exhibit many 'artistic' features. The challenges are posed by several stakeholders typically involved in such exercises. The problem structure is even worse than in engineering design. Imagine designing a support system for the communities that are establishing themselves around such subtle topics as history, cultural heritage or philosophy. This may be an interesting and rather unusual extension (and validation) of the formal theory and model presented in this thesis.

12.4.5 'Scholarly' issue 2: Additional patterns recognisable inductively

Lastly we mention another interesting parallel that emerged while analysing and visualising the results of our empirical study. The re-occurring patterns of reasoning in the annotated sessions suggested that perhaps an inductive approach could reveal more and even subtler patterns that could further extend and complement the schemas proposed in chapter 10. Suppose we have a repository of design sessions that can be annotated according to our RFD model (or any of its future extensions). Would an informed inductive learning method be able to discover new, interesting generalised 'rules' or 'schemas' among the specific design problems?

Induction is one logical technique that was not mentioned in section 6.4. Its principle is finding more general commonalties between (usually many) specific examples. Performance of this method may be improved by providing a 'background knowledge' to it (Michalski 1980); in our case, this can be the current version of RFD model. Is it a far-fetched idea to apply methods of pattern recognition and analysis in the setting of design? It may be so, but it may make interesting research, too…

13 REFERENCES AND BIBLIOGRAPHY

ACM (1993). "Special Issue on Participatory Design." Communications of the ACM **36**(6).

Altshuller, G.S. (1984). Creativity as an Exact Science. USA, Gordon & Breach Science Publishers.

Anderson, W.L. and Crocca, W.T. (1993). "Engineering Practice and Co-development of Product Prototypes." Communications of the ACM **36**(6): 49-56.

Asimow, M. (1962). Introduction to design. Prentice-Hall, Inc. Englewood Cliffs, NJ, USA.

Barr, A. and Feigenbaum, E.A., Eds. (1982). The Handbook of Artificial Intelligence. California, USA, William Kaufmann, Inc.

Bhatta, S. and Goel, A. (1994). "Discovery of Physical Principles from Design Experiences." Artificial Intelligence for Engineering, Design, Analysis and Manufacturing (AIEDAM) **8**(2).

Bhatta, S., Goel, A., *et al.* (1994). Innovation in analogical design: A model-based approach. 3rd Intl. Conference on AI in Design (AID'94), Lausanne, Switzerland.

Black, P.E. (1999). Dictionary of Algorithms, Data Structures, and Problems. (accessed 2001).

Borst, W.N. (1997). Construction of engineering ontologies for knowledge sharing and reuse. CTIT PhD. Thesis, University of Twente.

Boyle, J.-M. (1989). "Interactive engineering systems design: a study for artificial intelligence applications." Artificial Intelligence in Engineering **4**(2): 58-69.

Brazier, F.M.T., van Langen, P.H.G., *et al.* (1998). "Strategic Knowledge in Design: a Compositional Approach." Knowledge-based Systems **11**(Special issue): 405-416.

Brazier, F.M.T., van Langen, P.H.G., *et al.* (1996a). "Redesign and reuse in compositional knowledge-based systems." Knowledge-based Systems **9**: 105-118.

Brazier, F.M.T., van Langen, P.H.G., *et al.* (1996b). "Modelling an elevator design task in DESIRE: the VT example." Int. Journal of Human-Computer Studies **44**(3): 469-520.

Brown, D.C. and Birmingham, W.P. (1997). "Understanding the Nature of Design." IEEE Intelligent Systems & their applications **12**: 14-16.

Buckhingham Shum, S. (1996). Design Argumentation as Design Rationale. Encyclopedia of Computer Science and Technology. New York, USA, Marcel Dekker Inc. **35:** 95-128.

Bylander, T. and Chandrasekaran, B. (1985). Understanding Behavior Using Consolidation. 9th Intl. Joint Conference on AI (IJCAI'85), Los Angeles, California.

Candy, L. (1998). "Representations of strategic knowledge in design." Knowledge-Based Systems **11**: 379-390.

Candy, L. and Edmonds, E. (1996). "Creative design of the Lotus bicycle: implications for knowledge support systems research." Design Studies **17**: 71-90.

Chandrasekaran, B. (1990). "Design Problem Solving: A Task Analysis." AI Magazine **11**(4): 59-71.

Chandrasekaran, B. (1993). "Functional Representation: A Brief Historical Perspective." Applied Artificial Intelligence.

Chandrasekaran, B., Josephson, J.R., *et al.* (1998). The Ontology of Tasks and Methods. 11th Banff Knowledge Acquisition for KBS Workshop, Canada.

Chandrasekaran, B., Josephson, J.R., *et al.* (1999). "What Are Ontologies, and Why Do We Need Them." IEEE Intelligent Systems & their applications **14**(1): 20-26.

Chen, X. and van Beek, P. (2001). "Conflict-Directed Backjumping Revisited." Journal of Artificial Intelligence Research **14**: 53-81.

Choueiry, B.Y., McIlraith, S., *et al.* (1998). Thoughts Towards a Practical Theory of Reformulation for Reasoning about Physical Systems. Stanford, USA, Knowledge Systems Laboratory.

Cohen, P.R. and Feigenbaum, E.A., Eds. (1982). The Handbook of Artificial Intelligence. California, USA, William Kaufmann, Inc.

Cook, S.D.N. and Brown, J.S. (1999). "Bridging Epistemologies: The Generative Dance Between Organizational Knowledge and Organizational Knowing." Organization Science **10**(4): 381-400.

Coyne, R.D., Rosenman, M.A., *et al.* (1990). Knowledge-based Design Systems, Adison-Wesley.

Cross, N. (1997). "Descriptive models of creative design: application to an example." Design Studies **18**: 427-440.

Cross, N. (1999). "Natural intelligence in design." Design Studies **20**(1): 25-39.

Davidson, J.E. (1995). The Suddenness of Insight. The Nature of Insight. Sternberg, R.J.and.Davidson, J.E. (eds.) USA, MIT Press**:** 125-155.

Davies, S.P. (1995). "Effects of concurrent verbalization on design problem solving." Design Studies **16**: 102-116.

d'Avila Garcez, A.S., Russo, A., *et al.* (1999). Restructuring requirements specifications for analysis and change management. 16th IEEE Conference on Automated Software Engineering, San Diego, California.

Davis, M. (1980). "The Mathematics of non-monotonic reasoning." Artificial Intelligence **13**(1): 73-80.

de Kleer, J. (1986a). "An Assumption-based TMS." Artificial Intelligence **28**(2): 127-162.

de Kleer, J. (1986b). "Problem Solving with the ATMS." Artificial Intelligence **28**(2): 197-224.

de Kleer, J., Mackworth, A.K., *et al.* (1990). Characterizing Diagnoses. Proceedings of AAAI'1990.

Dechter, R. (1992). Constraint Networks. Encyclopaedia of Artificial Intelligence. Shapiro, S.C., John Wiley & Sons**:** 276-285.

Dienes, Z. and Fahey, R. (1998). "The Role of Implicit Memory in Controlling a Dynamic System." The Quarterly Journal of Experimental Psychology **51A**(3): 593-614.

Domingue, J. (1998). Tadzebao and WebOnto: Discussing, browsing and editing ontologies on the web. 11th Knowledge Acquisition for KBS workshop, Banff, Canada.

Dominowski, R.L. and Dallob, P. (1995). Insight and Problem Solving. The Nature of Insight. Sternberg, R.J. and Davidson, J.E. (eds.) USA, MIT Press**:** 33-62.

Doyle, J. (1979). "A Truth Maintenance System." Artificial Intelligence **12**: 231-272.

Dzbor, M. (1999). Intelligent Support to Problem Formalisation in Design. 3rd IEEE Conference on Intelligent Engineering Systems (INES'99), Stara Lesna, Slovakia.

Dzbor, M. (2000a). Design as a Problem of Requirements Explication. ECAI'2000 Workshop on Knowledge-Based Systems for Model-Based Engineering Design, Berlin, Germany.

Dzbor, M. (2000b). Explication of Design Requirements Through Reflection on Solutions. 4th IEEE Conf. on Knowledge-based Intelligent Engineering Systems, Brighton, UK.

Dzbor, M. (2001). Knowledge re-formulation in design. 5th Conference on Knowledge-based Intelligent Engineering Systems, Osaka, Japan, IOS Press/Ohmsha.

Dzbor, M., J. Paralic, *et al.* (2000). Knowledge Management in a Distributed Organisation. 4th IEEE, IFIP Conference on IT for Balanced System (BASYS'2000), Berlin, Germany, Kluwer Publ.

Dzbor, M. and Zdrahal, Z. (2001a). Towards a Framework for Acquisition of Design Knowledge. 27th Design Automation Conference, (part of ASME DETC, ref# DETC01/DAC-21049), Pittsburgh, USA.

Dzbor, M. and Zdrahal, Z. (2001b). Towards Logical Framework for Sequential Design. 13th Intl. Conf. on Design Theory and Methodology, (part of ASME DETC, ref# DTM-21710), Pittsburgh, USA.

Dzbor, M. and Zdrahal, Z. (2002). Design as interactions between problem framing and problem solving. 15th European Conference on Artificial Intelligence (ECAI'2002), Lyon, France.

Ezekiel, M.L. (2000). Just-In-Time Design in a Fast-Paced Product Group. Designing Interactive Systems Conference, New York, USA.

Falkenhainer, B., Forbus, K., *et al.* (1989). “The Structure Mapping Engine: Algorithm and Examples.” Artificial Intelligence **41**(1): 1-63.

Fischer, G. (1992). Domain-Oriented Design Environments. 7th Knowledge-Based Software Engineering Conference (KBSE'92), IEEE Computer Society.

Fischer, G., Nakakoji, K., *et al.* (1993). Facilitating Collaborative Design Through Representations of Context and Intent. AAAI ’93, AI in collaborative design, Washington.

Fox, M.S. (1994). ISIS: A Retrospective. Intelligent Scheduling. Zweben, M. and Fox, M.S. (eds.), Morgan Kaufmann Publishers, Inc.**:** 3-29.

Garrett, J.H., Jr. (1998). “The computer-aided engineer: Prospects and risks.” Artificial Intelligence for Engineering, Design, Analysis and Manufacturing **12**: 61-63.

Genesereth, M.R. and Nilsson, N.J. (1987). Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publ., USA.

Gentner, D. and K. Forbus (1991). MAC/FAC: A Model of Similarity-based Retrieval. 13th Annual Conference of the Cognitive Science Society, Chicago, USA.

Gero, J. (1990). “Design prototypes: A knowledge representation schema for design.” AI Magazine **11**(4): 26-36.

Gero, J.S. (1996). “Creativity, emergence and evolution in design.” Knowledge-Based Systems **9**: 435-448.

Gero, J. (1998a). “Concept formation in design.” Knowledge-Based Systems **11**: 429-435.

Gero, J.S. (1998b). Conceptual designing as a sequence of situated acts. Artificial Intelligence in Structural Engineering. Smith, I. Berlin, Springer**:** 165-177.

Gero, J.S. (1998c). Towards a model of designing which includes its situatedness. Universal Design Theory. Grabowski, H., Rude, S. and Green, G. (eds.), Aachen, Shaker Verlag**:** 47-56.

Goel, A.K. (1997). “Design, Analogy, and Creativity.” IEEE Expert **12**(3): 62-70.

Goel, V. (1994). “A comparison of design and nondesign problem spaces.” Artificial Intelligence in Engineering **9**(1): 53-72.

Goel, V. (1995). Sketches of Thought. Massachussetts, USA, MIT Press.

Gomez de Silva Garza, A. and Maher, M.L. (1996). “Design by Interactive Exploration Using Memory-Based Techniques.” Knowledge-Based Systems **9**(1).

Groenbak, K., Kyng, M., *et al.* (1993). “CSCW Challlenges: Co-operative Design in Engineering Projects.” Communications of the ACM **36**(6): 67-77.

Gruber, T.R. (1993). “A Translation approach to Portable Ontology Specifications.” Knowledge Acquisition **5**(2): 199-221.

Gruber, T.R. and Iwasaki, Y. (1991). How things work: Knowledge-based modelling of physical devices. Stanford University, California, technical report.

Gruber, T.R. and Russell, D.M. (1991). Design Knowledge and Design Rationale: A Framework for Representation, Capture and Use. Stanford University, California.

Iwasaki, Y., Fikes, R., *et al.* (1993). How Things Are Intended to Work: Capturing Functional Knowledge in Device Design. 13th International Joint Conference on AI, Chambery, France.

Jackson, M. (2001). Problem Frames: Analyzing and structuring software development problems, ACM Press/Addison-Wesley.

Jagodzinski, P., Reid, F.J.M., *et al.* (2000). “A study of electronics engineering design teams.” Design Studies **21**(4): 375-402.

Kensing, F. and Munk-Madsen, A. (1993). “PD: Structure in the Toolbox.” Communications of the ACM **36**(6): 78-85.

Klir, G.J. (1985). Architecture of Systems Problem Solving. New York, USA, Plenum Press.

Kolodner, J.L. and Wills, L.M. (1996). “Powers of observation in creative design.” Design Studies **17**: 385-416.

Kumar, V. (1992). “Algorithms for Constraints Satisfaction Problems: A Survey.” AI Magazine **13**(1): 32-44.

Levesque, H. (1989). A knowledge-level account of abduction. 11th International Joint Conference on Artificial Intelligence, Detroit, USA.

Levin, M. (1974). Mathematical logic for computer scientists. Cambridge, Mass., MIT.

Liu, Y.T. (2000). “Creativity or novelty?” Design Studies **21**(3): 261-276.

Logan, B., Corne, D., *et al.* (1992). The Edinburgh Designer System: An Architecture for Solving Ill Structured Problems. European Conference on Artificial Intelligence (ECAI'92), Vienna, Austria.

Logan, B. and Smithers, T. (1992). Creativity and Design as Exploration, DAI technical report, University of Edinburgh.

MacCallum, K.J. (1990). “Does Intelligent CAD exist?” Artificial Intelligence in Engineering **5**(2): 55-64.

Maher, M.L. (1990). “Process Models for Design Synthesis.” AI Magazine **11**(4): 49-58.

Maher, M.L., Poon, J., *et al.* (1996). Formalising Design Exploration as Co-Evolution: A Combined Gene Approach. Advances in Formal Design Methods for CAD. Gero, J.S. and Sudweeks, F. (eds.), Chapman & Hall**:** 1-28.

McCarthy, J. (1980). “Circumscription - A Form of Nonmonotonic Reasoning.” Artificial Intelligence **13**(1): 27-39.

McCarthy, J. (1986). “Applications of Circumscription to Formalizing Common Sense Knowledge.” Artificial Intelligence **28**: 89-116.

McDermott, D. and Doyle, J. (1980). “Non-Monotonic Logic.” Artificial Intelligence **13**(1-2): 41-72.

Mendelson, E. (1979). Introduction to Mathematical Logic. USA, D. Van Nostrand Co.

Michalski, R.S. (1980). “Pattern Recognition as Rule-Guided Inductive Inference.” IEEE Transactions on Pattern Analysis and Machine Intelligence **PAMI-2**(4).

Miller, S.E. (1993). “From System Design to Democracy.” Communications of the ACM **36**(6).

Motta, E. (1997). Reusable Components for Knowledge Modelling. The Netherlands, IOS Press.

Motta, E., O'Hara, K., *et al.* (1996). “Solving VT in VITAL: A Study in Model Construction and Knowledge Reuse.” Intl. Journal of Human-Computer Studies **44**(3-4).

Motta, E. and Zdrahal, Z. (1996). Parametric Design Problem Solving, KMi technical report, The Open University, UK (*http://kmi.open.ac.uk/publications/techreports.html*).

Motta, E. and Zdrahal, Z. (1998). A principled approach to the construction of a task-specific library of problem solving components. 11th Banff Knowledge Acquisition for KBS Workshop, Canada.

Muller, M. (2002). Participatory Design: The Third Space in HCI. Handbook of HCI. Jacko, J. and Spears, A. (eds.) Minneapolis, Minnessota, (anticipated in 2002).

Muller, M. and Carey, K.D. (2002). Design as a minority discipline in a software company: Toward requirements for a community of practice. Computer-human interaction conference (SIGCHI), Minneapolis, Minnessota.

Mylopoulos, J. and Levesque, H.J. (1984). An overview of knowledge representation. On Conceptual Modelling. Brodie, M.L., Mylopoulos, J. and Schmidt, J.W. (eds.), New York, USA, Springer Verlag**:** 3-17.

Nakakoji, K., Sumner, T., *et al.* (1994). Perspective-based critiquing: helping designers cope with conflicts among design intentions. 3rd Conference on AI in Design (AID'94), Lausanne, Switzerland.

Nakakoji, K., Yamamoto, Y., *et al.* (1998). "From critiquing to representational talkback: Computer support for revealing features in design." Knowledge-Based Systems **11**: 457-468.

Newell, A. (1982). "The knowledge level." Artificial Intelligence **18**(1): 87-127.

Newell, A. and Simon, H. (1976). "Computer Science as Empirical Inquiry: Symbols and Search." Communications of the ACM **19**(3): 113-126.

Nidamarthi, S., Chakrabarti, A., *et al.* (1997). The significance of co-evolving requirements and solutions in the design process. 11th Int. Conference on Engineering Design (ICED), Finland.

Nuseibeh, B. (2001). "Weaving together requirements and architectures." IEEE Computer **34**(3): 115-117.

Nuseibeh, B. and Easterbrook, S. (2000). Requirements Engineering: A Roadmap. International Conference on Software Engineering, Limerick, Ireland.

Oxman, R. (1990). "Design shells: a formalism for prototype refinement in knowledge-based design systems." Artificial Intelligence in Engineering **5**(1): 2-8.

Poole, D. (1989a). "Explanation and prediction: an architecture for default and abductive reasoning." Computational Intelligence **5**(2): 97-110.

Poole, D. (1989b). Normality and Faults in Logic-Based Diagnosis. 11th International Joint Conference on Artificial Intelligence, Detroit, USA.

Poole, D. (1990). "A Methodology for Using a Default and Abductive Reasoning System." Intl. Journal of Intelligent Systems **5**(5): 521-548.

Prabhakar, S. and Goel, A. (1998). "Functional modeling for enabling adaptive design for new environments." Artificial Intelligence in Engineering **12**: 417-444.

Price, C. and Lee, M. (1988). "Applications of deep knowledge." Artificial Intelligence in Engineering **3**(1): 12-17.

Qian, L. and Gero, J.S. (1996). "Function-Behaviour-Structure Paths and Their Role in Analogy-Based Design." Artificial Intelligence for Engineering, Design, Analysis and Manufacturing **10**: 289-312.

Reddy, M.J. (1988). The Conduit Metaphor - A Case of Frame Conflict in Our Language about Language. Metaphor and Thought. Ortony, A., Cambridge University Press**:** 284-324.

Reiter, R. (1980). "A Logic for Default Reasoning." Artificial Intelligence **13**(1): 81-132.

Reiter, R. (1987). "A Theory of Diagnosis from First Principles." Artificial Intelligence **32**(1): 57-95.

Riesbeck, C. and Schank, R. (1989). Inside Case-based Reasoning. New Jersey, Lawrence Erlbaum.

Rose, A., Shneiderman, B., *et al.* (1995). An Applied Ethnographic Method for Redesigning User Interfaces, HCIL technical report, University of Maryland, USA.

Russo, A., Nuseibeh, B., *et al.* (1999). "Restructuring requirements specifications for analysis and change management." IEE Proceedings: Software **146**(1): 44-53.

Sabin, D. and Freuder, E.C. (1994). Contradicting Conventional Wisdom in Constraint Satisfaction. 2nd Intl. Workshop on Principles and Practice of Constraint Programming, Washington, USA.

Sara, R., Svec, M., *et al.* (2001). Texture Analysis of Sonographic Images for Hashimoto's Lymphocytic Thyroiditis Recognition.

Schön, D.A. (1983). Reflective Practitioner - How professionals think in action. USA, Basic Books, Inc

Schön, D.A. (1995). "Knowing-in-action: The new scholarship requires a new epistemology." Change - magazine of higher learning **27**(November/December 1995): 27-34.

Shulyak, L. (1999). Introduction to TRIZ. (accessed 1999, *URL: http://www.aitriz.org/Downloads/40Ptriz.pdf*).

