A Knowledge Level Theory of Design and Engineering

Rüdiger Klein
Daimler–Benz Research Dept., Knowledge Based Engineering Group (FT3/EW)
Alt-Moabit 96a, D-10559 Berlin; Tel.: +49-30–39982211;
email: klein@dbag.bln.daimlerbenz.com

Abstract

Design and engineering are knowledge intensive processes. Large amounts of different types of knowledge are needed in real world applications. Using this knowledge in the design and engineering process will be essential for the success of new information technologies like internet/intranet, concurrent engineering, or virtual product design. An essential pre–requisite for the usage of this knowledge is an adequate framework which describes the role or roles each 'piece' of knowledge has to play in the overall context. This framework is provided by a knowledge level theory of design. In this paper we outline the essential elements of such a theory. Based on the main categories of knowledge relevant in the design process the roles these categories play and how they interact will be described in a general way.

Keywords

knowledge based design, concurrent engineering, knowledge level theory of design

1 INTRODUCTION

Currently the meaning of information technologies is rapidly increasing, especially in industry. World Wide Web, Internet, concurrent engineering, virtual product and virtual enterprises are key words characterizing this development. Design and engineering are knowledge intensive processes - thus knowledge based techniques should play an essential role here. But up to now their practical meaning is restricted. In order to realize the visions of concurrent engineering, virtual product development, and virtual enterprises in which knowledge is so important powerful knowledge based techniques will be indispensable.

G. Jacucci et al. (eds.), *Globalization of Manufacturing in the Digital Communications Era of the 21st Century*

There are many reasons for the current deficits in the application of knowledge based techniques in design and engineering. One is the huge amount of very different knowledge needed. That is not simply a matter of quantity (though this is an essential point, too), but mainly a matter of quality. Very different kinds of knowledge have to interact in design problem solving in a well defined and tuned manner. That's especially true for collaborative and distributed design and engineering: Different »agents» (humans and/or systems) have to cooperate, to understand mutually their respective knowledge, to communicate this knowledge for coordinated problem solving, for explaining rationales, etc.

In order to make that work we need a *general and abstract description* of how the different kinds of knowledge will be used in the design process, and how they interact in design problem solving. Therefore we need a *general* or *conceptual theory of design* describing the essential semantical issues of design and engineering problem solving on the knowledge level.

In our paper we will outline a *general knowledge level theory of design.* We will show how these general considerations effect main issues of knowledge based design and engineering:

- Knowledge representation: which are the main kinds of design and engineering knowledge, how can they be formulated, and how will they interact in problem solving.
- Knowledge modeling: design and engineering knowledge is (at least in parts) rapidly changing. Thus knowledge bases have to be adapted frequently to changed conditions. New 'pieces of knowledge' have to fit coherently into the recent knowledge base(s),and they have to respect the applied modelling scheme.
- Design and engineering problem solving methods: as different as the problems are the applied problem solving methods. Not all of them are 'knowledge based'. What is essential for knowledge based design and engineering is to *understand* and *represent* the meaning of these methods. This enables us to describe the interaction of the problem solving methods with the (object–level) knowledge they are applied to, and with other problem solving methods.
- User interactions: usually, design and engineering will not be fully automated (at least not in the foreseeable future and for many applications). The main part of the problem solving will be done by human users - *assisted* by intelligent CAD and CAE systems. Thus communication between humans and systems, and among systems is essential.

Each of these items depends critically on the framework provided by the knowledge level theory of design. In this theory we can describe how the various parts of knowledge will be used, how they inertact with other parts, how modelling and problem solving are interrelated, and how a user can interact with a knowledge based system in collaborative problem solving.

The paper is organized as follows: In chapter 2 we outline our knowledge level theory of design. This allows us to describe the main aspects of design knowledge and how they are interrelated.

This formal framework is a necessary, but not sufficient pre–requisite of knowledge based design. The next essential step toward this goal - shown in chapt. 3 - is to develop a knowledge level description of the design *process*. As will be shown this is not only an operationalization of the design theory. It immediately follows from the characteristics of design problems, especially their inherent complexity. This will be done in a way which reflects the characteristics of the design process itself. It will be shown how the design knowledge and meta–knowledge on one side and the characteristics of the design process on the other side influence each other.