Simon, H.A. (1969). The Sciences of the Artificial. Cambridge, Massachusetts, MIT Press.

Simon, H.A. (1973). “The structure of ill-structured problems.” Artificial Intelligence **4**: 181-201.

Smithers, T., Conkie, A., *et al.* (1990). “Design as intelligent behaviour: an AI in design research programme.” Artificial Intelligence in Engineering **5**(2): 78-109.

Smutek, D., Tjahjadi, T., *et al.* (2001). Image Texture Analysis of Sonograms in Chronic Inflammations of Thyroid Gland, Czech Technical University research report.

Stefik, M. (1995). Introduction to Knowledge Systems. California, Morgan Kaufmann.

Steinberg, L. (1994). “Research methodology for AI and design.” Artificial Intelligence for Engineering Design, Analysis, and Manufacturing **8**: 283-287.

Sumner, T., Bonnardel, N., *et al.* (1997). The cognitive ergonomics of knowledge-based design support system. Computer-Human Interfaces conference (SIGCHI'1997), Atlanta, Georgia.

Sumner, T. and Buckingham Shum, S. (1998). From Documents to Discourse: Shifting Conceptions of Scholarly Publishing. Computer-Human Interaction conference (SIGCHI'1998), Los Angeles, California.

Sushkov, V.V., Mars, N.J.I., *et al.* (1995). “Introduction to TIPS: a theory for creative design.” Artificial Intelligence in Engineering **9**(2): 177-189.

Swartout, W. and Balzer, R. (1982). “On the Inevitable Intertwining of Specification and Implementation.” Communications of the ACM **25**(7): 438-440.

Takeda, H. and Nishida, T. (1994). Integration of aspects of design processes. 3rd Conference on AI in Design (AID'94), Lausanne, Switzerland.

Tang, M. (1997). “A knowledge-based architecture for intelligent design support.” Knowledge Engineering Review **12**(4): 387-406.

Tansley, D.S.W. and Hayball, C.C. (1993). Knowledge-based systems analysis and design, Prentice Hall International (UK) Ltd.

Thompson, G. and Lordan, M. (1999). “A review of creativity principles applied to engineering design.” Journal of Process Mechanical Engineering **213**(1): 17-31(15).

Tomiyama, T. (1994). “From general design theory to knowledge-intensive engineering.” Artificial Intelligence for Engineering Design, Analysis, and Manufacturing **8**: 319-333.

Tufte, E. (2001). Presenting Data and Information, Class notes by T. Miler. (accessed 2002*, URL: http://www.oreillynet.com/~terrie/tufte/*).

Ullman, D.G. (1997). The Mechanical Design Process, McGraw-Hill, Inc.

van Heijst, G., Schreiber, A.T., *et al.* (1997). “Using explicit ontologies in KBS development.” International Journal of Human-Computer Studies **46**(2/3): 183-292.

Watson, I. and Perera, S. (1997). “Case-based design: A review and analysis of building design applications.” Artificial Intelligence for Engineering, Design, Analysis and Manufacturing **11**(1): 59-87

Wevers, J.C.A. (1999). Physics formulary (*URL: http://www.xs4all.nl/~johanw/physics.pdf*).

Wielinga, B., Akkermans, J.M., *et al.* (1995). A Formal Analysis of Parametric Design Problem Solving. 9th Knowledge Acquisition for KBS Workshop, Banff, Canada.

Wielinga, B. and Schreiber, G. (1997). “Configuration-Design Problem Solving.” IEEE Journal on Intelligent Systems & Their Applications **12**(2): 49-56.

Yost, G.R. (1992). Configuring elevator systems, AI Research Group, Digital Equipment Corporation.

Zdrahal, Z., *et al.* (1997). DCCs Truck Cabin Design. ENCODE Research Project, (accessed 1999, *URL: http://kmi.open.ac.uk/~zdenek/dccs-truck.html*).

Zweben, M. and Fox, M.S. (1994). Intelligent Scheduling, Morgan Kaufmann Publishers, Inc.

Doctor of Philosophy Thesis

Appendices

Design as interactions of problem framing and problem solving

(A formal and empirical basis for problem framing in design)

Martin Džbor

Knowledge Media Institute
The Open University
United Kingdom

Supervised by: *Dr. Zdeněk Zdráhal*
Dr. John B. Domingue
Examined by: *Prof. Bashar A. Nuseibeh* (The Open University)
Dr. David Robertson (AIAI, University of Edinburgh)

February 1999 – June 2002

These appendices and the CD-ROM are an integral part of the thesis

TABLE OF CONTENTS

1 BRIEF SUMMARY OF EXPERIMENTAL SESSIONS

Each of the experimental sessions was given a unique identifier (e.g. *T21*) that is also used as an abbreviated reference throughout this thesis. In addition to a formal identifier, each session is named using the main artefact that was to be controlled/designed. In the abbreviated descriptions of the sessions, we look at the following 'characteristics' of the problems:

- expected primary goal(s);
- essential milestones of the designer's approach;
- quick description of the problem's character;

T2 … Robot arm lifting a box[1]

Primary goals:

- design of robot arm structure (incl. dimensions, joints, drives);
- consider the suitable means for lifting a box of given shape/size on a specified shelf;
- control of a smooth movement of an arm with a box (box cannot fall and tumble, maximal elevation angles, speeds, etc.)

Essential milestones:

- designer begins with a simple arm containing one joint, later a second joint is added to increase flexibility and stability of a box;
- task of lifting is divided in two parts – first lift to a suitable height, then shift a box on a shelf;
- constraints are formulated on minimal distances between the rows of shelves needed for arm manoeuvring and safety;
- smoothness is attended in a simplified manner; i.e. to avoid complexity of the control circuitry, designer uses so-called step motors and a sequence of touch-sensors and incremental sensors

Problem character:

- the task is well described, and focuses on the selection of suitable variation of well-known building blocks and algorithms; there are prototypes available;
- it is a problem tackled on a medium to low conceptual levels, with a formulation of basic principles for movement control;
- designer was required to suggest both, a structure of an arm together with a control strategy

T3 … Balancing a ball on a beam

Primary goals:

- design of a method for balancing a vertical beam fixed in its gravitational centre so that a ball rolling in the beam guiderails is 'stabilised' exactly in the middle of the beam;
- suggestion for a method of measuring the actual position of a ball;

[1] Task numbers range from 2 to 25; task number 1 was used for presentation purposes and familiarisation with the available design capture toolkit.

- smooth and sufficiently sensitive controller

Essential milestones:

- designer begins with a routine approach as learned in a textbook case – resistance-based measurement of ball's location on a beam;
- articulation of basic assumptions – metallic ball and guiderail (to conduct electric current and enable measuring the resistance);
- introduction of a step motor connected to the beam off its centre via a single-arm shaft;
- shaft design modified to a two-armed in order to deliver smoother regulatory actions;
- introduction of an alternative with a 'generic' ball and guiderail (other, non-conductive materials considered);
- re-formulation of the original constraint on conductive materials to suit the new, more generic approach to position measurement (ball acts as a switch pressing a thin wire against a plate thus creating an electric circuit for resistance measurement)

Problem character:

- initially, well-structured routine task that changes to an innovative when the designer becomes aware of the limitations of the textbook solution;
- structure of the controlled assembly is well known, the available means for controller design as well; ambiguity arises with a method for position measurement (for generic cases);
- knowledge transfer and problem re-formulation occur in order to adapt a simple measurement technique for changed conditions (conductive vs. non-conductive materials)

<u>T4 … 2D gantry crane</u>

Primary goals:

- design of a control strategy for a crane movement in two dimensions (left-right, up-down);
- refinements of the simple control between the extreme points to address swaying
- need for measuring the actual position, angle of the transported load

Essential milestones:

- the structure of a crane is known, designer adds the sensors monitoring the extreme position of a crane on the vertical rail;
- to ensure smooth movement, a sequence of sensors is introduced to precede the extreme points; when the crane passes the individual sensors, it is automatically slowed down;
- clarification and simplification of the lifting assembly, introduced an assumption of working with a rod of given length and stiffness with a hook at its end;
- introduction of constraints regarding swaying of a load during transport; followed by a major modification to the approach to address the issue of swaying;
- smooth one-directional movement replaced by an oscillatory shifting that countered swaying

Problem character:

- original, relatively straightforward task of controlling smoothness of crane movement changed to a practical level taking into account safety and efficiency (load swaying);

- thus, problem re-formulation occurs and changes the task from ‘movement control’ to ‘safe movement without swaying’;
- task is not routine but it is neither inventive – practical manageability is achieved by simplifying assumptions in respect to lifting mechanism
- problem is approached by decomposition (first, vertical crane movement, then anti-sway)

T5 … Simple airconditiong in a room

Primary goals:

- design of an algorithm governing the behaviour of an A/C unit of given structure for a room of a particular shape/size;
- adaptation of a prototypic solution typical for thermal regulation for a particular case

Essential milestones:

- in the given structure of an A/C unit, the controlled and controlling parameters were distinguished (e.g. room temperature as a variable controlled by rotational velocity of a fan);
- proposal of a suitable system of locations for a reliable measurement of room temperature (incl. transformation of measured values in suitable form for a controller);
- decomposition of a problem to separate sub-problems with controlling each major technical module of A/C unit (heater, fan, flaps);
- prototypic approach using a two-level step control for switching the heaters on/off; later refined to include a ‘region of insensitivity’ in order to prevent frequent on/off changes;
- co-ordination of the individual sub-controllers and subordinate control strategies

Problem character:

- medium to high conceptual level; the solution consists of a series of sketches and schemes;
- routinely formulated task is refined to incorporate non-routine details (e.g. co-ordination)

T6 … 3D gantry crane

Primary goals:

- design of a regulation strategy for a crane movement in 3D (left-right, up-down, forward-backward), with a cable-based lifting mechanism;
- partial sub-problems include the control of the vertical movement, lifting of a load, antisway;

Essential milestones:

- analogy with the existing solution of a 2D gantry crane (see task T4)[2], its retrieval;
- familiarisation with task T4, decision to re-use the rough structure and the approach; however, several particulars identified for the current case (e.g. a cable instead of rigid rod for lifting);
- per-partes approach – first, design of a strategy for movement control, later enhanced by a sub-procedure controlling the lifting and height measurement, and eventually, an antisway strategy;
- formulation of a new (innovative) requirement to measure the length of the unwound cable and the weight of a load simultaneously, without additional mechanisms

Problem character:

- prototypic approach known and documented both semi-formally and informally;
- refinement of a known structure of the controlled system, and introduction of less usual extensions (e.g. height measurement);
- task cannot be decomposed into standalone, independent modules – step-by-step refinement and enhancement of a prototypic solution

T7 … Quaker oats blender

Primary goals:

- improvement of an existing strategy controlling an oat blender behaviour;
- selection of suitable (and sufficient) controlled and controlling parameters from a large set;
- design of a control strategy for a non-linear process that is hard to describe precisely;
- design of sub-controllers for the identified sets of parameters; complemented by an 'intelligent' master controller to achieve reliability, robustness, and efficiency of control

Essential milestones:

- familiarisation with an existing control strategy based on the common, standalone PID controllers;
- identification of gaps in the current approach (non-linearity, parameters measured in a location that is rather far from the place where a control action may occur, etc.);
- partial simplification of a complex technological process (e.g. thermal losses through walls shall be neglected, only heat propagation taken into account, etc.);
- design of simple controllers for each identified parameter (i.e. steam temperature in a blender, velocity of a conveyer, volume of oats on input);
- identification of insufficient performance (i.e. difference between theoretically predicted and actually measured and required values);
- new control layer introduced – a master controller handling the situations that are exceptional in respect to the assumptions and limitations of simple controllers (fuzzy-logic rules);

Problem character:

- design of a controller for a typical (prototypic) behaviour, extended to include expected and predicted behaviour;
- hierarchically superior control level reflects the vagueness of problem specification (incl. 'fuzziness' of the formalism for expressing control strategy on this level);
- rather routine task of a process control becomes innovative in respect to an advanced multi-level control taking advantage of 'new conceptual knowledge' (fuzzy rules)

T8 … Oven for ceramic materials

Primary goals:

- design a control strategy that would follow a given curve for production of a specific material;

[2] Two different designers were solving tasks T4 and T6. Therefore, any 're-use' in T6 had to rely on the explicitly

- ensure the given curve is followed with certain precision, i.e. introduce a simple ‘prediction’;
- design a mechanism for controlling oven temperature in a non-linear environment

Essential milestones:

- design of a control strategy lies in as precise curve tracking as possible, selected a two-level step controller (almost prototypic for thermal systems) with a simple feedback (temperature measured inside oven);
- identification of potential drawbacks of a feedback-based control – long reaction time, little flexibility, reduced precision if sharp changes in firing curve;
- introduction of a dynamic control based not only on the measured feedback but also historical data (trends) and prediction of a probable behaviour

Problem character:

- conceptual design of a robust firing oven, started as relatively routine task;
- refinement of the initial specification to focus on high precision of tracking;
- partly innovative extension featuring prediction of behaviour based on the historical data (typical e.g. in economics, statistical systems, etc.)

<u>T9 … Acid de-activation in a sewage plant</u>

Primary goals:

- design of a control strategy for a sewage basin where in-flowing acid is de-activated whilst passing through layers of active elements, output is formed by a ‘harmless’ liquid;
- management of in-flow / out-flow so that a desired volume is maintained in a basin;
- extension of simple diffusion control to degrading active elements (variable de-activation) and variable concentration on the input

Essential milestones:

- main principle is well-defined, nonetheless, designer begins with a simpler solution (simpler problem set-up) and restrictive assumptions (constant concentration, ideal active elements);
- prescriptive in- and out-flow management extended so that the plant performs more effectively (e.g. added stirring control, measurement of concentration, etc.)
- initial restrictive assumptions lifted to give way to more realistic solutions;
- a large number of influencing variables – need to choose the sufficient conditions and control components to keep the solution reasonably simple but robust and flexible;

Problem character:

- initially very simple, prototypic algorithm becomes through numerous refinements more robust and dynamic (to suit such a large scale system as a sewage plant);
- simple and severely limited algorithm is not discarded but rather extended and refined in a sequence of steps in which problem specification is getting more and more realistic

recorded statements made by a colleague in a previous session (T4).

<u>T10 … Mailings sorter</u>

Primary goals:

- design a conceptual model of a mail sorting machine recognising and handling two types of mailings, letters and packages;
- introduction of a simple character recognition technique (to handle postcodes);
- consideration given to 'marginal' problems (full sacks with sorted mail, full sacks removal, repetition of postcode recognition, handling of too large/heavy mailings, etc.)

Essential milestones:

- designer decides to work with simple mechanical devices rather than electronic (conditions dictated by a demanding environment);
- simple differentiation between standard letters and other mailings (a conveyor passing through an opening/sequence of openings with standard sizes);
- incorporation of a postcode recognition 'camera' to the size checker – mailings pass the gap one in time, which is a desired condition for recognition (unexpected but useful feature);
- unrecognised mailings physically flipped and fed back on a side conveyor to repeat the process of recognition or left for a manual sorting;
- the principle of a sorter is a 'tree-like' branching with a sack at the end of each branch ('chute')

Problem character:

- rough sketch of a controlling mechanism, refined in a sequence of consecutive steps;
- algorithm is relatively simple and easy-to-extend further, if additional requirements emerge;
- very simple objects and components used and connecting in a non-trivial and less usual manner; partly innovative character emerges with a series of unexpected but beneficial features

<u>T11 … Active shock absorber</u> (see also annotation and RFD analysis in sections 8.2 and 8.3)

Primary goals:

- design of a shock-absorbing device adapting to the changing road conditions and driving styles
- measurement of disturbances caused by uneven road surface;
- selection of applicable devices and respective controlled/controlling parameters (spring vs. pneumatic tubes, hybrid combination);

Essential milestones:

- early at the beginning, two different modes of operations considered – manual and automated adjustment of the shock dampening characteristics;
- problem attended in two parallel 'contextual worlds', the difference between them growing with a refined solution;
- selection of a pneumatic approach where dampening characteristic depends on pressure rather than the actual material of the absorber (as it was the case with a common spring);
- a set of assumptions formulated to clarify the requirement of an 'active adjustment of chassis clearance' that forms a part of initial problem specification ('active' in the context of chassis clearance is interpreted differently as in the context of shock absorption);
- eventually, a compromise considered and different contexts 'merged'

Problem character:

- task is clearly solved in several separate context (manual vs. automated tuning, shock absorption vs. clearance) that are highlighted by totally different physical principles and components suggested in each particular case;
- presence of multiple contexts resolved by separate controllers for specific situations with a hierarchically superior 'co-ordinator' (a switchboard activating an appropriate sub-module)

T12 … Distillation column

Primary goals:

- design a mechanism for segregating different chemical substances produced in a column (e.g. tar, petrol, oil, kerosene, etc.);
- re-formulation tackles a problem of a precise temperature control/maintenance in particular locations of a column that prevents different substances mixing together;

Essential milestones:

- interpretation of the problem in terms of thermal systems (rather than chemical);
- task changes to tracking/maintaining a given temperature curve at all desired locations;
- determination of extreme values, controlled and controlling variables, and deployment of a simple feedback-based controller

Problem character:

- conceptual design of a complex system control (many parameters, tight relationships, high precision of the regulation);
- after initial interpretation, task is well-defined, only minor clarifications are needed in the subsequent process
- solution comprising more-less standard approaches (prototypic solutions)

T13 … Simple paper production

Primary goals:

- design of a plant for achieving a vaguely defined functionality (smoothing raw paper);
- refinement of a vague specification with various quality- and process-related requirements (thickness reduction, prevention of paper damage, simple maintainability, etc.)

Essential milestones:

- early solution features a pair of rolling drums applying pressure on paper thus reducing its thickness and smoothing its uneven surface;
- in order to comply with qualitative concerns, a pre- and post-processing units added (dampen paper before rolling and dry it afterwards – lower danger of tearing);
- re-formulation of a linear layout of a solution to an alternate (zigzag rolling drums)

Problem character:

- task is relatively routine (with minor refinements of known prototypes);
- simultaneous design of plant layout/structure and appropriate control strategy;
- refinement of a vague specification and solution development

<u>T14 … Steel sheet rolling and coating</u>

Primary goals:

- a slab of metal on the input needs to be rolled into a desired thickness and coated in an electrolytic basin so that a uniform layer of protective film is applied to its surface;
- need for an efficient strategy for planning a sequence of operation with a different thickness is introduced;
- design of an adjustment of the worn-out rolling segments, when abrasion reaches certain level;
- control strategy maintaining uniform concentration/temperature of the electrolytic bath

Essential milestones:

- task is clearly split in two consecutive operations attended separately (rolling vs. coating);
- sheet rolling introduces an unexpected (and 'unrelated') issue of degradation of the rolling segments – additional requirements are formulated;
- degradation of rolling segments introduces an issue of planning (i.e. how much can be rolled with the current degree of abrasion, what is the next batch, etc.);
- control of electrolytic coating is stripped down to the maintenance of required concentration, temperature and uniformity of the bath

Problem character:

- relatively simple set of problems, made complex by numerous interactions and unexpected consequences of thickness reduction or rolled sheet immersion to electrolytic bath;
- newly articulated criteria on the control strategy (efficiency, adjustment of rolling segments);
- very strict decomposition in two distinct sub-problems that have independent solutions;
- initially simple problem specification made more complex (realistic) during its translation to an engineer's language (e.g. coating = uniformity + concentration + temperature)

<u>T15 … Intelligent house monitor and guard</u>

Primary goals:

- structure of a security monitoring system for given premises;
- design of individual sub-circuits for specified parts of the house plus their co-ordination;

Essential milestones:

- formulation of basic requirements in engineering language (e.g. integrity = movement sensors + glass breaking sensors + tactile sensors);
- one circuit is refined into details and a controller designed for it, this solution is then re-applied to remaining security circuits;
- distributed sub-circuits function as standalone controllers that are in certain circumstances co-ordinated and their operation is monitored/evaluated by a 'computer';
- addition of automated recognition of 'permitted objects' entering the premises (chip-based signal recognition) and their authorisation;

Problem character:

- rather routine task made interesting by a translation of vague initial specification into engineering language;

- design of a distributed control of separate, loosely connected sub-modules

<u>T16 … Dual elevator system in a building</u>

Primary goals:

- two elevators controllable from the cabins and the floors – serve the captured requests in an efficient manner (e.g. collect users requesting similar directions, stop if cabin not full, etc.);
- what initially seemed to be a task of elevator design translated to a problem of planning, the main concept becomes a queue of requests from elevator users;

Essential milestones:

- prototypic model of an elevator helps define basic safety features and parameters that must be observed;
- once elevator-specific features are handled, the problem is shifted to the management of requests coming asynchronously, from multiple locations;
- several interesting features considered to refine the planning algorithm (e.g. recognition of a full cabin forces elevator to continue its travel and ignore otherwise relevant request)

Problem character:

- initially taken a traditional approach trying to control a moving object in line with a norm;
- later re-framed as a problem of request management (i.e. information processing problem);
- rather routine techniques used for a broad sketch of solution, refined with several innovative and less usual details transferred from different domains (e.g. optical recognition of full cabin)

<u>T17 … PhotoLab</u>

Primary goals:

- automation of a film development and photograph production process based on given parameters (film sensitivity, type of photographic paper, photograph size);

Essential milestones:

- transfer of a manual procedure for photograph production to an automated sequence of basic operations (develop film, wash it, apply fixative, dry it; the same for photographs);
- separation of film development from photograph processing – two loosely connected modules (co-ordinating parameters are e.g. customer's number, chemical process number, etc.);
- introduction of a planner to process films with similar parameters in one operation → need to manage several queues of incoming and outgoing film rolls (achieved higher quality)

Problem character:

- conceptual design on the level of processing modules, no implementation details;
- decomposition in two separate sub-systems with a co-ordinating higher-level unit (manager);
- application of a routine solution and its automation ('translation' to a different perspective)

<u>T18 … Multi-storey parking management</u>

Primary goals:

- design of a 'complete' solution for managing a multi-storey parking lot (incl. indication of a full place, parking time tracking and payments, navigation to empty bays, etc.);

- formulation of additional goals improving the existing approaches (e.g. easier payments, real-time updates and navigation, security merged with monitoring)

Essential milestones:

- each of the identified goals attended as a separate sub-problem;
- when similar technical approaches emerge, designer merges seemingly different goals into a single module (e.g. time tracking, payments, navigation are basically different forms of the same information);
- design of a procedure for entrance management (ticket issue, record, info update);
- re-use of the entrance procedure also for exit, with a changed order of basic operations and replacement of ticket issue with ticket acceptance and payment;
- utilisation of security cameras and sensors also for the indication of current capacity;
- co-ordination of inputs from entrance/exit ramps and sensors providing updated information and rough navigation to level where empty place are still available

Problem character:

- relatively simple, everyday task approached by an application of prototypic solutions for each sub-problem (entrance, exit, payment, etc.);
- initial loose decomposition becomes tighter towards the end; different units basically access the same information repository and influence each other in a range of channels;
- innovative details proposed to improve safety, efficiency and convenience of operation

T19 … Network proxy server

Primary goals:

- design of a structural model of a device for network access management;
- articulation of critical requirements for network management in engineering language;

Essential milestones:

- overall requirement of managing network connections towards an external world, is translated to a sequence of engineering goals (i.e. filtering, cache, access management, etc.);
- modular architecture and design of a strategy for achieving each of the identified goals;
- network management re-formulated to a problem of efficient execution of the individual operations (some of them are dependent on others, and may re-use the available results);
- no central co-ordinating unit needed, all modules access a database of internal client computers directly; once designed, the sequence of operations is static for all connections

Problem character:

- task is interesting in the translation of a broad goal to a sequence of subordinate, technical requirements and their ordering;
- routine modules for the individual operations, small hints of innovation emerge in the attempts to merge certain operations in a single module (e.g. cache + filtering);
- problem is well decomposable with modules being mutually relatively independent

<u>T20 … Dry materials blender and mixer</u>

Primary goals:

- design of an overall strategy given a desired functionality and basic structural modules;
- merger of individual units to a 'plant' controlled on a basis of input/output parameters;

Essential milestones:

- identification of parameters suitable for a control of a technological process in a given functionality and available structural units;
- translation of a task to a problem of communication management among multiple sources of technological signals (new conceptual frame: buffer, queue, FIFO approach, signal matrix);
- differentiation of preparation of a desired mix within a single batch from batch management;
- two hierarchical levels of control (technological process and batch planning);
- strategic design refined to a modular sketch and finally, design of plant details (e.g. weighing of carts with a material mix, mixing)

Problem character:

- innovative utilisation of the knowledge of parallel information processing;
- interesting is an atypical interpretation of the problem – instead of controlling a purely technological process, management of signals coming from multiple sources is proposed;
- non-traditional transfer of knowledge from unexpected (and rather distant) domains (e.g. IT);
- conceptual design on several levels of abstraction (broad strategy, modular layout, details);
- re-use of knowledge from functionally similar problems (e.g. *T9* – material blender seen as a sewage basin → similar sensors required for level indication, stirring, etc.)

<u>T21 … Paper smoothing plant</u> (see also annotation and analysis in sections 8.3 and 8.4)

Primary goals:

- design of a plant for achieving a vaguely defined functionality (smoothing raw paper);
- refinement of a vague specification with various quality- and process-related requirements (thickness reduction, prevention of paper damage, simple maintainability, etc.);
- selection of a suitable physical principle and subsequently layout of a plant

Essential milestones:

- early solution features a pair of rolling drums applying pressure on paper thus reducing its thickness and smoothing its uneven surface;
- in order to comply with qualitative concerns, a pre- and post-processing units added (dampen paper before rolling and dry it afterwards – lower danger of tearing);
- articulation of restrictive assumptions and additional constraints (e.g. regarding dimensions);
- re-formulation of a linear layout of a solution to an alternate (zigzag rolling drums);
- eventually, principle of rolling is considered in a broader context (re-interpretation of rolling from pressure application to abrasion → instead of drum pairs a zigzag layout of single rolls is sufficient for rolling);
- incorporation of pre- and post-processing units into feeding and driving units

Problem character:

- task starts relatively routinely, eventually becomes innovative (a sequence of re-formulations);
- simultaneous design of plant layout/structure and appropriate control strategy;
- well-visible oscillation between refinement of a vague specification and solution development;
- initially simple solution is getting more complex as additional constraints and assumptions are considered; after re-interpretation the complexity is radically reduced

<u>T22 … Robust car speed measurement</u>

Primary goals:

- given functionality is rather well-known but still hard to be described precisely → articulation of primary engineering goals and solution strategies for a robust speed control;
- design of an automated speedometer – from measuring the speed, through evaluation and recording, ending with driver identification and ticketing

Essential milestones:

- familiarisation with available techniques for speed monitoring, identification of major shortcomings (e.g. immediate speed vs. average speed over certain distance);
- need to identify a car – a problem for pattern recognition algorithms and modules;
- re-use of an available technology for current speed measurement and introduction of a higher processing level calculating average – additional requirements (e.g. remember records for each passing car for a certain time);
- comparison of calculated average speed with an estimate for a given track (if the average is significantly higher, the car was speeding – ticket issue)

Problem character:

- technological level is rather routine and prototypic (immediate speed, average, pattern recognition, etc.);
- consideration of a multilevel structure of the system with multiple related controllers brings an innovative aspect to the problem;
- design conducted on the level of modules, without implementation details

<u>T23 … Bark remover (saw mill pre-processor)</u>

Primary goals:

- design of a plant achieving well-defined functionality and criteria of quality (low waste);
- automation of a bark removal process for a wide range of trunk lengths, shapes, surfaces

Essential milestones:

- selection of an applicable technological principle (incl. position of a milling knife);
- introduction of an adaptive behaviour, the knife elevation and pressure may change during processing a single trunk (due to its uneven surface);
- unexpected consequence of allowing uneven trunks – the side walls holding trunk in position have to adapt to a changing shape of trunk;
- tactile sensor introduced to precede the knife and notify the approaching changes in shape;

- in order to improve quality, a simple predictive mechanism based on the sensor proposed to feature adaptation of the milling knife

Problem character:

- conceptual design of a detailed structure of a solution;
- a significant time devoted to analysis of different principles for bark removal;
- extension of initial specification to comply with tacit requirements on efficiency (introduction of a predictive sensor, for example)

T24 … Tree trunk saw mill[3]

Primary goals:

- design of a plant for sawing a single trunk into boards of given thickness;
- algorithm must comply with a criterion of producing minimal waste (leftovers from sawing)

Essential milestones:

- consideration of different applicable sawing mechanisms (circular saw vs. ribbon saw);
- consideration of different moving mechanisms (fixed console + moving saw, or vice-versa);
- articulation of additional assumptions enabling the choice of a 'better' alternative (seems to be easier to move a console and make a saw more massive = static);
- introduction of additional functionality of trunk flipping to achieve a minimal possible waste;
- addition of means for trunk centring, flexible side walls holding the trunk, etc.

Problem character:

- stepwise refinement of an initial modular design (saw + console) to detailed structures and control strategies for each module;
- basic principle of sawing is almost prototypic but its extension in the aspect of trunk shifting, flipping, etc. is rather innovative and unusual in practice;
- a strong role of analysis – consideration of several alternative approaches and a need to articulate additional conditions for choosing a 'more acceptable' one

T25 … Energy distribution management

Primary goals:

- design of a control strategy for an optimal management of a set of different power sources and power consumers;
- focus given to prediction and flexibility of energy distribution in an area with given resources

Essential milestones:

- familiarisation with a (simplified) process of energy distribution;
- consideration of different sources of energy and their typical characteristics in respect to flexibility, reaction time in an exceptional situation, etc.;
- analysis of the suitability of different power sources for energy production and consumption (this is needed in case there is a surplus energy produced, to store it for future);

[3] This task is a continuation of *T23* – bark removal plant. In practice, both plants would form a sawmill.

- in order to increase the flexibility of the network, designer suggests using water dams and power stations as a dual element that can easily switch from energy production to consumption;
- due to high reaction time of energy networks, the importance of prediction and historical data is emphasised;
- prediction considered in the aspects of easily recorded energy consumption in a given (time of a day, day of a week, season, temperature range, and similarly)

Problem character:

- the task is difficult to solve, mainly due to a large number of interacting parameters of a system and a long reaction time → reason for introduction of additional requirements in prediction;
- in-depth analysis of the available power supplies and their parameters, selection of a set of 'sufficient' parameters for controlling/predicting the network loads;
- introduction of a new conceptual power source to account for sub-optimal prediction and unexpected situations;
- this is a typical task with a vague specification of 'admissibility' and 'acceptability' criteria, functionality is rather well known but is too complex to attend to all its aspects

2 EXAMPLE OF A WELL-STRUCTURED PROBLEM

In accordance to the reviewed design literature, we argued that most of the design tasks in the real-world practice are not well-structured, unless they are severely restricted (Gero 1990; Goel 1994). However, there are numerous studies and design tools for these well-structured problems. In this appendix, we show how a well-defined problem can be modelled on the knowledge level using the operations of abduction, deduction and evaluation.

For these purposes, we selected a common benchmark problem – Sisyphus-2 elevator design (Yost 1992). The design of an elevator system is a task involving number of interconnected variables, value ranges, constraints and fixes. For the reader's illustration, there are approximately 200 variables or parameters that need to be assigned values, and numerous auxiliary parameters, constants and factors that must be determined during the design. In addition, there is around 50 compatibility constraints plus many safety regulations and structural dependencies. In other words, Sisiphus-2 elevator design is definitely not a simple and easy task. Nevertheless, it is very well structured, and it does not require the designer to tackle any unexpected issues and use any tacit knowledge.

Before reviewing a typical sequence of decisions and actions in the design of an elevator, we look at the reasons why this task is considered well structured. In section 2.1, we highlight the essential points contributing to a good structure, and illustrate them on extracts from the Sisyphus-2 knowledge base. The records are first presented in a verbal form, followed by a formal expression in a predicate calculus. All the statements describing the task knowledge are taken from the initial Gregg Yost's (1992) report. However, the claims are supported by the extensive experience with this domain we acquired at KMi in the past projects (Motta, O'Hara *et al.* 1996; Motta and Zdrahal 1996; Motta 1997; Motta and Zdrahal 1998).

2.1 Sisyphus-2 elevator design is well structured

Below, we mention a few essential reasons that contribute to the well-structured nature of the Sisyphus-2 problem, as well as some observations why it cannot be considered 'computationally simple'. In this introduction, we use Simon's (1973) criteria that are typical for many well-structured problems, and develop our justifications around them. The main hallmarks of well-structured design problems are highlighted in the following (non-exclusive) list:

a) Existence of a criterion for testing the suitability of any candidate solution;
b) Existence of a problem space where all phases of the design can be represented unambiguously;
c) All moves from one state of design to the next one can be clearly and unambiguously represented;
d) Any knowledge about the problem can be represented within the given problem space(s).

Let us now translate Simon's generic characteristics into the specific domain of Sisiphus-2 elevator design (Yost 1992). In the subsequent text, all direct citations from Yost's report appear in italicised font and quoted. First, we look at the existence of a clear criterion for the determination if a candidate solution is admissible and acceptable. Yost formulates the criterion of solution admissibility very clearly at the beginning: '*to configure an elevator system, one must assemble a collection of components that satisfies both customer demands and safety regulations*'. In other words, design – in this case the configuration of the basic components, must satisfy all applicable constraints whether they are imposed by compatibility issues, safety regulations or a customer. Designer typically builds an elevator system stepwise, and in case the current state fails to satisfy any arbitrary constraint, designer must usually backtrack; i.e. return back to the last consistent state and re-consider the decision made.

Sisiphus-2 elevator design is concerned solely with the satisfaction of various kinds of constraints. There are many different sources for the constraints. Nevertheless, at the end of the process, all of them are equal; i.e. all of them must be satisfied. In one way or another, all constraints represent explicit conditions that can be easily evaluated by any truth-checking engine. There are no 'partial truths' in this problem – a constraint is either satisfied (evaluated as logically true) or violated (logically false). The issues with the personal preferences, aesthetics or efficiency are not considered.

In addition to the well-defined criteria for the evaluation of constraint satisfaction and eventually the solution admissibility, there is also a well-defined problem space. In this particular case, various elevator sub-systems, modules, components, and sub-components form the problem space, and are known in advance. Moreover, these blocks of a problem space are not only known in advance, but they are also complete; i.e. *all* the modules necessary for the design are available. Each of the components and modules is described by specified parameters that must be assigned values during the design process. The values may be selected from the available enumeration associated with each particular parameter, number interval range, or often computed by given algebraic formulae.

All constraints and subsequently fixes for the violated constraints are also expressed using the same terminology of descriptive parameters. Many constraints are expressed by a straightforward and clear algebraic formula containing common arithmetical and relational operators. Other constraints are often expressed as compatibility pairs or tables. In general, no parameters, value ranges, constraints, and fixes require any re-interpretation during the design process. The constraints and fixes available are *sufficient* to solve the problems of elevator design – at least in this particular, restricted domain.

We shall mention that Sisyphus-2 elevator design does not work with the general elevators. It is concerned with a sub-category of elevators that satisfy certain circumscribing conditions. Among other boundaries, the focus is on the cable-powered elevators with machinery rooms always being at the top. The task assumes that all floors in the building are of the same height, and the shaft is perfectly even throughout the building. There are certain maximal values for the elevator speed and capacity. All in all, these boundary conditions guarantee that the resulting logical theory is a sound problem solving theory for the design of elevator systems. As long as the design is within these boundaries, the design space is defined very precisely, and no surprising and unexpected circumstances may occur.

Not only parameters and constraints are defined precisely, but also the means for the navigation in the design space. The rule that applies to the assignment of values to most parameters is very simple and straightforward; in Yost's words: "*Initially, start with the smallest (model), though (this) may be upgraded to fix certain constraint violations*". Yost suggests also a typical strategy for the control of a design process, when he breaks the process into the following steps or moves:

- *First, get values for all of the input parameters from the customer.*
- *Then, derive values for all of the other parameters discussed.*
    - *Values for parameters can be derived in any order, once values have been derived for any other parameters, on which the currently attended ones depend.*
- *Process constraints and design modifications (fixes). Repeat derivation of values, if needed.*
- *Stop when there are no more parameters or constraints to process, and report the values of the output parameters.*

We follow this ‘algorithm’ for an elevator design in the subsequent sections to show different operations and a typical order, in which they are executed. In general, this algorithm is common for many problems that involve constraint satisfaction (Kumar 1992). However, many authors suggest various modifications to this sequence, in order to improve the operational efficiency (Brown 1996). Such a sequence can be easily implemented in a common predicate calculus, with the control mechanism disguised as a completeness-checking engine. In other words, having a schema of an elevator system, the completeness checker ‘knows’ what predicates need to be established in order to evaluate the admissibility of the schema. A predicate may be evaluated, if it exists in the logical theory as a statement, or if it can be inferred in the given logical theory. The role of a completeness checker is to assign the relevant parameters trying to complete a schema of a system or a module, so that a constraint checker may evaluate the feasibility of such assignment.

In respect to the knowledge needed for the task and its representation, Sisyphus-2 elevator design is also well defined. It works only with the explicit knowledge of design components, modules, systems, and corresponding parameters. Such knowledge may be very effectively represented as *schemas* or *frames* (Barr and Feigenbaum 1982; Cohen and Feigenbaum 1982) or *prototypes* (Gero 1990) with the descriptive parameters, allowed values for the parameters, and possibly procedures for the computation of missing pieces. Schema is an easy-to-understand knowledge representation; it is also rather straightforward to translate into other formal representations, such as logic. All knowledge that is needed for designing the elevators within Sisyphus-2 domain can be explicitly expressed in such formalism.

If there is something originating in the designer’s experience (e.g. design fixes resolving the constraint violations), then such knowledge has been pre-processed and formulated in clear, explicit terms. No additional representation is required for encoding this knowledge. We return to the statement already made earlier that all knowledge required for the elevator design is already present in the Sisyphus-2 knowledge base. There is no need to ‘invent’ new components, introduce new constraints, or modify the design fixes. In other words, the conceptual frame for the Sisyphus-2 elevator tasks is ‘static’ – i.e. well articulated, not requiring any changes during design.

2.2 Domain knowledge in Sisyphus-2

Let us briefly look at knowledge that is available to a designer working with a Sisyphus-2 elevator design problems. Here we move from the generic statements towards specific terms, and give examples of what constitutes particular types of domain knowledge. Sisyphus-2 is a typical example of design problems with a vast design space, in which many individual variables and parameters need to be set during the design. However, despite the huge number of parameters that need to be assigned a value, the overall structure of the task is very comprehensible and can be navigated with ease. Roughly speaking, there is the following information available in the Sisyphus-2 design task:

- Information about elevator sub-systems, modules, and components;
- Information about constraints placed on the modules and parameters;
- Information about useful fixes for the resolution of inconsistent partial assignments.

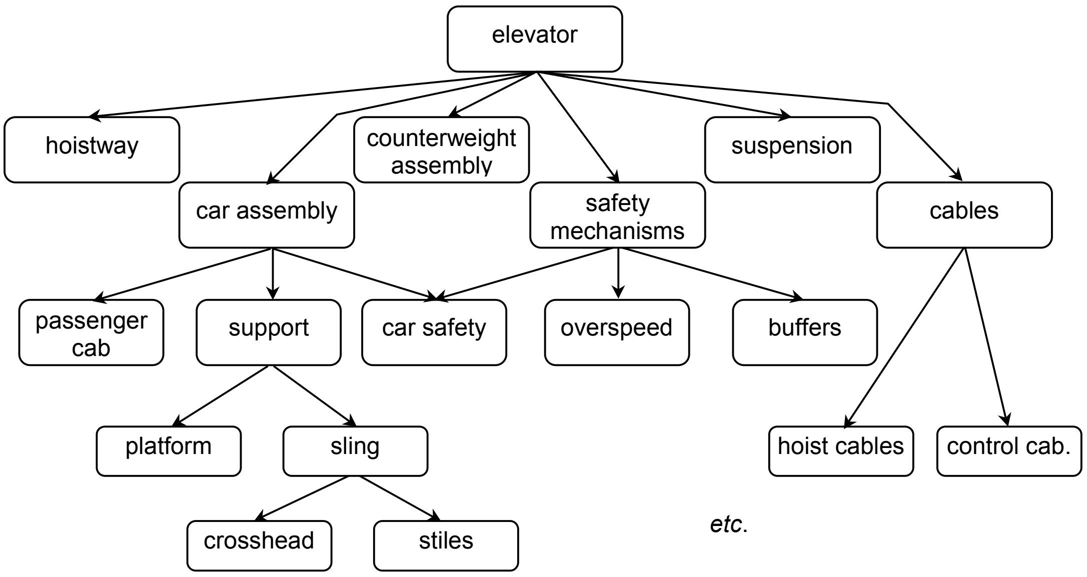


Figure 2–1. Relation of 'containment' – a hierarchical structure of Sisyphus-2 domain

First, knowledge of elevator modules and components is structured hierarchically; i.e. the structure of a design space is based on the relationship of 'containment'. Sub-systems 'contain' (or consist of) several, usually simpler modules; modules in turn may 'contain' simpler components, etc. The hierarchical breakdown of the problem ends with the components that are readily available in various catalogues. It is then the task of a designer to choose the right set-up from the available models. Thus, according to Yost, an elevator 'contains' a car assembly, counterweight assembly, hoistway, cables, suspension, and safety mechanisms. A car assembly typically 'contains' a passenger cab and a support structure; support is made of a platform and so-called sling (a frame around the cab), etc. An illustrated extract of such a breakdown is depicted in Figure 2–1, for more details see also (Yost 1992).