Due to the complexity of design problems an automatic problem solving seems to be unrealistic. So, in many cases knowledge based design means intelligent assistance to a human designer. Therefore we need a tied cooperation between a human designer and a knowledge based intelligent design assistant. But how does this approch influence the design theory and its operationalisation? This will be discussed in chapt. 4. In chapt. 5 we give a discussion and summarize our conclusions.

2 A KNOWLEDGE LEVEL THEORY OF DESIGN

The knowledge level theory of design will be described in three steps: we start with a specification of the main knowledge categories in design (chapt. 2.1). In chapt. 2.2 the knowledge level theory of design will be given as a description of the roles and interactions these main knowledge categories play in design problem solving. Finally, in chapt. 2.3 we explain the role this knowledge level theory of design plays in knowledge based design.

2.1 The main knowledge categories in design

Though different descriptions are conceivable there are mainly three basic categories of design knowledge:

- a *general domain theory* containing generic object level knowledge about classes, relations, attributes, constraints, mechanisms, behaviours, etc. in the domain;
- *case–specific* object level knowledge about the concrete design problem; and
- *problem solving* and *control* knowledge.

These categories can be further refined: for instance generic knowledge describing the relations between structures, functions and behaviours, or generic knowledge about different levels of abstraction (on the structure and/or behaviour side), different views, etc.

In general, the knowledge in a domain theory will be very complex. Take, for instance, the extraction of behaviour from a given structural description and the applied mechanisms [CGI93, BGP94]. Such a behaviour may consist of very complex states and transitions between them (continuous or discrete).

The case–specific object level knowledge mainly consists of two basic categories: requirements and (complete or partial) design descriptions. Normally, the starting requirements will be functional, i.e., it will be described what the purpose of the thing to be designed will be (and maybe in which context). But also structural requirements, type restrictions to components, geometrical constraints, and other forms of consistency information (for instance cost limitations) may occur. In many cases the requirements will not be complete at the beginning of the design process. Not necessarily because anything has been forgotten (though this may happen, too), but mainly as a consequence of requirements evolving during problem solving (see also the description of the design process in the next chapter).

A design description consists of a description of the »designed thing»: the contained components with their types, their attributes (mechanical, electrical, shape, etc.), and their (mechanical, electrical, geometrical, etc.) relations. From this description the functions, behaviours, etc. can be deduced using the general domain knowledge.

2.2 The knowledge level theory of design

These considerations about relevant knowledge categories can be summarized (in a semi–formal way) as follows: we have

- a general domain theory D (containing definitional knowledge as well as various forms of general consistency information);
- a set R of requirements the thing to be designed should fulfill;
- a (complete or partial) design description M consisting of component descriptions, their attributes, relations, etc.

Usually, these properties also depend on the *context*[1] K in which the designed object exists and acts.

Now we are ready to formulate our *knowledge level theory of design.* To fulfill the necessary generality requirements this will be done in a very abstract way. For this

---

[1]. Formally, one can dscribe the interrelation between the designed object and the context it exists in in the same way as the relation between a system and its components. The physical laws etc. are the same. The main difference is that the context *not necessarily* has to be an artifact, too. It also can be a natural environment.

purpose we introduce four (meta–level) predicates which represent the main relations between the various knowledge categories.

First, we introduce the meta–level predicate 'complete' which allows us to characterize a set R of requirements as completely describing all necessary pre–conditions for the thing to be designed. Generally, this completeness depends on the domain and the context:

$$\text{complete}_{D,K}(R)$$

The predicate 'fulfills' relates a designed object (with its components, attributes and relations) to the requirements R. This relationship is mediated by the general domain theory D and holds in the context K. Thus, formally we say

$$\text{fulfills}_{D,K}(M,R)$$

with the meaning, that a design object described by the design description M fulfills the set R of (*all* structural, behavioural, and functional) requiremetns in a context K and where the generell domain knowledge D is relevant.

The predicate 'consistent' applies to a design description M in a context K if the consistency conditions in our domain theory D are satisfied:

$$\text{consistent}_{D,K}(M)$$

This notion of consistency has two aspects: a positive and a negative one. A design description must not contain contradictions, and it must fulfill all necessary (positive) conditions.

From these descriptions we come to the definition of the 'solution' predicate. A designed object with design description M is a solution to a design problem described by a set R of requirements (in a domain characterized by D and existing in a context K):

$$\text{solution}_{D,K}(M,R)$$

This allows us to give the core equation of our design theory:

$$\text{complete}_{D,K}(R) \wedge \text{fulfills}_{D,K}(M,R) \wedge \text{consistent}_{D,K}(M) \leftrightarrow \text{solution}_{D,K}(M,R)$$

i.e., that M describes a solution to the problem formulated by a complete requirement set R *if and only if* M is consistent and fulfills these requirements.

2.3. The role of the knowledge level theory in knowledge based design

This equation relates the main knowledge categories in design in a clear and abstract way on the *knowledge level* - without any reference to computations or even implementations. It also leaves aside the *problem solving* aspect: nowhere any

assertions are made about how to get the design description M from the requirements R, how to determine the fulfillment relation or the consistency of M, etc. Though, of course, these questions *are essential*, it is important to have a general description of design in this abstract way. It also provides the necessary guidance to answer the questions about problem solving, consistency considerations, knowledge representation, etc.

Beside guidance - what is the value of this knowledge level theory of design? The following issues are worth mentioning:

- The knowledge level theory of design is completely formulated on the *object level*. No reference is needed to problem solving methods, strategies, etc.
- This theory describes the semantics of design problems. It relates »input» (requirements) to »output» (the design description) - without any reference to *intermediate* problem solving states. Such intermediate states as well as transitions between them (problem solving steps) get their meaning only from their reference to the overall semantics described in the knowledge level theory of design. The 'fulfills' and 'consistent' meta–level predicates can get a well–defined declarative semantics by formalizing them in an appropriate calculus.
- For a given design problem there can be many solutions. The design theory allows us to discriminate solutions from non–solutions. A *ranking* between solutions (»the best one» etc.) can be made on the set of all solutions or on a subset of them. Seen in this way, ranking is a kind of meta operation - not directly being part of the design theory. A restriction of the solution set can be reached by adding stronger constraints which filter out those designs which do not fulfill them (which is described *in* the design theory).

3 A KNOWLEDGE LEVEL DESCRIPTION OF THE DESIGN PROCESS

3.1 Design states

The knowledge level theory of design described in the previous chapter relates a complete requirements set R to a complete and consistent design desciption M. This theory neither explains how to come to this complete requirement set nor how to generate a design description therefrom. Therefore, based on the knowledge level theory of design a knowledge level description of the design *process* will be formulated now. One could say that this is simply an operationalization of the design theory. But that's in the best case a very academic view. Only in very simple cases we can straightforwardly »map» the initially complete requirements to a solution. Typically, design problems are much more complex: they include many different aspects from different »sources» of knowledge (for instance, in concurent engineering), quite often we have conflicting requirements (to which no

solution can be found in a direct way), incomplete or tacit requirements, etc. Consequently, such complicated design problems need comprehensive requirements analysis, principle decisions in a conceptual phase, conflict analysis and resolution, simulations, hierarchical decomposition of functions and/or structures, etc.

This has significant consequences for the knowledge level description of the design process. First, we have to deal with incomplete requirement sets which gain *successive completion* during design problem solving. Second, the design process is inherently incremental. We have very frequently interactions between synthesis and analysis problem solving activities. Analysis may reveal new conflicts or result in new or modified requirements. Thus, third, the design process normally includes revisions and iterations.

The knowledge level theory of design formulated in Chapt. 2 is based on two basic categories of knowledge: the case independent domain theory D (and a context description K); and the case–specific parts of knowledge (the design problem characterized by the requirements R, and the partial or complete design description M). Taking the case independent parts D and K for fixed, a knowledge level description of the design process should be based on the two case–specific categories: requirements R and solution M. Consequently, we will formulate this process theory on pairs M.R of design descriptions and requirements.