2.2.1 Knowledge of components and modules

Since the hierarchy depicted in Figure 2–1 is based upon relation of 'containment', it does not have any sense to discuss the inheritance of the attributes associated with the individual items. Nevertheless, each of the modules or components in the hierarchy has one or more descriptive parameters – some of them variables with given ranges of allowed values, others constant. Already-mentioned *schemas* offer a particularly suitable representation for this kind of information. A schema is, roughly speaking, a kind of template with some slots already pre-populated and others associated with certain procedures for establishing their value according to some other parameters. Below we show two 'simple' schemas of components needed for a passenger cab – cab platform and sling.

As we can see in Table 2–1 and Table 2–2, schemas consist of attributes, allowed values, and procedures. Some of the attributes have 'empty' values – this is a sign that the actual value (e.g.

number or model) is chosen by the designer, and depends only on the initial requirements. For instance, parameter ‘*platf-width*’ is set up using a value specified in an initial problem specification. Other attributes in a schema may depend on the values of certain specified parameters from the same schema or from a different schema. In other words, there may exist a specific procedure for the calculation of a value based on the values of other attributes. Thus for instance, parameter ‘*platf-height*’ depends only on the selected ‘*platf-model*’ (see Proc. 1 in Table 2–1). Whereas ‘*factor-Z*’ refers also to the value of parameter ‘*door-speed*’ borrowed from another schema, in this case ‘*Door*’ (see Proc. 2 in Table 2–1).

Table 2–1. Representation of component ‘platform’ as a schema.

Schema:	**Platform**	***myPlatform***		
Attributes:	*Name(s):*	*Type(s):*	*Value(s):*	*Unit(s):*
	- platf-model	enum	{2.5B, 4B, 6B}	
	- platf-width	number	*init-attribute-value (platf-width, W)*	inch
	- platf-depth	number	*init-attribute-value (platf-depth, D)*	inch
	- platf-height	number	*See proc. 1*	inch
	- factor-Z	number	*See proc. 2*	
	- factor-AP	number	*See proc. 3*	
	- factor …	…	…	
	- platf-weight	number	*See proc. N*	pound
Procedures:				

	Proc. 1	IF platf-model = 6B THEN platf-height = 6.6875
		ELSE platf-height = 6.625		

	Proc. 2	IF door-speed = single THEN factor-Z = 1.72
		ELSE factor-Z = 2.0		

	Proc. 3	IF platf-model = 4B THEN
		factor-AP = (platf-width – 0.125) / 9.5
		ELSE IF platf-model = 2.5B THEN …		

	…	…		

	Proc. N	IF platf-model = 4B THEN platf-weight =
		(35 + platf-width * factor-X + 2.774 * platf-depth +
		+ 0.03 * platf-width * platf-depth + 0.226 * opening-width +
		+ factor-AP * (0.226 * platf-depth – factor-Z))
		ELSE IF platf-model = 6B THEN …
		ELSE …		

Table 2–2. Representation of component ‘sling’ as a schema.

Schema:	**Sling**	***mySling***			
Attributes:	*Name(s):*	*Type(s):*	*Value(s):*		*Unit(s):*
	- sling-model	enum	{2.5B-18, 2.5B-21, 4B-HOSP, 4B-GP, 6C}		
	- stile-length	number	*See proc. 1*		inch
	- factor-A	number	*See proc. 2*		
	- factor-B	number			
	- factor-C	number			
	- sling-weight	number	*See proc. N*		pound
Procedures:					
	Proc. 1	stile-length = = sling-underbeam-dist + crosshead-height + platf-height – 0.81			
	Proc. 2	**SLING MODEL**	**A**	**B**	**C**
		2.5B-18	1.5	1.002	56
		2.5B-21	1.75	1.002	94
		4B-HOSP	1.8	1.2	223
		4B-GP	2.5	1.6	223
		6C	3.1	2.2	317
	…	…			
	Proc. N	sling-weight = = factor-A * platf-width + factor-B * stile-length + factor-C			

Procedures can have many different forms. In the examples, there are some procedures written in form of production rules; e.g. Proc. 2 in Table 2–1: '*IF door-speed = single THEN factor-Z = 1.72*'. Production rules are a favourite representation because it is to translate to the logical implications. Other procedures are formally represented as arithmetic equations; e.g. Proc. N in Table 2–2 asserts: '*sling-weight = factor-A * platf-width + ...*'. Another typical representation from the industrial catalogues, are various charts and tables; as for instance, small table for choosing the values for factors *A*, *B*, *C* according to the selected '*sling-model*' (see Proc. 2 in Table 2–2). We can see that several formal representations may be used in a definition of the same schema. These are however, not the only options for representing our knowledge of attributes and hierarchy.

Schemas as a formal representation may be very efficiently translated to the predicate logic. The following logical formulae express extracts from the same knowledge as the hierarchy depicted in Figure 2–1 or the two tables. In order to make the statements readable, the parameters to be assigned values are written in small letters (e.g. '*platf-model*'), and modules or components begin with a capital letter (e.g. '*Platform*'). Numeric values of the constants appear as numbers (e.g. *1.72*), and string constants are quoted (e.g. '*single*'):

- ∃ Elevator, Car, Counterweight, Hoistway, Cables ∈ *Schemas*:
  comp (Car) ∧ comp (Counterweight) ∧ comp (Hoistway) ∧ comp (Cables) ∧ … ∧
  ∧ system (Elevator) ⇔ contains (Elevator, { Car, Counterweight, Hoistway, Cables, …})
- ∃ Car, Cab, Support, Safety ∈ *Schemas*:
  system (Car) ∧ comp (Cab) ∧ comp (Support) ∧ comp (Safety) ⇔
  ⇔ contains (Car, { Cab, Support, Safety })
- ∃ Support, Platform, Sling ∈ *Schemas*:
  system (Support) ∧ comp (Platform) ∧ comp (Sling) ⇔
  ⇔ contains (Support, { Platform, Sling })

- ∀ Platform: has-attribute-value (Platform, platf-model, '2.5B') ⇒ comp (Platform)
  ∀ Platform: has-attribute-value (Platform, platf-model, '4B') ⇒ comp (Platform)
  ∀ Platform: has-attribute-value (Platform, platf-model, '6B') ⇒ comp (Platform)
  // this says that allowed platform models 2.5B, 4B and 6B – all instantiate platform 'schema'
- ∀ Platform: has-attribute-value (Platform, factor-Z, 1.72) ⇔
  ⇔ (∃ Door: has-attribute-value (Door, door-speed, 'single'))
  ∀ Platform: has-attribute-value (Platform, factor-Z, 2.0) ⇔
  ⇔ (∃ Door: has-attribute-value (Door, door-speed, 'double'))
  // factor-Z is equal to 1.72 iff the value of parameter door-speed is 'single', otherwise it is 2.0

In a similar manner, we can continue and write down logical statements for all hierarchical relationships, and all remaining attributes and procedures. As we can see, such a representation is rather extensive and time-consuming. Nevertheless, the result is a very powerful theory that represents our knowledge of designing elevators in the Sisyphus-2 domain.

2.2.2 Compatibility and structural constraints

Next, let us briefly illustrate typical examples of constraints that must be taken into account, when designing an elevator system in Sisyphus-2. Similarly as in the previous section, a verbal formulation of a constraint is followed by a translation to the logic. All constraints are taken from Yost's (1992) report, and where applicable we use the constraint number and section, in which it is defined in the report. This has no impact on the translation of the constraint; however, the reader may be interested in checking the context of a particular constraint or find further information.

Consider, for instance, the illustrative example of a typical constraint below. This is marked as C-3, and addresses the assignment of values to the parameter '*blocking height*' of the elevator component '*Car Buffer*'. Its verbal definition reads as follows:

- **C-3a**: The blocking height of a **CAR BUFFER** must be at least zero inches.
- **C-3b**: The blocking height of a **CAR BUFFER** must be at most 120 inches.

The constraints described above may be used if a designer wants to validate the selection of a particular buffer model against the domain constraints. These restrictive minimal and maximal limits are typical in the validating contexts. It is already another question when the validation takes place. It may precede the actual selection of a component to consider only those alternatives that comply with certain constraints, or it may follow the selection of a component to 'approve' it. The constraints that have form as showed above are in Sisyphus-2 domain playing a role of structural or safety constraints.

Another type of constraints in the Sisyphus-2 domain is a compatibility constraint. These constraint express knowledge that only certain components and modules can 'co-exist', if the design is to be admissible. The following examples may be found in Yost's report, section 7 (the numbers at the beginning refer to Yost's numbering of constraints and fixes):

- **C-45:** **PLATFORM MODEL** 2.5B is only compatible with **SLING MODEL**s 2.5B-18 or 2.5B-21
- **C-46a**: When the **SLING MODEL** is either 2.5B-18 or 2.5B-21, the underbeam distance must be between 108 and 123 inches, inclusive.

Let us now translate the illustrative constraints to the language of logical expressions and formulae. In the translation, we comply with the perspective seeing these statements as validation conditions rather than generative, production rules. For sake of simplicity and without placing too big burden on the reader of this text, we translate only one constraint from each list.

$\forall$ CarBuffer:
has-attribute-value (CarBuffer, blocking-height, BH) $\Rightarrow$
$\Rightarrow$ { greater-than (BH, 0) $\wedge$ smaller-than (BH, 120) }

$\forall$ Platform, Sling, PM, SM:
{ equal (PM, '2,5B') $\wedge$ equal (SM, '2.5B-18') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Platform, platf-model, PM) $\wedge$ has-attribute-value (Sling, sling-model, SM) }

$\forall$ Platform, Sling, PM, SM:
{ equal (PM, '2,5B') $\wedge$ equal (SM, '2.5B-21') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Platform, platf-model, PM) $\wedge$ has-attribute-value (Sling, sling-model, SM) }

2.2.3 Design fixes and recommendations

In addition to design components and constraints, Yost also formulates various recommendations how to repair a partial solution, if a particular constraint is violated. These 'fixes' may be seen as a form of experiential knowledge, or a kind of best practice that was acquired empirically. Nevertheless, in the Sisyphus-2 domain, the designer is not asked to learn or otherwise acquire such knowledge. For the purposes of designing elevators, the described fixes are explicit representations of moves or actions that must be taken in order to restore the validity/admissibility of the designed product. Thus, they may be encoded as a part of an inference engine or an overall design control strategy.

Each of the proposed recommendations has some impact on the current state of the design. Yost expresses the impact of a particular fix by so-called desirability level ('value function'). There are up to ten desirability levels ordered from the most to the least desirable (i.e. preference ordering). For instance, level D1 represents '*no negative impact at all*', and level D4 stands for '*changes in a minor equipment selection or size*'. The least desirable level D10 represents changes that would lead to '*a modification of major contract specifications as given by the customer*'. For example, the constraints from the previous section, have the following fixes or recommendations associated with them:

- Fix to **C-3a**: If C-3a is violated, increase the **HOISTWAY PIT** depth in one inch steps (**D9**).
- Fix to **C-3b**: If C-3b is violated, decrease the **HOISTWAY PIT** depth in one inch steps (**D9**).
- Fix to **C-45**: If this constraint is violated, upgrade the **PLATFORM MODEL** (**D8**).
- Fix to **C-46**: To fix any of the constraints listed, try upgrading the **SLING MODEL** (**D4**).

The control strategy for the elevator design tries to use the fixes with the smallest possible number. In other words, this strategy is known in the constraint satisfaction problems as '*if there are more alternatives find the one with the least negative impact on the current assignment*' (Kumar 1992). The impact is measured by a number of schemas and constraints that must be re-validated and/or re-calculated if a particular change happens. In our specific example, the selection of a '*platform model*' (C-45) affects more components, and may be more difficult to realise than selecting another '*sling-model*' (C-46).

2.3 Designing an elevator in Sisyphus-2

Every design task begins with the initial requirements describing the desired features the designed product should exhibit. Elevator design in Sisyphus-2 domain is no exception, and Yost (1992) addresses also the initial requirements in his report. In section 2 of his document, he lists the information the customer must supply before an elevator can be configured. These initial requirements include certain characteristics of the elevator system itself (such as for instance, car speed or capacity), as well as relevant building dimensions (such as opening count or hoistway pit depth). Consider, for instance, the following values for the assignment in the initial specification of the problem:

- init-attribute-value (car-capacity, 3000);
- init-attribute-value (platf-width, 100);
- init-attribute-value (platf-depth, 100);
- init-attribute-value (cab-height, 96);
- init-attribute-value (opening-width, 65);
- init-attribute-value (door-speed, 'double')

One requirement that is implicitly present in Yost's text is that the customer indeed wants an elevator and not some other artefact. We shall represent this requirement as an explicit desire, so that the actual design process can be started. Term '*Elevator*' in Italics stands for a schema from the applicable domain theory (see section 2.2), and '*myElevator*' is a specific instance of such a schema in a logical theory of Sisyphus-2 we aim to achieve.

- $\exists$ MyElevator $\in$ *Elevator* : Sisyphus-2-domain $\vdash$ system (myElevator)

The desired goal is not proven in the theory, therefore, a designer tries to find such a proof. In other words, the designer may apply the available domain knowledge to infer the components an elevator system is (typically) consisting of. Verbally said, an elevator may be designed, if all its components are available (or can be designed) in the same theory. Semi-formally, the statement '*system (Elevator)*' is implied by the logical theory, if such statements as '*comp (Car)*', '*comp (Counterweight)*', or '*comp (Hoistway)*' are already prove in the theory. Similarly, statement '*comp (Car)*' is implied by valid statements '*comp (Cab)*' and '*comp (Support)*', and so on until a 'primitive' component is reached. The following sequence of logical formulae from the domain theory expresses the inference of the components that build up a higher-level module/component/system.

∃ Elevator, Car, Counterweight, Hoistway, Cables ∈ *Sisyphus-2-schemas*:
comp (Car) ∧ comp (Counterweight) ∧ comp (Hoistway) ∧ comp (Cables) ∧ … ∧
∧ contains (Elevator, { Car, Counterweight, Hoistway, Cables, …}) ⇒ system (Elevator)
// Main problem decomposed: need to prove '∃ Car' (etc.) to prove '∃ Elevator'

∃ Car, Cab, Support, Safety ∈ *Schemas*:
comp (Support) ∧ comp (Safety) ∧ comp (Cab) ∧ contains (Car, { Cab, Support, Safety }) ⇒
⇒ system (Car)
// Sub-problem 1 decomposed further: need to prove '∃ Support' to confirm '∃ Car'

∃ Support, Platform, Sling ∈ *Schemas*:
comp (Platform) ∧ comp (Sling) ∧ contains (Support, { Platform, Sling }) ⇒
⇒ system (Support)
// Sub-problem 2 decomposed further: need to prove '∃ Platform' to confirm '∃ Support'

∀ Platform: has-attribute-value (Platform, platf-model, '2.5B') ⇒ comp (Platform)
// A proof to ∃ Platform may be achieved if we select a specific platform model, e.g. '2.5B'

- ∀ Platform = myPlatform: has-attribute-value (Platform, platf-model, '2.5B')
  // the first commitment sets the platform model to type '2.5B'

Now, the designer already has an instance of a '*Platform*' schema that needs to be configured in order to confirm this selection as admissible. There is a specific condition within the Sisyphus-2 domain that restricts the selection of platform model '*2.5B*' only for the car capacity smaller than 2500 pounds, the width smaller than 84 inches, and depth smaller than 60 inches. In our case however, the values of the 'constraining' parameters are beyond the allowed ranges. The desired width and depth of the car platform were initialised to 100 inches. In other words, selection of model '*2.5B*' is not admissible in the current problem solving theory. Consider this version of the mentioned constraint articulating a condition of the admissibility of a particular platform model (see also in section 2.2.2):

∀ Platform, Car, W, D, C:
{ has-attribute-value (Platform, platf-model, '2.5B') } ⇒
⇒ { has-attribute-value (Platform, platf-width, W) ∧ has-attribute-value (Platform, platf-depth, D) ∧
∧ has-attribute-value (Car, car-capacity, C) ⇒
⇒ smaller-than-equal (C, 2500) ∧ smaller-than-equal (W, 84) ∧ smaller-than-equal (D, 60) }

Thus, the partial selection of a particular model violated at least one constraint. The selection is inadmissible and must be rectified in order to continue. The Sisyphus-2 theory contains also a fix rectifying the violation, that recommends upgrading the platform model. It should be possible to find another statement in the domain theory that would confirm the existence of a valid component '*Platform*'. From the domain knowledge, we know that there is also model '*4B*':

∀ Platform: has-attribute-value (Platform, platf-model, '4B') ⇒ comp (Platform)
// Another proof to '∃ Platform' may be achieved if we select a specific platform model, e.g. '4B'

- ∀ Platform = myPlatform: has-attribute-value (Platform, platf-model, '4B')
  // second commitment after constraint rectification sets the platform model to type '4B'

Domain theory contains again a set of conditions to validate the admissibility of this selection. This time, model '*4B*' is allowed if the car capacity is more than 2500 pounds and the platform width and depth are smaller than 128 and 108 inches, respectively. Alternatively, width and depth may be smaller than 115 and 126 inches, respectively. This constraint is not violated by the partial selection, therefore design may continue.

```
∀ Platform, Car, W, D, C:
  { has-attribute-value ( Platform, platf-model, '4B' ) } ⇒
  ⇒ [ { has-attribute-value ( Platform, platf-width, W ) ∧ has-attribute-value ( Platform, platf-depth, D ) ∧
        ∧ has-attribute-value ( Car, car-capacity, C ) ⇒
      ⇒ greater-than ( C, 2500 ) ∧ smaller-than-equal ( W, 128 ) ∧ smaller-than-equal ( D, 108 ) } ∨
    ∨ { has-attribute-value ( Platform, platf-width, W ) ∧ has-attribute-value ( Platform, platf-depth, D ) ∧
        ∧ has-attribute-value ( Car, car-capacity, C ) ⇒
      ⇒ greater-than ( C, 2500 ) ∧ smaller-than-equal ( W, 115 ) ∧ smaller-than-equal ( D, 126 ) } ]
```

// Constraint applicable for platform model '4B'; see (Yost 1992), section 5.2

Having changed the partial assignment, so that platform model is '*4B*', the designer satisfied the constraint, and consequently, value '*4B*' is an admissible partial instantiation of the schema '*Platform*'. This selection may be used in order to deduce further consequences of so-far admissible partial design. Among the consequences that should be determined according to the '*Platform*' schema, are values for the factors *AP*, *X*, *Z*, and parameter '*platform height*'. These assignments are auxiliary, and eventually lead to the computation of a value for the attribute '*platform weight*'. The following logical 'rules' are applicable for the purpose of deducing appropriate values for the individual factors:

```
∀ Platform: has-attribute-value ( Platform, factor-Z, 1.72 ) ⇔
   ⇔ ( ∃ Door: has-attribute-value ( Door, door-speed, 'single' ) )
∀ Platform: has-attribute-value ( Platform, factor-Z, 2.0 ) ⇔
   ⇔ ( ∃ Door: has-attribute-value ( Door, door-speed, 'double' ) )
```

// e.g. factor-Z is equal to 1.72 iff the value of parameter 'door-speed' is 'single', otherwise it is 2.0

```
∀ Platform: has-attribute-value ( Platform, factor-X, 2.26 ) ⇔
   ⇔ ( ∃ Door: has-attribute-value ( Door, door-speed, 'single' ) ) ∧
     ∧ has-attribute-value ( Platform, platf-model, '4B')
∀ Platform: has-attribute-value ( Platform, factor-X, 2.354 ) ⇔
   ⇔ ( ∃ Door: has-attribute-value ( Door, door-speed, 'double' ) ) ∧
     ∧ has-attribute-value ( Platform, platf-model, '4B')
```

// factor-X is equal to 2.26 for platform model '4B' iff the value of parameter door-speed is 'single'; if the door speed is 'double' then the value is 2.354

```
∀ Platform: has-attribute-value ( Platform, factor-X, 2.441 ) ⇔
   ⇔ ( ∃ Door: has-attribute-value ( Door, door-speed, 'single' ) ) ∧
     ∧ has-attribute-value ( Platform, platf-model, '6B')
∀ Platform: has-attribute-value ( Platform, factor-X, 2.394 ) ⇔
   ⇔ ( ∃ Door: has-attribute-value ( Door, door-speed, 'double' ) ) ∧
     ∧ has-attribute-value ( Platform, platf-model, '6B')
```

// factor-X is equal to 2.441 for platform model '6B' iff the value of parameter door-speed is 'single'; if the door speed is 'double' then the value is 2.394

Nevertheless, the above-mentioned formulae cannot be readily evaluated, because so far, there is no information about the speed of cab doors proven in our theory. In other words, the designer has to postpone this calculation until he obtains the necessary information about the component '*Door*'. This component was not defined in the previous sections, but it has a similar structure as the schemas presented in section 2.2.1. Component '*Door*' is described using attributes '*door speed*', '*door opening type*', and '*opening strike*'. All these parameters are instantiated from the customer's initial requirements, and may have the following values. '*Door speed*' is either '*single*' or '*double*'; '*door opening type*' may be '*one-side*' or '*central*'; and the allowed values for '*opening strike*' are '*left-hand*' or '*right-hand*'.

That amount of information is slightly redundant because the only parameter that may in any way influence the 'postponed' deduction is '*door speed*'. Thus, checking the customer's requirements at the beginning of this section, the designer may set up the component of '*Door*', and assert its attribute as '*has-attribute-value (myDoor, door-speed, 'double')*'. Such an assertion enables the deduction of values for factors *X = 2.354* and *Z= 2.0* that are needed for the calculation of '*platform weight*'. This deduction may draw on a simple arithmetical engine that calculates the above formula. The result of the calculation eventually shows that '*platform weight*' could be instantiated with a value '*1071.88*', and from the schema it is possible to associate unit '*pounds*' with this number. In order to summarise, the following statements were inferred so far in our design problem:

- ∀ Platform = myPlatform: has-attribute-value (Platform, platf-model, '4B')
  // first commitment selected the platform model '4B'
- ∀ Platform = myPlatform: has-attribute-value (Platform, platf-height, 6.625)
  // then, platform height was deduced to be 6.625 inches
- ∀ Platform = myPlatform: has-attribute-value (Platform, factor-AP, 10)
  // next, factor-AP was calculated and is equal to 10
- ∀ Platform = myPlatform: has-attribute-value (Platform, factor-Z, 2.0)
  // factor-Z was calculated, and equals to 2.0
- ∀ Platform = myPlatform: has-attribute-value (Platform, factor-X, 2.354)
  // factor-X is also calculated, and equals to 2.354
- ∀ Platform = myPlatform: has-attribute-value (Platform, platf-weight, 1068.49)
  // and finally, platform weight was determined, and is equal to 1068.49 pounds

At this stage, the designer finished one of the necessary components of an elevator– a car platform, and may continue to extend the partial design. The next component that can be attended, is for instance, the triplet of '*sling*', '*safety beam*', and '*crosshead*'. These components form a rectangular frame that surrounds the passenger cab from the sides, the top and bottom, and are essential for lifting the cab. Since they are primitive structures of the Sisyphus-2 domain, they can be selected similarly, as it was shown earlier with the component '*Platform*'. All three components are necessary for the design of a car 'support' structure, and as such must be determined before any further development of passenger cab module can occur.

Let us begin with the '*Sling*' that was already mentioned in section 2.2.1 in form of a schema. Similarly as it was the case with the '*Platform*' component, the underlying statements show the sequence of formulae that can be used to select a '*Sling*' component. Based on the Sisyphus-2 recommendation from section 2.2.3, we start with the selection of the smallest model:

∀ Sling: has-attribute-value (Sling, sling-model, '2.5B-18') ⇒ <u>comp (Sling)</u>
// A proof to '∃Sling' may be achieved if we select a specific sling model, e.g. '2.5B-18'

- ∀ Sling = mySling: has-attribute-value (Sling, sling-model, '2.5B-18')
  // the first commitment sets the sling model to type '2.5B-18'

At the moment, there are no structural or compatibility constraints applicable for the selected '*sling model*' (2.5B-18). This selection is also admissible in respect to the parameters of already designed components (e.g. '*platform model*' equal to '*4B*'). It means that the designer can continue with the deduction of consequences for the select '*sling*'; namely the calculation of the weight and weight factors. Consider the following formulae that may help to determine the desired values:

∀ Sling: has-attribute-value (Sling, sling-model, SM) ∧ equal (SM, '2.5B-18') ∧
  ∧ has-attribute-value (Sling, factor-A, 1.5) ∧ has-attribute-value (Sling, factor-B, 1.002) ∧
  ∧ has-attribute-value (Sling, factor-C, 56)
∀ Sling: has-attribute-value (Sling, sling-model, SM) ∧ equal (SM, '2.5B-21') ∧
  ∧ has-attribute-value (Sling, factor-A, 1.75) ∧ has-attribute-value (Sling, factor-B, 1.002) ∧
  ∧ has-attribute-value (Sling, factor-C, 94)
// factors A, B, and C are translated from Table 2–2 to the form of logical statements (extract)

∀ Sling ∃ Platform: has-attribute-value (Sling, sling-weight, 〈A*W + B*L + C〉) ⇔
  ⇔ { has-attribute-value (Sling, factor-A, A) ∧ has-attribute-value (Sling, factor-B, B) ∧
  ∧ has-attribute-value (Sling, factor-C, C) ∧ has-attribute-value (Platform, platf-width, W) ∧

$\wedge$ has-attribute-value (Sling, stile-length, L) }
// 'sling weight' can be computed according to given formula provided all the parameters are known

As it is visible from the last formula, one more parameter is missing in our partial design – a '*stile length*'. Nevertheless, this parameter can be easily computed by the following procedure that is defined for the respective parameter in the '*Sling*' schema:

$\forall$ Sling $\exists$ Platform, Crosshead: has-attribute-value (Sling, stile-length, $\langle$UB + C-H + P-H – 0.81$\rangle$) $\Leftrightarrow$
$\Leftrightarrow$ { has-attribute-value (Sling, underbeam, UB) $\wedge$
$\wedge$ has-attribute-value (Crosshead, cross-height, C-H) $\wedge$
$\wedge$ has-attribute-value (Platform, platf-height, P-H) }
// 'stile length' is given by the underbeam distance, crosshead height and platform height

However, deduction cannot continue yet, because some information is still missing. Namely, we do not have any knowledge in our partial design about '*crosshead height*' or '*underbeams*'. Nevertheless, the Sisyphus-2 domain theory knows of such components, and they can be tackled as two separate sub-problems. If we want to use that particular arithmetical formula, we suppose there is an object in our theory referred to as '*Crosshead*'. Therefore, the designer must find and configure such an object explicitly, including its parameters. This means to postpone the evaluation of the formula, and look at '*Crosshead*', which is a part of a '*Safety*' structure of the elevator's '*Car*'. Consulting the domain knowledge, the designer learns that there are several types of crossheads:

$\forall$ Crosshead: has-attribute-value (Crosshead, cross-model, 'W8x18') $\Rightarrow$ <u>comp (Crosshead)</u>
$\forall$ Crosshead: has-attribute-value (Crosshead, cross-model, 'W8x21') $\Rightarrow$ <u>comp (Crosshead)</u>
$\forall$ Crosshead: has-attribute-value (Crosshead, cross-model, 'C8x11.5') $\Rightarrow$ <u>comp (Crosshead)</u>
// alternative choices for a selection of a 'crosshead-model' (extract)

$\forall$ Crosshead: has-attribute-value (Crosshead, cross-model, CM) $\wedge$ equal (SM, 'W8x18') $\Rightarrow$
$\Rightarrow$ has-attribute-value (Crosshead, cross-height, 8.125)
$\forall$ Crosshead: has-attribute-value (Crosshead, cross-model, CM) $\wedge$ equal (SM, 'W8x21') $\Rightarrow$
$\Rightarrow$ has-attribute-value (Crosshead, cross-height, 8.25)
$\forall$ Crosshead: has-attribute-value (Crosshead, cross-model, CM) $\wedge$ equal (SM, 'C8x11.5') $\Rightarrow$
$\Rightarrow$ has-attribute-value (Crosshead, cross-height, 8.0)
// parameter 'crosshead height' depends on the selected crosshead models (extract)

To assist with the selection of an admissible '*crosshead*', there is a compatibility constraint relating the '*crossheads*' and '*slings*'. The constraint may be applied to inform the selection. Such an approach typically helps the designer choose an admissible model of a '*Crosshead*' straight away, without extensive backtracking. Consider the following samples of compatibility constraints:

$\forall$ Crosshead, Sling, CM, SM: { equal (CM, 'W8x18') $\wedge$ equal (SM, '2.5B-18') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Crosshead, cross-model, CM) $\wedge$
$\wedge$ has-attribute-value (Sling, sling-model, SM) }
// crosshead model 'W8x18' is only compatible with sling model '2.5B-18'

$\forall$ Crosshead, Sling, CM, SM: { equal (CM, 'W8x21') $\wedge$ equal (SM, '2.5B-21') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Crosshead, cross-model, CM) $\wedge$
$\wedge$ has-attribute-value (Sling, sling-model, SM) }
// crosshead model 'W8x21' is only compatible with sling model '2.5B-21'

$\forall$ Crosshead, Sling, CM, SM: { equal (CM, 'C8x11.5') $\wedge$ equal (SM, '4B-HOSP') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Crosshead, cross-model, CM) $\wedge$
$\wedge$ has-attribute-value (Sling, sling-model, SM) }
// crosshead model 'C8x11.5' is only compatible with sling model '4B-HOSP', etc.

Taking into account the partial design with a commitment to a sling model '*2.5B-18*', the designer quickly infers that a compatible crosshead model is '*W8x18*'. As soon as the commitment to a specific model of crosshead is recorded, it enables the deduction (calculation) of parameter '*crosshead height*' using the domain theory formulae mentioned above. Thus, the partial design is refined as follows:

- $\forall$ Crosshead = myCrosshead: has-attribute-value (Crosshead, cross-model, 'W8x18')
  // commitment to the crosshead model following the applicable constraint
- $\forall$ Crosshead = myCrosshead: has-attribute-value (Crosshead, cross-height, 8.125)
  // commitment to the crosshead model implies also the height of a crosshead

In a similar manner, the designer can discover that parameter '*underbeam*' can be calculated using the available value of '*cab-height*', and a recommendation for setting up parameter '*underbeam-space*' from the domain theory. Under such circumstances, the value of '*underbeam*' would be equal to the sum of parameters '*cab-height*' and '*underbeam-space*'. We skip the formal calculation of this parameter because it is essentially the same as the previous ones. The value is calculated to be 117 inches (see the first assertion below). However, there is an important constraint placed on the sling models and the underbeam distances:

- $\forall$ Sling = mySling: has-attribute-value (Sling, underbeam, 117)
  // value of underbeam distance for a given cab height is 117 inches

$\forall$ Sling, SM, UB: { has-attribute-value (Sling, sling-model, '2.5B-18') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Sling, underbeam, UB) $\Rightarrow$
$\Rightarrow$ greater-than-equal (UB, 108) $\wedge$ smaller-than-equal (UB, 123) }
$\forall$ Sling, SM, UB: { has-attribute-value (Sling, sling-model, '2.5B-21') } $\Rightarrow$
$\Rightarrow$ { has-attribute-value (Sling, underbeam, UB) $\Rightarrow$
$\Rightarrow$ greater-than-equal (UB, 108) $\wedge$ smaller-than-equal (UB, 123) }
// Constraints applicable for selected sling models; for details see (Yost 1992), section 7 (C-46)

Referring to a simple arithmetical engine we can now establish that *117 inches* is indeed within a given range $\langle 108, 123 \rangle$, and the constraint is not violated. The selected '*underbeam*' is compatible with the partial design. Consequently, the value of parameter '*underbeam*' is valid and may be used for further deductions. The designer may now return to the postponed calculation of the '*stile length*', and finally to the determination of the '*sling weight*'. The results of these two calculations are noted below in a form of asserted formulae:

- $\forall$ Sling = mySling: has-attribute-value (Sling, stile-length, 130.94)
  // value of stile length for a given platform model, crosshead model and underbeam is 130.94 inches
- $\forall$ Sling = mySling: has-attribute-value (Sling, sling-weight, 337.2)
  // value of sling weight for a given sling model and platform width is 337.2 pounds

Having determined the weights of '*platform*' and '*sling*' components, plus deciding about the '*crosshead*' and '*door*' component, the designer follows the algorithm. According to the algorithm, he could now return to the postponed definition of the elevator's '*car*' module. As it is visible from the rule repeated below, the next component to be configured is a '*safety beam*' structure. And similarly as with all previous components, there are certain '*safety models*' available to choose from, and according to the choice, other parameters would be computed.

$\exists$ Car, Cab, Support, Safety $\in$ *Sisyphus-2-schemas*:
comp (Support) $\wedge$ $\underline{\text{comp (Safety-B)}}$ $\wedge$ comp (Cab) $\wedge$ contains (Car, { Cab, Support, Safety }) $\Rightarrow$
$\Rightarrow$ $\underline{\text{system (Car)}}$
// problem decomposed, need to prove '$\exists$Safety-beam' to confirm '$\exists$Car'

$\forall$ Safety-B: has-attribute-value (Safety-B, saf-model, 'B1') $\Rightarrow$ $\underline{\text{comp (Safety-B)}}$
// 'saf-model' attribute must be selected in order to have a valid safety beam model

- $\forall$ Safety-B = mySafety-B: has-attribute-value (Safety-B, saf-model, 'B1')
  // first commitment made sets the safety beam model to type 'B1'

The tentative commitment to the particular safety beam model '*B1*' must be however, subjected to a constraint check before it can be used for further deduction. A few structural constraints determine admissibility of the different beam models, and its compatibility with other elevator components such as ‘*platform*’:

```
∀ Safety-B, Platform, SBM, W:
  { has-attribute-value ( Safety-B, saf-model, ‘B1’ ) } ⇒
    ⇒ { has-attribute-value ( Platform, platf-width, W ) ⇒ smaller-than-equal ( W, 93 ) }
```
// safety beam model ‘B1’ is only compatible with platforms that are less than 93 inches wide

```
∀ Safety-B, Platform, SBM, W:
  { has-attribute-value ( Safety-B, saf-model, ‘B4’ ) } ⇒
    ⇒ { has-attribute-value ( Platform, platf-width, W ) ⇒ smaller-than-equal ( W, 114 ) }
```
// safety beam model ‘B4’ is only compatible with platforms that are less than 114 inches wide
// Constraint applicable for different safety beam models; see (Yost 1992), section 5.4

Without going any deeper, it is clear that the tentative selection of ‘*safety model*’ ‘*B1*’ violates the first constraint. The designer may stop this particular path because under current circumstances, it will never lead to a successful (i.e. admissible) design. The fix for the rectification of the violated constraint is given by the Sisyphus-2 theory and is very simple –upgrade the model if needed (until it ceases violating any constraints). Thus, the designer re-visits the commitments, and chooses a different model of a ‘*safety beam*’ structure – this time ‘*B4*’:

∀ Safety-B: has-attribute-value (Safety-B, saf-model, ‘B4’) ⇒ <u>comp (Safety-B)</u>
// a new ‘saf-model’ is selected in order to have a valid safety beam model

- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, saf-model, ‘B4’)
  // the rectified commitment sets the safety beam model to type ‘B4’

Since ‘*platform width*’ was required by the customer to be *100 inches*, the newly selected safety model ‘*B4*’ is comfortably within the given range of the constraint *⟨93, 114⟩*. It is therefore an admissible assignment. Parameters that need to be determined include e.g. ‘*safety height*’ (the vertical dimension of the beam), ‘*safety-BG*’ (horizontal dimension between the guiderails), and ‘*safety weight*’ (given formula). Similarly as with the previous deductions, the values of the beam parameters largely depend on the selected safety beam models. The logical formulae below show the rules for deduction of parameters ‘*safety height*’, ‘*safety constant*’, and weight factors *A* and *B*.

```
∀ Safety-B: has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B1’ ) ⇒
        ⇒ has-attribute-value ( Safety-B, saf-height, 9.0 )
∀ Safety-B: has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B4’ ) ⇒
        ⇒ has-attribute-value ( Safety-B, saf-height, 10.0 )
∀ Safety-B: has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B6’ ) ⇒
        ⇒ has-attribute-value ( Safety-B, saf-height, 13.0 )
```
// safety beam heights depend on the selected models: B1 = 9.0; B4 = 10.0; and B6 = 13.0

```
∀ Safety-B:
      has-attribute-value ( Safety-B, saf-model, SBM ) ∧ [ equal ( SBM, ‘B1’ ) ∨ equal ( SBM, ‘B4’ ) ] ⇒
      ⇒ has-attribute-value ( Safety-B, saf-const, 2.25 )
∀ Safety-B:
      has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B6’ ) ⇒
      ⇒ has-attribute-value ( Safety-B, saf-const, 2.625 )
```
// safety beam length constants also depend on the selected models: B1 or B4 = 2.25; B6 = 2.625

```
∀ Safety-B: has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B1’ ) ∧
      ∧ has-attribute-value ( Safety-B, factor-A, 1.69 ) ∧ has-attribute-value ( Safety-B, factor-B, 1 )
∀ Safety-B: has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B4’ ) ∧
      ∧ has-attribute-value ( Safety-B, factor-A, 2.3 ) ∧ has-attribute-value ( Safety-B, factor-B, 540 )
∀ Safety-B: has-attribute-value ( Safety-B, saf-model, SBM ) ∧ equal ( SBM, ‘B6’ ) ∧
      ∧ has-attribute-value ( Safety-B, factor-A, 2.6 ) ∧ has-attribute-value ( Safety-B, factor-B, 1035 )
```
// factors A and B are taken directly from Yost’s report (section 5.4)

The deduction of values that would be admissible in the context of our particular partial design is straightforward. A single glimpse on the rules above reveals that the following assertions are the admissible ones (they comply with the selected mode '*B4*'):

- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, saf-height, 10.0)
  // the height of the safety beam model 'B4' equals 10 inches
- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, saf-const, 2.25)
  // the length constant of the safety beam model 'B4' equals 2.25 inches
- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, factor-A, 2.3)
  // the weight factor A of the safety beam model 'B4' equals 2.3
- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, saf-height, 10.0)
  // and finally, the weight factor B of the safety beam model 'B4' equals 540 pounds

Having successfully deduced the necessary parameters, the designer may now move to the next important deduction, which is the calculation of the weight of selected safety beam. In addition to knowing the weight of a beam, it is important to know its length; i.e. parameter '*safety-BG*' (distance between guiderails). The following statements represent the arithmetical equations for the calculation. First, the formula for the deduction of '*safety weight*', followed by formula for '*safety-BG*':

∀ Safety-B ∃ Platform: has-attribute-value (Safety-B, saf-weight, ⟨A*W + B⟩) ⇔
⇔ { has-attribute-value (Safety-B, factor-A, A) ∧ has-attribute-value (Safety-B, factor-B, B) ∧
∧ has-attribute-value (Platform, platf-width, W) }
// 'safety beam weight' can be computed according to given formula provided all parameters are known

∀ Safety-B ∃ Platform: has-attribute-value (Safety-B, saf-weight, ⟨ W + LC⟩) ⇔
⇔ { has-attribute-value (Safety-B, saf-const, LC) ∧
∧ has-attribute-value (Platform, platf-width, W) }
// 'safety beam weight' can be computed according to given formula provided all parameters are known

The actual values of the above mentioned parameters after the calculations are given below:

- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, saf-weight, 770.0)
  // the weight of the safety beam model 'B4' equals 770 pounds
- ∀ Safety-B = mySafety-B: has-attribute-value (Safety-B, safety-BG, 102.25)
  // and finally, the length of the safety beam model 'B4' between guiderails equals 102.25 inches

By now, the pattern of the design process is well visible. We believe that we can stop the example at this stage before the reader develops a strong dislike for the elevators. In the next section, we summarise the lesson learned from the Sisyphus-2 elevator design. We believe that we proved this domain to be well structured. Any difficulties with the task may arise only due to a large number of components, parameters, and constraints. Different control strategies may also contribute to the efficiency of a design process, frequency of backtracking, and severity of constraint violations. Nevertheless, it is not the purpose of this document to propose yet another algorithm for the Sisyphus-2 domain. To learn more about various approaches to the elevator design problem, we recommend the following references: (Yost 1992; Wielinga, Akkermans *et al.* 1995; Brazier, van Langen *et al.* 1996; Motta 1997; Motta and Zdrahal 1998).