At the beginning, a design problem is typically described by an initial set R_0 of start requirements (and sometimes a non–emtpy initial design description M_0). At the (successful) end of problem solving, we arrive at a state $M_f.R_f$ which is characterized by the knowledge level theory of design given in the previous chapter:

$$\text{complete}_{D,K}(R_f) \wedge \text{fulfills}_{D,K}(M_f,R_f) \wedge \text{consistent}_{D,K}(M_f) \leftrightarrow \text{solution}_{D,K}(M_f,R_f)$$

The design process describes the sequence of transformations starting with the initial and ending with a final state:

$$M_0.R_0 \longrightarrow \ldots \longrightarrow M_i.R_i \longrightarrow M_{i+1}.R_{i+1} \longrightarrow \ldots \longrightarrow M_f.R_f$$

In general. there will be (much) more than one such sequence ending in different solutions to the same original problem (which can be evaluated, ranked, etc.). Search and backtracking may be needed because not every intermediate state will be consistent.

3.2 Elaboration

In order to characterize the progress made during problem solving we introduce a measure of 'elaboratedness'. Generally, elaboration of design descriptions and requirements means to add more information to them, to make them more specific:

- to add new items (objects, relations, constraints);
- to make existing items more specific on their own: giving values to variables, restricting sets of possible values in finite domains or intervals, restricting types to subtypes, etc.

Normally, the design process progresses from less elaborated states to more elaborated ones. Elaboration can (and will) occur on both sides - the design description M as well as the requirements set R. Formally, elaboration means to add a set δM or δR, of new elements to the currently existing design description M or requirements set R:

elaboration: $M \longrightarrow M \cup \delta M$ or $R \longrightarrow R \cup \delta R$

There are two main aspects in design problem solving: synthesis and analysis. Both interact during problem solving in different ways. Synthesis essentially means elaboration of the design description M, analysis means elaboration of requirements R. Of course, synthesis as well as analysis take »the other side» into consideration: elaboration of design descriptions (i.e., synthesis) is done using the requirements set as a guideline; elaboration of the requirements set (i.e., analysis) is done with respect of the up to here generated design.

3.3 Synthesis and analysis

In the following the interactions between sysnthesis and analysis, between elaboration of design descriptions and requirements sets will be described in more detail. We start with the synthetic aspect:

The key point here is the fulfillment relation. This relation guides the synthesis process, i.e., the incremental elaboration of the design description. Given the initial state $M_0.R_0$ or an arbitrary state $M_i.R_i$ in the course of problem solving the decision about what to do next depends on this fulfillment relation: which of the requirements in R are already fulfilled, which are contradicted, and which are 'neither nor' in the current design description M. A design description M induces a complete partition on the requirement set R into those which are fulfilled (R^+), those which are violated (R^-), and those for which neither assertion can be made (R^0):

$$R \overset{M}{\longrightarrow} R+ \mid R0 \mid R-$$

First, we describe a standard synthesis step. Each such step has the objective to increase the number of fulfilled requirements - if possible without contradicting up to here fulfilled requirements. Typically, for this purpose the domain theory D will be used to find an elaboration δM of the current design description M which allows to fulfill a subsets of the unfulfilled requirements δR:

$$\text{synthesis:} \quad dR \overset{D, M}{\text{'''}} dM$$

which is done in such a way that the elaborated design description $M \cup \delta M$ fulfills the requirement subset δR:

$$\text{fulfills}_{D,K}(M \cup \delta M, \delta R)$$

Normally, this process is not deterministic: there can be many elaboration $\delta M^{(1)}$, $\delta M^{(2)}$, ..., $\delta M^{(n)}$ of a given design description M which allow to fulfill a subset δR of requirements. Quite often it is even not clear if and how they can be enumerated[2]. The selection of the subset δR of requirements to be fulfilled next as well as the selection of the alternative design elaboration $\delta M^{(i)}$ may strongly depend on decisions made earlier in the design process (i.e., on the set R_i of all requirements and the design description M_i generated so far in a problem solving step i).