2.4 Lessons learned from Sisyphus-2 design

Let us very briefly recapitulate the essential points of Sisyphus-2 elevator design that was sketched in the preceding sections. This section looks at specific lessons we learned from this excellent example of a logical and well-structured design. We also summarise the patterns that were observed in the example in a form of a schematic algorithm for well-defined problems.

The first observation was that Sisyphus-2 domain was very well defined. All components that were needed for the elevator design were given, including all relevant relations and dependencies. Also, domain constraints were expressed in great details using the vocabulary of domain components and parameters, which made it easier to navigate in the vast design space. There may exist many different configurations of elevators in the Sisyphus-2 domain; however, there is little need to 'explore' this space and apply any 'tacit' feelings. The task of elevator design in the Sisyphus-2 domain can be considered to be a *search for an admissible solution* or even better, an admissible configuration that forms a solution. The search can be implemented relatively efficiently using various problem solving methods, such as Propose&Revise (Motta and Zdrahal 1996) or Propose-Critique-Modify (Chandrasekaran 1990).

Another observation from the example is showing very little need for any creative contribution. The creativity or innovation is not demanded in the example; it can be even said that there is no space for any creativity at all in the Sisyphus-2 elevator design. The problem is defined in a typical circumscriptive fashion; i.e. what is given is all that is needed in order to solve the problem. In the case when a designer is stuck in a dead-end branch of the problem solving space, there are readily available recommendations and fixes. The fixes are formulated so, as to 'save' the designer the effort of exploring many possible rectification strategies, or invent novel strategies. Most of the 'innovative' attempts can be stopped right at the beginning by referring to the domain restrictions.

However, it may happen that a particular constraint does not have any 'fix' associated with it. In such case, the problem with a particular specification does not have any solution in the domain of Sisyphus-2. Such a scenario happens for instance, with the parameter *car-cab-height* that must be in the interval ⟨84, 240⟩ inches, and there are no fixes to rectify the violation of this constraint.

The well-structured nature is also the main reason why the Sisyphus-2 elevator design became a benchmarking example for various constraint satisfaction tools and methodologies. The structure of the problem almost calls for the optimisation of a configuration process and the development of efficient algorithms. The problem as it is defined, is perhaps more suitable for the constraint satisfaction research rather than motivating the research into nature of design and designing. Nevertheless, a core pattern of certain elementary actions is visibly repeated throughout the design task. Two patterns mentioned in the design of an elevator in Sisyphus-2 domain are depicted in the schemes below.

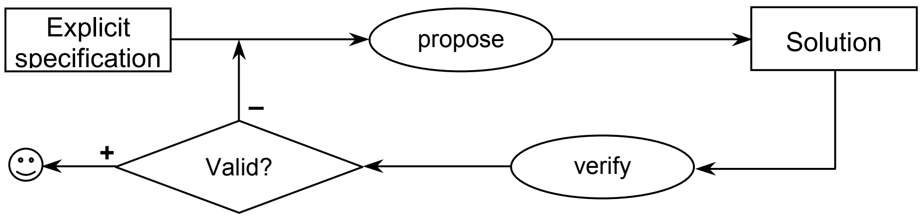


Figure 2–2. Elevator design as constraint satisfaction (simple version)

Both figures above and below comply with the generic (Chandrasekaran 1990)'s method named 'Propose-Critique-Modify'. Figure 2–2 considers a simpler situation, when a '*propose*' operation is responsible for the retrieval of necessary component(s) and the calculation of all relevant dependencies. The resulting candidate solution is verified against the set of constraints (e.g. mutual compatibility of components is checked), and in the case the verification proves inconsistency, a new candidate solution is proposed. This process may yield too much backtracking, thus decreasing the efficiency of the algorithm.

Figure 2–3 shows a more complex variant of the constraint satisfaction algorithm, where the initial proposal is verified against the relevant constraints before calculating any further dependencies. If the initial constraint check fails, the algorithm backtracks immediately and proposes a new partial solution. Once the initial verification is accomplished, the dependencies may be computed, and the refined solution is again verified against the relevant constraints once. This algorithm is more efficient because it stops, as soon as it discovers that a partial solution is inadmissible. It does not need to reach the end of a branch to find out that it is a dead end. The reader may remember that we analysed the task of an elevator design using this strategy and assessed the admissibility of the partial solutions.

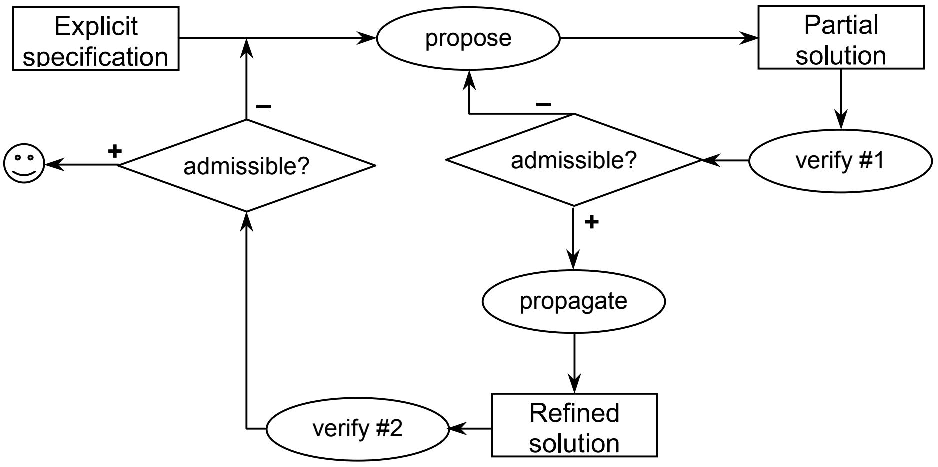


Figure 2–3. Elevator design as constraint satisfaction (with pre-validation)

Nevertheless, in both cases, the algorithm is looping as many times as necessary, until an ‘acceptable’ solution is found (if it exists). Perhaps, it would be better to say ‘until an *admissible* and *complete* solution is found’, if we want to account for the design tasks that can be decomposed in a similar manner as the elevator in the Sisyphus-2 domain. In the same time this vocabulary clearly differentiates between a completeness and tacit acceptability. Deeper analysis of the Sisyphus-2 domain and the methods mentioned in the schemes is beyond the scope of this document. For more details on the implementation of the operations ‘*propose*’, ‘*verify*’, and/or ‘*propagate*’, see for instance, (Yost 1992; Wielinga, Akkermans *et al.* 1995; Brazier, van Langen *et al.* 1996; Motta 1997; Motta and Zdrahal 1998).

3 TRANSCRIPTS OF DESIGN CONTEXTS

3.1 TASK-11: Active shock absorption & chassis clearance

Design brief:
Suspension and shock absorption are crucial elements in the car design, particularly with regard to the passengers' comfort. Design an active shock absorber and suggest a control strategy, so that the whole suspension mechanism would be able to adjust its properties according to the current state of the road. In addition to the absorption of shocks caused by an uneven road, we would like to adjust also the chassis clearance, so that the car has the optimal aerodynamic coefficients for a particular type of surface.

- **Initial problem context (DC-I.)**
  Articulated requirements (objectives):
  – none

  Articulated solutions (solution models):
  – none

[*end of context*]

- **Context number 2 (DC-II.)**
  Articulated requirements (objectives):
  - **R-1**. Control of shock absorption
    (The system should be able to control the car suspension/shock absorption.)
    - **R-1.1** Automatically adjustable shock absorption
      (Moreover, the absorption must be automatically adjustable in response to the actual terrain, road surface.)
  - **R-2**. Chassis clearance control
    (The system should be also able to work with chassis clearance – height above road.)
    - **R-2.2** Manually adjustable clearance
      (Let's take first the simpler case, when chassis clearance is constant, and can only change in response to driver's explicit request – e.g. off-road, motorway.)
  - **R-3**. Need to measure the deflection of the wheel
    (In order to control the shock absorber's behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation)
  - **R-4**. Need to measure also the vibrations
    (OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration)
  - **R-5**. Shock absorber rigidity must be adjustable!
    (This is essential to have an 'active absorber', the shock absorbing properties must be adjustable in response to road conditions. Moreover – this should be easy to do!!)
  - **R-6**. Controller for the shock absorption
    (See scheme D as an application of a simple control loop that can be used in this case.)

Articulated solutions (solution models):

- **S-1**. Simple control of the shock absorbers
  (See scheme D., so far we work with a constant clearance that can't be changed.)

[*end of context*]

- **Context number 3 (DC-III.)**
  Articulated requirements (objectives):
  - R-1. Control of shock absorption
    (The system should be able to control the car suspension/shock absorption.)
    - **R-1.3** Automatically AND manually adjustable absorption (***new in this context***)
      (This is a better alternative, the driver should be able to choose himself if he wants auto-mode or manually sets up the absorber rigidity/toughness.)
  - R-2. Chassis clearance control
    (The system should be also able to work with chassis clearance – height above road.)
    - **R-2.3** Automatically & manually adjustable clearance (***new in this context***)
      (Similarly, the driver should be able to say if he wants an automated optimisation of chassis clearance or sets the value manually to address a specific request.)
  - R-3. Need to measure the deflection of the wheel
    (In order to control the shock absorber's behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation)
  - R-4. Need to measure also the vibrations
    (OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration)
  - R-5. Shock absorber rigidity must be adjustable!
    (This is essential to have an 'active absorber', the shock absorbing properties must be adjustable in response to road conditions. Moreover – this should be easy to do!!)
  - **R-7**. Controlling the clearance (***new in this context***)
    (OK, when we want an auto- or manual mode for chassis adjustment, we need also some method for its control and an appropriate actuator.)
  - **R-8**. Joint controller of absorption AND clearance (***new in this context***)
    (This is a kind of 'master' controller that will coordinate the actions of the two simpler or dedicated sub-controllers. This controller will also be responsible for translating the driver's choices and commands to the values suitable for the sub-controllers, it should also activate and deactivate the controllers as needed – to respond to the driver's command)

Articulated solutions (solution models):

- **S-2**. Active control of the shock absorption and chassis clearance (***new in this context***)
  (See sch. F and all the details in the discussion.) ** ***See justification in App.D, part 4.1*** **

[*end of context*]

- **Context number 4 (DC-IV.)**
  Articulated requirements (objectives):
  - R-1. Control of shock absorption
    (The system should be able to control the car suspension/shock absorption.)
    - R-1.3 Automatically AND manually adjustable absorption

(This is a better alternative, the driver should be able to choose himself if he wants auto-mode or manually sets up the absorber rigidity/toughness.)

- R-2. <u>Chassis clearance control</u>
  (The system should be also able to work with chassis clearance – height above road.)
  - R-2.3 <u>Automatically & manually adjustable clearance</u>
    (Similarly, the driver should be able to say if he wants an automated optimisation of chassis clearance or sets the value manually to address a specific request.)
- R-3. <u>Need to measure the deflection of the wheel</u>
  (In order to control the shock absorber's behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation)
- R-4. <u>Need to measure also the vibrations</u>
  (OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration)
- R-5. <u>Shock absorber rigidity must be adjustable!</u>
  (This is essential to have an 'active absorber', the shock absorbing properties must be adjustable in response to road conditions. Moreover – this should be easy to do!!)
- R-7. <u>Controlling the clearance</u>
  (OK, when we want an auto- or manual mode for chassis adjustment, we need also some method for its control and an appropriate actuator.)
- R-8. <u>Joint controller of absorption AND clearance</u>
  (This is a kind of 'master' controller that will coordinate the actions of the two simpler or dedicated sub-controllers. This controller will also be responsible for translating the driver's choices and commands to the values suitable for the sub-controllers, it should also activate and deactivate the controllers as needed – to respond to the driver's command)
- **R-9**. <u>Need to have some 'interface'</u> (***new in this context***)
  (This interface is needed to give the driver a friendly 'control panel' and language for expressing his commands. The role of the interface would be to translate qualitative terms such as 'hard', 'soft' or 'high', 'very low' to the quantitative values that can be passed to the appropriate controllers and actuators.)

Articulated solutions (solution models):

- **S-3**. <u>Controlled system with a higher-level controller + interface</u> (***new in this context***)
  (See the schematic in sketch G.)

[*end of context*]

- **<u>Context number 5</u> (DC-V.)**
  Articulated requirements (objectives):
  - R-3. <u>Need to measure the deflection of the wheel</u>
    (In order to control the shock absorber's behaviour we need to monitor the actual state of the road surface. This can be done by simply measuring the magnitude of deflection of the wheel against the chassis = compression or prolongation)
  - R-4. <u>Need to measure also the vibrations</u>
    (OK, wheel deflection is simple but not sufficiently robust, may lead to undesirable control actions. Therefore, we shall have another sensor independently monitoring the chassis behaviour against the road – i.e. vibration or a magnitude of vibration)
  - R-5. <u>Shock absorber rigidity must be adjustable!</u>
    (This is essential to have an 'active absorber', the shock absorbing properties must be adjustable in response to road conditions. Moreover – this should be easy to do!!)

– R-7. Controlling the clearance
(OK, when we want an auto- or manual mode for chassis adjustment, we need also some method for its control and an appropriate actuator.)

Articulated solutions (solution models):

– **S-4**. Details – physical input/output information and control flow (***new in this context***)
(See the labelled scheme sketched as H.)

[*end of context*]
[*end of design*]

Generated by MD discussion engine v0.1.

3.2 TASK-21: Control of paper-smoothing process

Design brief:
Design a control device for a paper mill that takes a coil of raw paper as input and ensures that paper at the machine output has desired thickness, is smooth and clean from any debris. Finished product is another coil of a homogenous paper free of any flaws (rippling, tearing, crumpling, ...)

- **Initial problem context (DC-I.)**
  Articulated requirements (objectives):
  – none

  Articulated solutions (solution models):
  – none

[*end of context*]

- **Context number 2 (DC-II.)**
  Articulated requirements (objectives):
  – **R-1**. Ensure that paper is ripple-free
  (Basic req. from a customer - final product should be a coil of a smooth paper, without ripples, folds, tears, etc.)
  – **R-2**. Ensure that paper is smooth and debris-free
  (Another basic req. from the customer: the final product should be 'polished)
  – **R-3**. Even thickness of paper
  (Another req. directly from customer - ensure paper has even thickness all along its length)
  – **R-4**. Detect the end of coil
  (To avoid paper tearing and/or mill damage, make sure to stop the whole procedure before the actual end of input coil occurs.)

  Articulated solutions (solution models):
  – **S-1**. Sketch of customer's desires
  (See sketch A on paper ...)

[*end of context*]

- **Context number 3 (DC-III.)**
  Articulated requirements (objectives):
  – R-1. Ensure that paper is ripple-free
  (Basic req. from a customer - final product should be a coil of a smooth paper, without ripples, folds, tears, etc.)
  – R-2. Ensure that paper is smooth and debris-free
  (Another basic req. from the customer: the final product should be 'polished)
  – R-3. Even thickness of paper
  (Another req. directly from customer - ensure paper has even thickness all along its length)
  – R-4. Detect the end of coil
  (To avoid paper tearing and/or mill damage, make sure to stop the whole procedure before the actual end of input coil occurs.)

- **R-5**. Dampening of paper before rolling (***new in this context***)
  (In order to comply with the technology, paper must be dampened before any further manipulation.)
- **R-6**. Rolling mechanism ... (***new in this context***)
  (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.)
- **R-7**. Drying after rolling (***new in this context***)
  (To store paper, we must ensure it is not damp after rolling => drying mechanism.)

Articulated solutions (solution models):

- **S-2**. Paper rolling principle (***new in this context***)
  (See sketch B showing the principle how to roll paper – simple solution.)

[*end of context*]

- **Context number 4 (DC-IV.)**
  Articulated requirements (objectives):
  - R-1. Ensure that paper is ripple-free
    (Basic req. from a customer - final product should be a coil of a smooth paper, without ripples, folds, tears, etc.)
  - R-3. Even thickness of paper
    (Another req. directly from customer - ensure paper has even thickness all along its length)
  - R-5. Dampening of paper before rolling
    (In order to comply with the technology, paper must be dampened before any further manipulation.)
  - R-6. Rolling mechanism ...
    (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.)
    - **R-6.1** Extended rolling mechanism ... (***new in this context***)
      (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. When having a sequence of these cylinders we may actually not only smooth the paper but also reduce its thickness when te gap between cylinders is decreasing ..)
  - R-7. Drying after rolling
    (To store paper, we must ensure it is not damp after rolling => drying mechanism.)

Articulated solutions (solution models):

- **S-3**. Principle for rolling and thickness adjustment (***new in this context***)
  (See sketch C for details - the principle is to re-use the mechanism in a sequence with decreasing gaps between the rolling cylinders.)

[*end of context*]

- **Context number 5 (DC-V.)**
  Articulated requirements (objectives):
  - R-1. Ensure that paper is ripple-free
    (Basic req. from a customer - final product should be a coil of a smooth paper, without ripples, folds, tears, etc.)
  - R-3. Even thickness of paper

(Another req. directly from customer - ensure paper has even thickness all along its length)

- R-5. <u>Dampening of paper before rolling</u>
  (In order to comply with the technology, paper must be dampened before any further manipulation.)

- R-6. <u>Rolling mechanism ...</u>
  (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.)
  - **R-6.2** <u>Rolling mechanism using alternately placed cylinder pairs...</u> (***new in this context***)
    (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. The sequence of cylinder pairs remains, just it won't be placed in one 'layer' but instead alternate - one up, next down, up ...)
- R-7. <u>Drying after rolling</u>
  (To store paper, we must ensure it is not damp after rolling => drying mechanism.)

Articulated solutions (solution models):

- **S-4**. <u>Principle for rolling and thickness adjustment – alternate cylinders</u> (***new in this context***)
  (See sketch C1 for details of this little modification in cylinders layout - the principle is to re-use the mechanism in a sequence with decreasing gaps between the rolling cylinders. We want to 'squeeze' the whole plant to a smaller space...)

[*end of context*]

- **<u>Context number 6</u> (DC-VI.)**
  Articulated requirements (objectives):
  - R-1. <u>Ensure that paper is ripple-free</u>
    (Basic req. from a customer - final product should be a coil of a smooth paper, without ripples, folds, tears, etc.)
  - R-3. <u>Even thickness of paper</u>
    (Another req. directly from customer - ensure paper has even thickness all along its length)
  - R-5. <u>Dampening of paper before rolling</u>
    (In order to comply with the technology, paper must be dampened before any further manipulation.)
  - R-6. <u>Rolling mechanism ...</u>
    (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.)
    - R-6.2 <u>Rolling mechanism using alternately placed cylinder pairs...</u>
      (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. The sequence of cylinder pairs remains, just it won't be placed in one 'layer' but instead alternate - one up, next down, up ...)
  - R-7. <u>Drying after rolling</u>
    (To store paper, we must ensure it is not damp after rolling => drying mechanism.)
  - **R-8**. <u>Assume that the thickness of paper is constant</u> (***new in this context***)
    (We can make this assumption that the thickness of paper on one coil is not changing (for the simplified approach when we adjust the gap once only and keep it as it was set, without the need to adjust).)