Extending the solution normally includes propagation of the new information by various kinds of constraints.

These propagations also contribute to the second, the analytical, aspect of problem solving: we can analyse - by usage of the domain theory - the reached design state M.R and so further elaborate on the requirements set R which results in new requirements δR from the extended (partial) solution $M \cup \delta M$:

$$\text{analysis:} \quad M \mid dM \text{'''} dR^{D}$$

The new requirements will be added to the old ones ($R \rightarrow R \cup \delta R$). Also *emergence* can be described in this analytical way: new constellations, synergies, abstract views may result from such analytical processes.

3.4 Non–monotonicity of the design process

One of the essential and general aspects of design problem solving is its inherent non–monotonicity. This needs careful consideration in the formulation of the design process description. The elaboration of the design description M does not necessarily monotonically extent the set R^+ of fulfilled requirements.

---

[2]. related to the well known search vs. exploitation problem in design

Requirements which had been fulfilled previously can now be in conflict with newly elaborated elements in the design description. Take, for example, any mechanical system which fulfills certain functional requirements. If one adds some other components this may change the behaviour considerably - up to loosing the original functionality. Otherwise, conflicts can not necessarily be solved only by withdrawing some previous elaboration steps: sometimes further elaboration of the design description (for instance, adding some new components) may resolve the conflict. Or geometry: adding new geometric 'entities' to a given geometric object can result in a significantly changed new geometry. Many other aspects can contribute to the non–monotonicity of the design process.

3.5 Dependencies

The 'fulfills' and 'consistent' predicates apply - as desribed in Chapter 2 - also for subsets of requirments and design descriptions. This dependency information is essential to guide the problem solving process. Synthesis steps are performed in order to fulfill certain currently unfulfilled requirements. Thus, for each element in the design description it can be said for which purpose it was generated (i.e., to fulfill which requirements). Inconsistencies are caused by interactions of different parts of the design description. Using the fulfillment relation allows us to detect the conflicting requirements behind this inconsistency. The inconsistency can be removed by withdrawal of previous synthesis steps, or by relaxation of requirements, or sometimes by other repair operations. Explanations may be generated from these 'fulfills' and 'consistent' dependencies which contain, for instance, the rationales behind made decisions and their interdependencies.

3.6 The role of the knowledge level design process description

Inevitably, these considerations are very general and abstract. Some of the essential aspects of the design process (the interplay of synthesis and analysis, i.e. the elaborationof requirements and design descriptions; the incrementality and the non–monotonicity) have already been discussed. But the given knowledge level description of the design process allows us to draw some more conclusions about the design process and the underlying mechanisms.

- We did not make any prediction or commitment about the *kinds of knowledge* applied in a problem solving step. This knowledge may be very different from case to case, from domain to domain, from step to step. For instance, for a whole sequence of problem solving steps, only functional requirements may be analysed and refined. Then, at a sufficiently detailled description level, the functional specifications can be »mapped» to structures which fulfill them. Also the opposite way may be taken: to specify structures (in replay to some functional requirements) which then results in new functional requirements.

- We also did not make any assertion about the concrete form of the *problem solving methods* to be applied. Heuristic rules (maybe somehow »compiled» from deeper domain knowledge or experience) can be applied to map functional requirements to design descriptions which satisfy them. Parts from former cases may be retrieved in case based reaoning and adopted to the existing design description in order to make some more requirements satisfied. Hierarchical refinement, function to structure mappings using quailtative reasoning, etc. fit well into this general scheme.
- Conflicts of different kinds frequently occur in design problem solving. Some of them are hard, i.e. they express really impossible designs (which violate essential laws or rules). Others more or less reflect a kind of »nice to have» approach in design, i.e., optimality considerations of various kinds. Different solutions to a given design problem may be compared and evaluated (by Pareto analysis or simply by qualitative considerations). Conflicts can be solved in different ways: by propose and revise, by chronological or dependency–directed backtracking (i.e., undoing a design problem solving step), by withdrawal of violated requirements (or constraints), or by relaxation of them (depending on their kind). The non–monotonicity of the design process can be used here, too: if in a design state M_i a property p occurs which contradicts a certain requirement r in some cases also an *extension* of M to $M \cup \delta M$ may result in a *disappearance* or *change* of p (thus resolving the conflict with r).
- The knowledge level description of the design process gives a lot of freedom about how to come to a solution, i.e., which problem solving methods to apply. A subset of the unsatisfied requirements may be selected, one of the problem solving methods »somehow» applied to it, a modification of the design description generated therefrom, etc. Strategic knowledge of any kind can be formulated on top of this knowledge level description of the design process in order to control the process.