- **R-9**. Regulation according to the input coil (***new in this context***)
  (Taking previous assumption into account we may say that the parameters influencing the regulation will be given solely by the type/size of input coil.)
- **R-10**. Unrolling of input coil and rolling on the output (***new in this context***)
  (We need a mechanism that will ensure a smooth flow of paper through the cylinders and also the constant density on the output coil (which was one of initial customer's requirements).)
- **R-11**. Need of a motor to turn the output coil (***new in this context***)
  (Since we need to move the paper through the machinery, the easiest way is to power the output coil. So we need a motor or something able to turn the coil.)

Articulated solutions (solution models):

- **S-5**. Added regulation mechanism and paper movement (***new in this context***)
  (Paper will be moved through the cylinders using the output coil as a pulling and stretching device. For details see sketch D.)

[*end of context*]

- **Context number 7 (DC-VII.)**
  Articulated requirements (objectives):
  - R-1. Ensure that paper is ripple-free
    (Basic req. from a customer - final product should be a coil of a smooth paper, without ripples, folds, tears, etc.)
  - R-3. Even thickness of paper
    (Another req. directly from customer - ensure paper has even thickness all along its length)
  - R-5. Dampening of paper before rolling
    (In order to comply with the technology, paper must be dampened before any further manipulation.)
  - R-6. Rolling mechanism ...
    (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds.)
    - R-6.2 Rolling mechanism using alternately placed cylinder pairs...
      (The purpose of rolling mechanism is to produce even thickness of paper and remove all unwanted ripples and folds. The sequence of cylinder pairs remains, just it won't be placed in one 'layer' but instead alternate - one up, next down, up ...)
  - R-7. Drying after rolling
    (To store paper, we must ensure it is not damp after rolling => drying mechanism.)
  - R-8. Assume that the thickness of paper is constant
    (We can make this assumption that the thickness of paper on one coil is not changing (for the simplified approach when we adjust the gap once only and keep it as it was set, without the need to adjust).)
  - R-9. Regulation according to the input coil
    (Taking previous assumption into account we may say that the parameters influencing the regulation will be given solely by the type/size of input coil.)
  - R-10. Unrolling of input coil and rolling on the output
    (We need a mechanism that will ensure a smooth flow of paper through the cylinders and also the constant density on the output coil (which was one of initial customer's requirements).)
    - **R-10.1** Unrolling of input coil and rolling on the output (***new in this context***)

(We need a mechanism that will ensure a smooth flow of paper through the cylinders and also the constant density on the output coil (which was one of initial customer's requirements). We can achieve that quite easily when we change 'unrolling' and 'rolling' (thus refinining the paper flow) to: - pull paper off the input coil, - pull it towards the output through the whole machinery, - coil it tightly at the output)

– R-11. <u>Need of a motor to turn the output coil</u>
(Since we need to move the paper through the machinery, the easiest way is to power the output coil. So we need a motor or something able to turn the coil.)

– **R-12**. <u>Need of a motor for input coil</u> (***new in this context***)
(This is the result of solving a potential problem with tearing wet paper, we need something to pull the paper off the input coil, something that is not located on the coil itself (no pushing) but perhaps a little bit after it so that it can actually pull paper very gently but rather efficiently.)

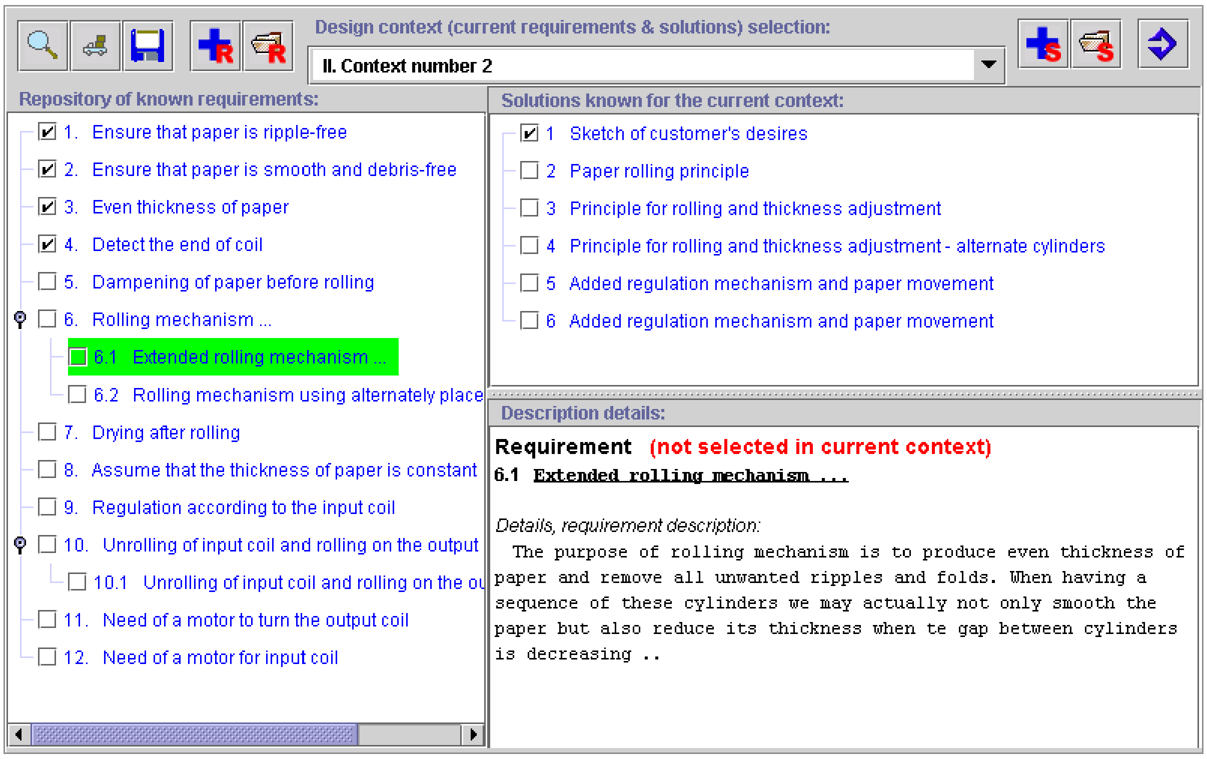
Articulated solutions (solution models):

– **S-6**. <u>Added regulation mechanism and paper movement</u>
(Paper will be moved through the cylinders using the output coil as a pulling and stretching device, and simultaneously there will be another pulling device closer to the input coil to ensure even unrolling and prevent paper tearing. For details see sketch D1.)

[*end of context*]
[*end of design*]

Generated by MD discussion engine v0.1.

Below is a snapshot of how the transcribed requirements and solutions looked like for context 2 in the actual tool, the designers had at their disposal.



Design context (current requirements & solutions) selection:
II. Context number 2

Repository of known requirements:
- ☑ 1. Ensure that paper is ripple-free
- ☑ 2. Ensure that paper is smooth and debris-free
- ☑ 3. Even thickness of paper
- ☑ 4. Detect the end of coil
- ☐ 5. Dampening of paper before rolling
- ☐ 6. Rolling mechanism ...
  - ☐ 6.1 Extended rolling mechanism ...
  - ☐ 6.2 Rolling mechanism using alternately place
- ☐ 7. Drying after rolling
- ☐ 8. Assume that the thickness of paper is constant
- ☐ 9. Regulation according to the input coil
- ☐ 10. Unrolling of input coil and rolling on the output
  - ☐ 10.1 Unrolling of input coil and rolling on the ou
- ☐ 11. Need of a motor to turn the output coil
- ☐ 12. Need of a motor for input coil

Solutions known for the current context:
- ☑ 1 Sketch of customer's desires
- ☐ 2 Paper rolling principle
- ☐ 3 Principle for rolling and thickness adjustment
- ☐ 4 Principle for rolling and thickness adjustment - alternate cylinders
- ☐ 5 Added regulation mechanism and paper movement
- ☐ 6 Added regulation mechanism and paper movement

Description details:
Requirement (not selected in current context)
6.1 <u>Extended rolling mechanism ...</u>

Details, requirement description:
```
  The purpose of rolling mechanism is to produce even thickness of
paper and remove all unwanted ripples and folds. When having a
sequence of these cylinders we may actually not only smooth the
paper but also reduce its thickness when te gap between cylinders
is decreasing ..
```

Figure 3–1. Snapshot of a design context management tool corresponding to task T21

4 TRANSCRIPTS OF THREADED JUSTIFICATIONS

4.1 TASK-11: Control of active shock absorption

Design brief:
Suspension and shock absorption are crucial elements in the car design, particularly with regard to the passengers' comfort. Design an active shock absorber and suggest a control strategy, so that the whole suspension mechanism would be able to adjust its properties according to the current state of the road. In addition to the absorption of shocks caused by an uneven road, we would like to adjust also the chassis clearance, so that the car has the optimal aerodynamic coefficients for a particular type of surface.

- **J-489**. Design brief
  Please, specify here any changes, amendments, suggestions to the initial problem spec.
  - **J-490**. Specification
    I. Automated adjustment of shock absorption (terrain)
    II. Manual adjustment of chassis clearance (for different terrain)
    III. Automated adjustment of clearance (??)
  - **J-495**. Active suspension = ?
    What we expect from an active shock absorber
    - **J-496**. Ideal state
      Ideally, the driver should not feel any bumps or changes of the road surface. The system should absorb and eliminate the uneven road -> intelligently.
    - **J-497**. Specification, principle
      We can take passive absorber and add some logic that can adjust its rigidity according to road surface. For smooth road we can have a bit tougher absorber (sporty); for bumps we need softer absorptiion.
      - **J-498**. Required components
        To control shock absorber we need a sensor measuring the degree of road 'unevenness', then some processing logic for deciding how to adjust the absorber, and then the actual absorber with adjustable rigidity/toughness.
    - **J-499**. Manual inputs
      Active suspension should have some input so that driver can set what he wants. This manual input will override the automatically adjusted value for the absorption (as logic calculated) .
  - **J-500**. Automated clearance
    This should behave so that when the load in a car changes, the clearance should be adjusted too. This adjustment should be automated - always or once?
    - **J-501**. Principles
      We can measure the actual clearance, logic would evaluate and some actuator would make the changes - to get into initial or recommended height...
    - **J-502**. Manual input
      Manual input may be request from the driver to change clearance immediataly -> e.g. change of terrain, off-road ride. Manual input will give the new value for maintaining the clearance.

[*end of thread*]

- **J-491**. Chassis clearance = const.
  Constant clearance should be a simpler case - assume we work with a given value, first.

- **J-492**. Suspension = const.
  If nothing changes in the absorber => plain, passive device
  - **J-494**. Nothing new
    This is useless to continue, we can forget any passive device ..
- **J-493**. Suspension - variable
  Active absorber should be adjustable
  - **J-503**. How it works
    When wheel gets on a bump, its position against the chassis changes - as in sketch B. Spring inside absorber should minimise the transfer of deflection on the chassis. Bigger bump = bigger deflection -> faster elimination is needed.
    - **J-508**. Just not spring...
      The problem is that the whole thing cannot be realised as a spring - it would rather tricky to adjust the toughness of springs -> need something that can be changed easier (hydro, pneu?)
      - **J-509**. Pneumatic piston
        OK,take a pneumatic piston instead of spring - the same behaviour, only rigidity is given by the volume/pressure of air inside the piston. That would need a simple pump to increase pressure and vent to release air (= softer abosrber)
  - **J-504**. Sensor + details
    Sensor measures delta between wheel axle and chassis. This can be a device measuring distance wheel-chassis placed on the absorber. Measured value goes to logic -> control alg.
  - **J-505**. Control algorithm
    This is the 'logic' that takes deflection of a wheel from normal position, and decides how to change the current toughness of the absorber.
    - **J-506**. Source of deflection/vibration (too hard abosrbers)
      I. With too hard springs we enter bumpy road -> immediately transfered onto chassis -> car vibrates => make absorber softer.
      - **J-507**. Source of vibration - too soft abs.
        II. Too soft suspension on smooth surface may also trigger vibration. => in this case, we need to make it harder, so that the car is less sensitive.
        - **J-512**. Ordering the measurements of wheel deflection and vibration
          The transition from smooth to bumpy road will first affect the wheel and a second later chassis. i.e. deflect BEFORE chassis
          When we go from bumpy onto a smooth road, the deflection is minimal/none but vibration still lasts, and it may even force the wheel to deflect unnecessarily, i.e. vibration and NO deflection (or better deflection AFTER vibration)
      - **J-510**. Minimise deflection - not good
        Measuring the position of wheel to the chassis is not good, because the ideal state is 'no deflection' = rigid rod... Of course, then we don't need any absorbers when we start with rods -> the bump has bad effect on chassis. so, this will need to be checked by the control logic - a max.limit on pressure, perhaps.
        - **J-511.** Need to measure vibration on chassis
          The wheel deflection can remain - simple measurement, but we need another sensor measuring the magnitude of the chassis vibration. In this way, we have two sources from measurement, they can validate each other...
    - **J-513**. Need to trace the order between deflection/vibration measurements
      So the rule can be
      IF deflection BEFORE vubration THEN decrease pressure in absorber

IF deflection AFTER vibration THEN increase pressure
(increase pressure = pump more air into piston (see sketch C))

[*end of thread*]

- **J-514**. SOL1: Controlling active suspension
  See sch. D., the chassis clearance is const. not included in the control alg./schema.
  - **J-515**. The rules extended
    The control logic has two simple rules that take as input the values measured on absorber (deflect/vibrate). one more thing that the controller will have to take care is that 'rigid rod' problem if trying to increase pressure too much, and the other thing is certain insensitivity in the rule firing...
    - **J-516**. 'Try & see' heuristic
      This could be a heuristic rule for the control of suspension incorporating the insensitivity region, so that the controller does not switch the pump on/off too often. When nothing's happening, we try to increase pressure a little, and see what effect that has on the deflection/vibration. If there is no effect or a positive effect (the vibration disappears) then continue with this heuristic. End it when you reach a kind of 'saturation'=it starts to worsen the vibraitons. The diff betwen the original pressure given by rule and this heuristic one, forms a region of absorber insensitivity
      This heuristic may be perhaps applied reversely, i.e. try to go softer, and see what happens. This could help on a terrain that becomes uneven 'continuously'...
      - **J-517**. Make it harder or softer first?
        The question is what shall be done first in a general case - try to increase the pressure or decrease it? This can be helped again by the order in which the disturbances are captured by sensor. So, if car enters a bumpy terrain, the wheel deflection shall be captured before the chassis vibration. In such case does not make sense to increase pressure any further, on the contrary.
        When car enters a smooth road from an off-road terrain, the chassis may continue to vibrate, so the pressure may be increased. We may perhaps measure some kind of dependency of the delay in the measurements, and use this as an estimate of what may happen in the next moments - prediction?!?

[*end of thread*]

- **J-518**. Clearance-variable, suspension-variable
  So far, we worked with constant and unchanging clearance of chassis from the ground; only the suspension was adjustable. We can try to take also the effect of chaning load distribution on the control... The clearance adjustment can be done mechanically, kind of screw going up and down - see illustr. E.
  - **J-519**. Relation clearance - suspension
    In principle, chassis clearance may be independent from the suspension - if there is certain minimal height maintained and if the load does not exceed some maximum. But suspension may be influenced by uneven distribution of load on board - the load may shift, e.g in bends or uphill...
  - **J-520**. Definition of adjustable clearance
    Clearance may be adjusted manually (command from driver) or automatically. In both cases, the adjustments can be done without caring for load distribution, and the right values could be tuned later - should be easier to tune than make big changes.
    In manual mode, the driver sets the requested clearance, the system changes it and maintains the car on this value (if poss.)
    In auto-mode, depending on the type of terrain (given by sensors used for suspension - vibration), we can set also the clerance
    - **J-522**. Manual mode principle
      Driver enters the value before starting the ride, this must be maintained during the ride.

- **J-523**. Adjust when stationary
  We may check the values of piston compression and screw extrusion before starting the ride. From these values we compute the current clearance. Now we can adjust screw up/down as needed to get requested clearance...
  - **J-524**. Stationary vehicle???
    Classical physics - vehicle behaves similarly when stationary and moving steadily with constant velocity... So shuld we allow to change the clerance when moving steadily? Then, when we have load aboard and it's loosely fitted, it may shift around. Change in the load distribution affects the wheels differently...
    Seems that we will need to adjust the clearance also while driving = more difficult, less safe???
    - **J-525**. Median of piston position
      Instead measuring piston/screw extrusion when stationary, we may use a median value for each piston as measured during a ride. In this way, we can adjust value of clearance also while driving. In certain small limits, this should be pretty safe...
- **J-526**. Automated mode
  This is an adjustment automatically responding to the terrain type - with changed suspension, a similar set of rules adjusts also clearance.
  - **J-527**. Translation table
    We can model how different terrains affect the effective behaviour of the piston, and how clearance influences this behaviour. From these findings we may construct a table that 'translates' the values of sensors into 'terrain types'. Each terrain type may be also assigned an optimal clearance; thus auto-mode would only grab the values from sensors, finds an appropriate value in the table, and send it to the actuator/motor.
- **J-521**. Implementation
  The clearance can be changed mechanically - using a screw connected to the absorbing piston and moving into chassis.
  it can be controlled by a small motor commonly available. Since, the wheel dimensions and screw pitch is known, it is easy to calculate how much the screw needs to be turned to effect particular clearance.

[*end of thread*]

- **J-528**. Chassis vibration sensor
  OK,let's have a look at how can we measure the vibration of the chassis - this is crucial for the control logic.
  - **J-529**. Sensor from airbags
    To measure the actual value of chassis vibration we can re-use the devices commonly appearing in the electronics for airbags. This is basically the same need, find out when chassis vibrates faster than it would 'naturally' => we found a bump => control action.

[*end of thread*]

- **J-530**. Piston deflection sensor
  This is a complementary measurement on the absorber that gives the actual position of a wheel agains chassis base.
  - **J-531**. Deflection = diff distance
    To measure deflection we may use a simple distance sensor, e.g. resistance-based measurement is pretty robust and precise. Also, we may a trivial incremental impulse generator or anythin simialr.

[*end of thread*]

- **J-532**. Adjustment of absorber
  To have an active suspension, we must be able to adjust the absorber's properties (rigidity or toughness) -> easily!!

  - **J-533**. Adjustment of pressure inside piston
    Whether we use pneumatic or hydraulic pistons, they both give the same fuctionality; they both contain media that can be compressed - perhaps air is easier compressible than oil or other liquids => so, change in air pressure with a constant volume affects the rigidity of the whole pneu absorber

---

[*end of thread*]

- **J-534**. Suspension control logic
  To implement the control, we have to interpret the verbal rules in a suitable 'machine' language
  - **J-535**. Hardware leaving aside
    The IF-THEN rules and heuristics seem to be pretty straightforward to translate to logical gates or perhaps, we can use a car computer, modern cars have it anyway, so we can use it for the active suspension as well

---

[*end of thread*]

- **J-536**. Clearance adjustment
  As we decided, this should be a simple, robust and safe mechanism that can withstand heavy loads and performs safely in higher speeds.
  - **J-537**. Jack/screw with an incremental motor
    The incremental motor is a good device because we won't need another sensor measuring the extrusion - we need only remember the current values and changes in the 'increments'...

[*end of thread*]

- **J-538**. SOL2: Control of suspension and clearance (together)
  This is basically an extension of the previous control loop, see sch. F.
  - **J-539**. Co-ordinating controller
    Seems that we need a 'master' controller that would coordinate the work of the specialised controllers - one specialises on suspension, the other on clearance
    - **J-540**. Control inputs for suspension ctrl
      On of the roles of 'master' would be translate the driver's commands into values recognisable by the sub-controllers. A kind of switch/input is needed to switch between manual and auto modes. Depending on this switch, the suspension controller can be overriden or left working.
      Also, this 'master' would translate a type of terrain into a value for the clerance controller, activate the heuristic rules, etc.
    - **J-541**. Control inputs for clearance adj.
      Similarly as with suspension, if the switch is in auto mode, then the sub-controller has full responsibility for determining the optimal clearance (tables, heuristics). Otherwise, in the manual mode, the driver specifies the requested value and if this can be achieved, the controller will maintain it - regardless of optimality; the rules defined earlier still apply, only using a 'master' controller, we may activate or deactivate any of them (as needed)

[*end of thread*]

- **J-542**. Controller for suspension/clearance
  hmm, see the SOL2...