These issues also indicate how complicated the next steps in the formulation of a knowledge level theory of design will be (see also the discussion in chapter 5). In order to get practical, in the next chapter will be shown how a certain instantiation of this design process approach could look like.

4 AN INTELLIGENT ASSISTANT

4.1 The intelligent assistant paradigm in knowledge based design

The knowledge to be applied in design and engineering problem solving can be very complicated. Currently, there is no way to have all this knowledge in automated knowledge based design systems. What is really needed is an intelligent assistant which helps the human user(s) to deal with all the relevant information and knowledge. The following considerations are essential for the

conceptualization of the intelligent assistant approach: only parts of the relevant design knowledge will be available in the design assistant - the other, usually more complicated parts of the knowledge will stay on the human designer's side. Only the interplay of these two parts of knowledge will result in the desired functionality. Three main questions have to be answered to come to an intelligent assistant:

1. How to deal with incomplete knowledge on the intelligent assistant's side?
2. How to organize the interplay of the two parts of knowledge, i.e., the interplay between designer and intelligent assistant?
3. How to guarantee that the knowledge level theory of design and of the design process will be reflected by the system?

4.2 Incomplete knowledge:

Say, the relevant domain knowledge D is split into two parts: the knowledge D_{in} which is in the system, and the other parts of knowledge D_{out} which are not (available only on the user's side). For instance, in mechanical design all knowledge about mechanisms, functions, etc. may be left with the user. Then all requirements concerning such aspects have to be transformed into - equivalent - requirements and constraints on components, their attributes, structures, relations, etc. Formally, we can transform the original design problem characterized by a set R of requirements into an *equivalent* one R' by usage of the knowledge in D_{out}:

$$R \overset{D_{out}}{\text{'''}} R'$$

Now the *transformed* design problem R' can be solved using the knowledge in D_{in}. What is essential is that the transformation process may not be an isolated initial step: repeatedly new functional requirements may result from analysis processes during problem solving. This analysis can include steps which go the opposite way: from descriptions in terms of knowledge available to the intelligent assistant to those descriptions only understandable by the user.

The critical point here is to guarantee an adequate expressiveness of the intelligent assistant's knowledge D_{in}. Can we transform all those requirements not directly expressible in D_{in} into equivalent ones in D_{in}? Or do we need »hidden commitments» which can not be represented adequately in D_{in} and thus easily be overlooked during problem solving?

4.3 Interplay between user and intelligent assistant:

The intelligent assistant has to assist the user in problem solving. Thus the main control is on the user's side. He can control problem solving in various ways:
- make selections about which requirements to fulfill next;
- in which way they should be fulfilled
- he can add new requirements (following his analysis of intermediate problem solving states)
- he can analysis and resolve conflicts: by withdrawing former problem solving steps.

Also automatic and semi–automatic problem solving is possible. The assistant can search through large search spaces using intelligent backtracking. So also automatic proposal generation is possible.

4.4 The knowledge level design theory and process description:

Two points have been shown to be essentail in the design theory: requirement fulfillment and consistency of the design description. Both aspects are taken into account by the intelligent assistant. First, the assistant has to manage the set of currently not fulfilled requirements. Those requirements selected for elaboration are removed from this set, those newly appearing are added. Second, the intelligent assistant propagates all decisions, elaborations of the design description, etc. It also maintains the dependencies between expressions. In the case any of the elements in the design description is withdrawn all depending terms will be removed, too (which is, for instance, important for conflict resolution or for constraint relaxation).

5 DISCUSSION AND CONCLUSIONS

A knowledge level theory of design is an essential pre–requisite for knowledge based design. It provides the necessary framework for application–independent knowledge modeling and thus for the re–use of design relevant knowledge. Theoretical as well as practical progress will result therefrom. In this paper we outlined such a knowledge level theory. Starting with a characterization of the main knowledge categories we described their interaction in design by a set of meta–level predicates.

This design theory provides the platform for a knowledge level description of the design process. Though the semantics of design is clearly given by the design theory, the design process description is essential in order to handle the complexity inherent in general design problems. Our formulation of the design process reflects the semantics of the underlying design theory. It gives an incremental formulation

of the design process which includes synthesis and analysis and their interplay. It shows the inherent non–monotonicity of the design process. The various problem solving activities like choices, propagations, conflict resolutions, case adaptations, etc. can be formulated within this problem solving paradigm.

Many issues which are essential for design and engineering applications of the next future are supported by such a general knowledge level theory of design:

- *Conceptual and system design*: Whereas current CAD systems are mainly dedicated to *component* shape design, in many applications the *interrelations* between *systems* and their *components* are a main issue. Conceptual design, preliminary design on higher abstraction levels, concurrent engineering taking down–stream aspects into account, etc. need support by next–generation CAD systems.
- *Interactivity and flexibility*: Due to the complexity of many design and engineering tasks problem solving must be incremental and iterative (including revisions and variants and their evaluation), and distributed between cooperating partners. This implies to have systems which allow *incomplete* specifications, *revisions* of previous decisions, the formulation of *relations and constraints*, and the *propagation* of changes and revisions.
- *Knowledge integration*: One way to get more powerful support from CAD systems to human users is the use and re–use of various forms of knowledge in large KBE systems.
- *Retrieval, reuse, and adaptation*: Today many design tasks start from scratch - simply because retrieval of previous cases from drawing repositories or from more or less unstructured CAD data libraries is too complicated. In the future the *retrieval, reuse*, and *adaptation* of previously solved design tasks will be essential. Cooperative design may include varying partners which have to exchange specifications, fit their designs to user requirements, reuse and adapt older designs to meet new requirements, etc. Therefore much more information capturing design intent, decision rationales, various forms of constraints, conflict resolutions, etc. have to be handled.
- Geometric shapes and relations are an essential part in many design and engineering problems. Though features and constraints have been integrated successively into CAD systems, many requirements of knowledge representation are not fulfilled yet. A knowledge level description of geometry is essential for a closer coupling between CAD and knowledge based systems.

6 REFERENCES

[BGP94] Bhatta, S.R., Goel, A.K., and Prabhakar, S.: »Analogical design: a model–based reasoning», in: [Gero94].

[BKN94] M. Buchheit, R. Klein, W. Nutt: »Constructive Problem Solving: A Model Construction Approach towards Configuration», DFKI Report, 1994.
[Brazier et al. 95] Brazier, F.M.T., van Langen, P.H.G., and Treur, J.: A logical theory of design, in [Gero95], pp. 247-271.

[CGI93] Chandrasekaran, B., Goel, A.K., Iwasaki, Y.: »Functional representation as design rationale», IEEE Computer, 1/1993.