[*end of thread*]

- **J-543**. Interfaces
  This is an extension if we want to support a comfortable communication of the driver's requests/commands to the controllers.

  - **J-544**. Information channel
    OK, this would be mainly a translator changing the intuitive values from the driver (e.g. from a dial setting softer/harder suspension) to the specific quantitative values for controllers (e.g. X kPa for pressure or Y cm for clearance)

    - **J-545**. Suspension controls
      This can be somewhat similar to the knob for A/C control in cars - say Position 0 = automated mode, Position 1 = very soft, Position 2 = soft, etc. (like a heater). Physically, we need two channels, first saying auto/manual (on/off or 1/0). Second would be carrying the actual signal corresponding to the qualitative adjustment (e.g. 0.25, 1.25, 1.75, ...)

    - **J-546**. Clearance controls
      Again the same principle - first, on/off = auto or manual. Another wire would transfer the coded value for a particular cleraance adjsutment (e.g. high = 40cm, medium = 30 cm, low = 20cm,...)
      Interface can be again constructed as a simple 'dial', it may be also digital, but that seems to be too complex to be worthwhile... but it's possible.

[*end of thread*]

- **J-547**. SOL3: Control system with 'user interface'
  Sch. F remains, only an interface is added, as above

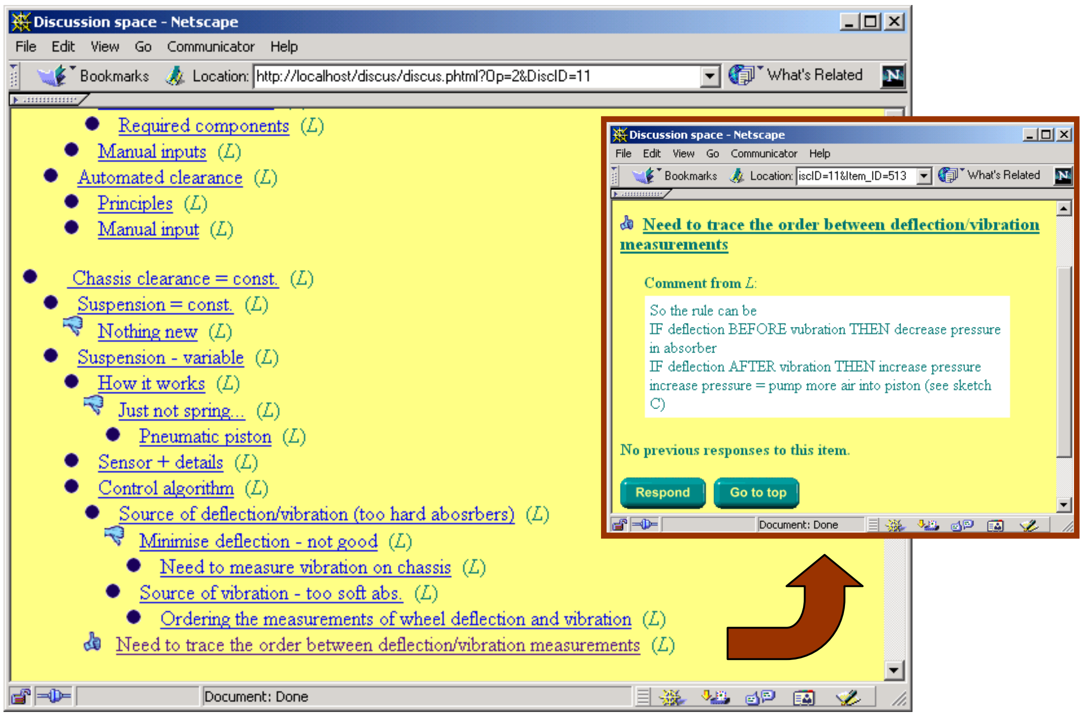
[*end of thread*]

- **J-548**. SOL4: Physical details and input/output flow
  See control the labelled sketch H (hope it's not too messy..)

[*end of thread*]
[*end of design*]

Generated by MD discussion engine v0.1.

This snapshot shows the transcribed justification space as the designer could have seen it.


Discussion space - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: http://localhost/discus/discus.phtml?Op=2&DiscID=11
What's Related
Document: Done

- Required components (L)
- Manual inputs (L)
- Automated clearance (L)
- Principles (L)
- Manual input (L)
- Chassis clearance = const. (L)
- Suspension = const. (L)
- Nothing new (L)
- Suspension - variable (L)
- How it works (L)
- Just not spring... (L)
- Pneumatic piston (L)
- Sensor + details (L)
- Control algorithm (L)
- Source of deflection/vibration (too hard abosrbers) (L)
- Minimise deflection - not good (L)
- Need to measure vibration on chassis (L)
- Source of vibration - too soft abs. (L)
- Ordering the measurements of wheel deflection and vibration (L)
- Need to trace the order between deflection/vibration measurements (L)

Need to trace the order between deflection/vibration measurements

Comment from L:

So the rule can be
IF deflection BEFORE vibration THEN decrease pressure in absorber
IF deflection AFTER vibration THEN increase pressure
increase pressure = pump more air into piston (see sketch C)

No previous responses to this item.

Figure 4–1. Snapshots from the justification threads record for task T11

4.2 TASK-21: Control of paper-smoothing process

Design brief:
Design a control device for a paper mill that takes a coil of raw paper as input and ensures that paper at the machine output has desired thickness, is smooth and clean from any debris. Finished product is another coil of a homogenous paper free of any flaws (rippling, tearing, crumpling, ...)

- **J-690**. Design brief (*this record was created automatically to start the design*)
  Please, specify here any changes, amendments, suggestions to the initial problem spec.
  - **J-692**. Technology sketch
    As an input we take a coil of (possibly) rippled, raw paper that should be transformed into a coil of paper with given requirements. For details, sketch A ...
  - **J-693**. Desired output
    What are desired parameters at the output?
    - **J-694**. Re: Desired output
      What are desired parameters at the output?
      --
      Paper should be smooth (no ripples), polished and of even thickness. Can we assume that the thickness is not changing significantly?
      - **J-695**. Re: Max. thickness reduction
        Paper should be smooth (no ripples), polished and of even thickness. Can we assume that the thickness is not changing significantly?
        --
        Yes, basically the maximal reduction should not exceed 15%, polishing and stretching are more important in this part of process.
  - **J-696**. Other restrictions from Customer
    Are there any other related restrictions - either from Customer or technology?
    - **J-697**. Re: Other restrictions from Customer
      Are there any other related restrictions – either from Customer or technology?
      --
      Perhaps, what about the weight and diameter of the coils?
      - **J-699**. Probably irrelevant
        Perhaps, what about the weight and diameter of the coils?
        --
        Seems to be irrelevant because the mass of paper cannot change during the process. Also, the diameter of the output coil will be at most equal to the input one - paper will be smooth = less space ...
  - **J-700**. Start & end of the process
    How does the process begin and how does it end? Any possible pitfalls there?
    - **J-703**. Re: Start & end of the process
      How does the process begin and how does it end? Any possible pitfalls there?
      --
      Before the machine is launched, there is a coil on the input, a small portion is unrolled and manually fed onto an output coil. The process should be finished before the paper is torn off the input coil!
      - **J-704**. Checking the end of coil
        ... The process should be finished before the paper is torn off the input coil!
        --
        Then I would suppose this is another requirement on the control device ...

[*end of thread*]

- **J-707**. SOL1: Initial sketch
  See sketch A on paper
  - **J-708**. Re: SOL1: Initial sketch
    Here, we have the desired function of the mill - now we can look at the details, what is actually involved.
    - **J-709**. Paper rolling
      Basic functionality is rolling, according to the technology, damped paper is better to shape, however we must ensure the 'right amount' so that it's not too wet.
      - **J-711**. Pre- and post-adjustment
        Basic functionality is rolling, according to the technology, damped paper is better to shape, however we must ensure the 'right amount' so that it's not too wet.
        --
        Before the actual rolling mechanism, we need some moisturiser... and again, the paper on output should be dry => after rolling we need a drying tunnel or something like that.

[*end of thread*]

- **J-717**. Smooth paper (damping, rolling, drying)
  To have paper even and smooth we suggest the technology of rolling that should remove all ripples, etc.
  - **J-718**. Paper damping
    Before any manipulation with paper it should dampened, afterwards it may be shaped.
    - **J-719**. Paper rolling and smoothing
      Before any manipulation with paper it should dampened, afterwards it may be shaped.
      --
      To achieve even paper we use a principle of rolling using a simple pair of two cylinders.
      - **J-720**. Paper drying and storing on a coil
        Before any manipulation with paper it should dampened, afterwards it may be shaped.
        --
        To achieve even paper we use a principle of rolling using a simple pair of two cylinders.
        --
        And finally, the reverse operation to dampening should be drying to comply with technology.

[*end of thread*]

- **J-721**. SOL2: Paper smoothing
  Taking into account all these basic requirements, we should take the simplest solution, as suggested by sketch B.
  - **J-722**. Re: SOL2: Paper smoothing
    Taking into account all these basic requirements we should take the simplest solution, as suggested by sketch B.
    --
    Works fine for the current reqs.

[*end of thread*]

- **J-723**. Extended rolling (incl. pressing)
  Now, we may attend the issue of reducing the thickness. As we learned in the disc. above, the thickness is not being changed significantly.

- **J-724**. Dampening
  Keep the dampening as with simple sol.
- **J-725**. Mechanism for pressing
  Basically the previous mechanism of smoothing works fine and may be reused for reducing the thickness as well – to avoid another dampening.
  - **J-726**. Sequence of pressing cylinders
    After rolling and smoothing but before drying we can introduce two-three pairs of similar cylinders. The sequence of cylinders can reduce the thickness.
    - **J-727**. Re: Sequence of pressing cylinders
      After rolling and smoothing but before drying we can introduce two-three pairs of similar cylinders. The sequence of cylinders can reduce the thickness.
      --
      This thickness reducing cylinders is in principle not different from the smoothing => they may be combined together.
      - **J-738**. Sequence of alternate cylinders
        To respond to a minor flaw of straight sequence of cylinders, we may require their alternate layout.

[*end of thread*]

- **J-728**. SOL3: Smoothing and thickness
  See sketch C for details of the layout.
  - **J-731**. Not sure of quality
    A little problem with this arrangement is that it may be rather long, which might be inpractical for maintenance and control. Also there may arise a danger of pressing debris into paper instead of removing them.
    - **J-733**. Squeezing cylinders
      A little problem with this arrangement is that it may be rather long, which might be inpractical for maintenance and control. Also there may arise a danger of pressing debris into paper instead of removing them.
      --
      We can 'squeeze' cylinders closer but again, there must be a little gap between them to ensure smooth operation. And probably that may not help much.
      - **J-734**. Re: Squeezing cylinders
        We can 'squeeze' cylinders closer but again, there must be a little gap between them to ensure smooth operation. And probably that may not help much.
        --
        Well, and what about 'squeezing' them in 2 dimensions - they don't have to be in a row, they may be one up, one down ... and paper may run through them.
        - **J-736**. Alternate layout of cylinders
          Well, and what about 'squeezing' them in 2 dimensions - they don't have to be in a row, they may be one up, one down ... and paper may run through them.
          --
          Sounds good and doesn't require a major change in the principles, just a little reshuffling of parts.
          - **J-742**. SOL4: Modification with using alternate cyl.
            See sketch C1 for details including that idea of 'squeezing' cylinders together.

[*end of thread*]

- **J-754**. Process of regulation
  Now we can look at the regulation itself
  - **J-755**. Assumption of constant thickness on output
    To start with we assume that the desired thickness of paper is not changing during the milling and is given at the beginning by the type of the input coil.

        - **J-756**. Regulation based on input coil
          Following this assumption we may design a simple direct control based on the parameters detected for the input coil.

    - **J-759**. Paper flow through cylinders
      The controlled device will be the output coil that must ensure both, the pulling of paper from input through the machinery of cylinders and rolling it smoothly on the output. This is the simplest case ...

        - **J-762**. Re: Paper flow through cylinders
          This is the simplest case ...
          --
          Attach a motor to the output coil and power it sufficiently to mvoe the paper and roll it tightly.

[*end of thread*]

- **J-765**. SOL5: Regulation and drive
  As sketched in Fig.D

    - **J-766**. Danger of paper tearin
      This solution satisfies the requirements however, paper when damp is a bit vulnerable concerning tearing - esp. if it's only pulled through a rather complex maze of cylinders

        - **J-767**. Both coils driven
          A solution to prevent paper damge would be perhaps powering also the input coil, thus having drive on two different locations and spread the 'tension' more evenly.

            - **J-768**. Re: Both coils driven
              But when the output is pulling, the input should be 'pushing' which may be tricky and may create more folds and ripples (paper is not sturdy enough)...

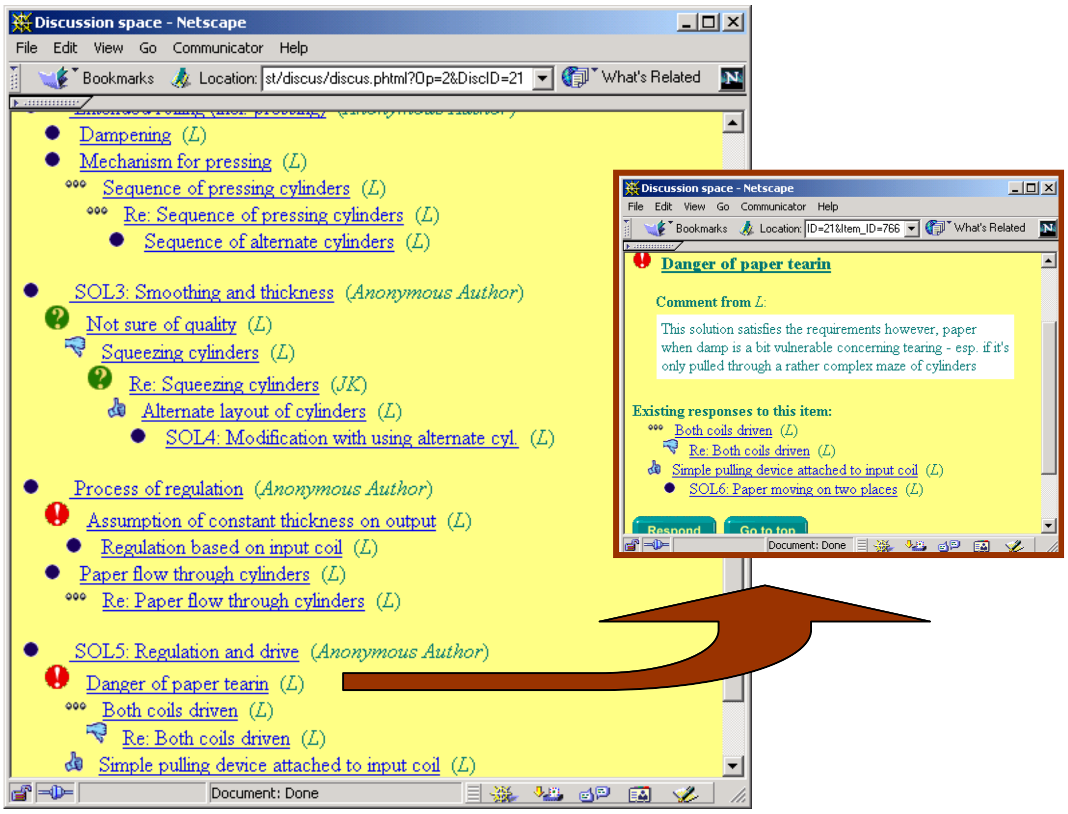
        - **J-769**. Simple pulling device attached to input coil
          We may try another alternative, where the paper is actually being pulled by a motor directly at the input, then fed in to a cylinder maze and pulled again at the end of the plant (output coil). This ensures that paper is actually pulled out when it is still dry, so the danger of tearing is reduced – we actually perform the task before the problem may occur.

            - **J-770**. SOL6: Paper moving on two places
              Sketch D1 shows a principle of pulling paper closer to the input thus reducing the danger of tearing.

[*end of thread*]
[*end of design session*]

Generated by MD discussion engine v0.1.

Below is a snapshot of the justification space as the designer could have seen it for task T21.


Discussion space - Netscape
File Edit View Go Communicator Help
Bookmarks
Location: st/discus/discus.phtml?Op=2&DiscID=21
What's Related
Document: Done
Discussion space - Netscape
Location: ID=21&Item_ID=766
Respond
Go to top

- Dampening (*L*)
- Mechanism for pressing (*L*)
  - Sequence of pressing cylinders (*L*)
    - Re: Sequence of pressing cylinders (*L*)
      - Sequence of alternate cylinders (*L*)

- SOL3: Smoothing and thickness (*Anonymous Author*)
  - Not sure of quality (*L*)
    - Squeezing cylinders (*L*)
      - Re: Squeezing cylinders (*JK*)
        - Alternate layout of cylinders (*L*)
          - SOL4: Modification with using alternate cyl. (*L*)

- Process of regulation (*Anonymous Author*)
  - Assumption of constant thickness on output (*L*)
    - Regulation based on input coil (*L*)
  - Paper flow through cylinders (*L*)
    - Re: Paper flow through cylinders (*L*)

- SOL5: Regulation and drive (*Anonymous Author*)
  - Danger of paper tearin (*L*)
    - Both coils driven (*L*)
      - Re: Both coils driven (*L*)
    - Simple pulling device attached to input coil (*L*)

Danger of paper tearin

Comment from *L*:

This solution satisfies the requirements however, paper when damp is a bit vulnerable concerning tearing - esp. if it's only pulled through a rather complex maze of cylinders

Existing responses to this item:

- Both coils driven (*L*)
  - Re: Both coils driven (*L*)
- Simple pulling device attached to input coil (*L*)
  - SOL6: Paper moving on two places (*L*)

Figure 4–2. Snapshots from the justification threads for task T21

References for Appendices

Barr, A. and Feigenbaum, E.A., Eds. (1982). The Handbook of Artificial Intelligence. California, USA, William Kaufmann, Inc.

Brazier, F.M.T., van Langen, P.H.G., *et al.* (1996). “Modelling an elevator design task in DESIRE: the VT example.” Int. Journal of Human-Computer Studies **44**(3): 469-520.

Brown, D.C. (1996). Some thoughts on configuration processes. AAAI 1996 Fall Symposium, Configuration Workshop.

Chandrasekaran, B. (1990). “Design Problem Solving: A Task Analysis.” AI Magazine **11**(4): 59-71.

Cohen, P.R. and Feigenbaum, E.A., Eds. (1982). The Handbook of Artificial Intelligence. California, USA, William Kaufmann, Inc.

Gero, J.S. (1990). “Design prototypes: A knowledge representation schema for design.” AI Magazine **11**(4): 26-36.

Goel, V. (1994). “A comparison of design and nondesign problem spaces.” Artificial Intelligence in Engineering **9**(1): 53-72.

Kumar, V. (1992). “Algorithms for Constraints Satisfaction Problems: A Survey.” AI Magazine **13**(1): 32-44.

Motta, E. (1997). Reusable Components for Knowledge Modelling. The Netherlands, IOS Press.

Motta, E., O'Hara, K., *et al.* (1996). “Solving VT in VITAL: A Study in Model Construction and Knowledge Reuse.” Intl. Journal of Human-Computer Studies **44**(3-4).

Motta, E. and Zdrahal, Z. (1996). Parametric Design Problem Solving, KMi, The Open University.

Motta, E. and Zdrahal, Z. (1998). A principled approach to the construction of a task-specific library of problem solving components. 11th Banff Knowledge Acquisition for KBS Workshop, Canada.

Simon, H.A. (1973). “The structure of ill-structured problems.” Artificial Intelligence **4**: 181-201.

Wielinga, B., Akkermans, J.M., *et al.* (1995). A Formal Analysis of Parametric Design Problem Solving. 9th Knowledge Acquisition for KBS Workshop, Banff, Canada.

Yost, G.R. (1992). Configuring elevator systems, AI Research Group, Digital Equipment Corporation.