[Cutcosky93] Cutkosky, M.R. et al, »PACT: An Experiment in Integrating Concurrent Engineering Systems», Computer, IEEE Computer Society, Los Alamitos, Calif., Vol. 25, No. 1, January 1993, pp. 28-37.
[FCF 93] Fox, M., Chionglo, J.F., and Fadel, F.G., (1993), »A Common Sense Model of the Enterprise», Proceedings of the 2nd Industrial Engineering Research Conference, pp. 425-429, Norcross GA: Institute for Industrial Engineers.
[Gero93] J. Gero, F. Sudweeks (eds.): Proc. IFIP WG 5.2 Workshop on Formal Design Methods for CAD, Tallinn, June 1993.
[Gero94] J. Gero, F. Sudweeks (eds.): Proc. AI in Design Conf., Lausanne, Switzerland, Kluwer, 1994.
[Gero95] J. Gero, F. Sudweeks (eds.): Proc. IFIP WG 5.2 Workshop on Advances in Formal Design Methods for CAD, Mexico City, June 1995.
[Gruber93] Gruber, Th.: »Towards principles for the design of ontologies», Techn. report KSL 93–04, Stanford Univ., 1993.
[HT94] Harmelen, F. van, and Teije, A. ter: »Validation and verification of conceptual models of diagnosis», UNiv. Amsterdam, The Netherlands, 1994.
[HW93] Heisserman, J. and Woodbury, R.: Geometric Design with Boundary Solid Grammars, in: [Gero93], pp. 79 - 99.
[KBN94] M. Buchheit, R. Klein, W. Nutt: »Configuration as Model Construction: the Constructive Problem Solving Approach», in: J. Gero, F. Sudweeks (eds.): Proceedings of »Artificial Intelligence in Design 1994» (AID94) conference, Lausanne, 1994.
[Klein91] R. Klein: »Model Representation and Taxonomic Reasoning in Configuration Problem Solving», Proc. 15th GWAI, Bonn, 1991, Springer Informatik Fachberichte 285, pp. 182–194.
[Klein96] Klein, R.: G–Rep - Geometry and feature representation for an integration with knowledge based systems, Proc. of the IFIP 5.2 Workshop on geometric modelling in CAD, Airlie, Virginia, May 1996, to be publ. at Chapman Hall, London.
[Klein96a] R. Klein: »A Knowledge representation perspective on geometric modelling», in: W. Strasser (ed.): Proceedings of the Blaubeuren-II conference on geometric modelling, Tübingen, Oct. 1996, Springer Verlag, Berlin, 1996 (to appear).
[Klein97] R. Klein: »Dependency Maintenance in declarative geometry modelling», 4th ACM Solid Modelling conference, Atlanta, Georgia, May1997.

[Lloyd 90] Lloyd, J.W.: Computational Logic, Proc. of the ESPRIT Basic Reasaerch Activities Symposium, Bruxels, Nov. 1990, Springer, Berlin, 1990.

[Newell 81] Newell, A.: »The knowledge level», AI Magazine, 1(1981)1–20.

[Reiter87] Reiter, R.: »A theory of diagnosis from first principles», Artificial Intelligence 32(1987)57–96.

[Smithers96]Smithers, T.: »On knowledge level theories of design process», in: J. Gero, F. Sudweeks (eds.): Proc. Int'l. conference on AI in design, Stanford, 1996, Kluver, 1996.

[Takeda et al. 90] Takeda, H., Veerpkamp, P.J., Tomiyama, T., and Yoshikawa, H.: Modelling Design Processes, in AI Magazine 11(4) 37-48, 1990.

[Takeda 93] Takeda, H.: Abduction for design, in: [Gero93], pp. 189-210.

[WB86] Wielinga, B.J., and Breuker, J.A.: »Models of expertise», Proc. ECAI–86, pp. 306-318.

[WAS94] Wielinga, B.J., Akkermans, J.M., and Schreiber, A.T.: »A formal analysis of parametric design problem solving, Univ. Amsterdam, The Netherlands, 1994.

[WS94] Wielinga, B.J., and Schreiber, A.T.: »Reusable and sharable knowledge bases», Univ. Amsterdam, The Netherlands, 1994.

7 BIOGRAPHY

Rüdiger Klein got his Diploma degree and his Ph.D. from Humboldt University in Berlin in 1975 and 1979, respectively. After years of basic and applied research at the Academy of Sciences in Berlin he joined Daimler–Benz Research in 1991. His main fields of interest include knowledge representation, constraint logic programming, knowledge based design, and the integration of knowledge based techniques and geometry modelling. He is the (co–)author of more than 30 scientific publications, and currently member of the scientific program committees of the International AI in Design conference and the Solid Modelling conference.