KMi Technical Report

KMi–TR–84

Creative Designer:
What & how?

(Formal Research Proposal)

Martin Dzbor

Knowledge Media Institute
The Open University

Supervised by Dr. Zdenek Zdrahal and Dr. John Domingue

December 1999

CREATIVE DESIGNER: WHAT & HOW?

Intelligent Support for Problem Formalisation in Engineering Design

Formal Thesis Proposal

Martin Dzbor
Knowledge Media Institute, The Open University
Walton Hall, Milton Keynes, MK7 6AA
United Kingdom
M.Dzbor@open.ac.uk

Abstract:
Engineering design is a kind of human activity that makes use of many different knowledge sources. Basically, this may be a well structured, explicit, and domain specific knowledge or tacit, implicit, and experience-based knowledge. Each type of knowledge has its particular role in design, and thus in knowledge-based design support systems. In this report several issues concerning knowledge-based design support systems are reviewed and discussed. The document begins with a brief introduction of a design process and highlights some of its features. The introduction is followed by a brief review of the nature of design. Two important features of design that are relevant from the knowledge-centred point of view are discussed more deeply – creativity and a need for reflection. Later, the implications toward design support are drawn. Section III discusses the knowledge-based support for the design; first it reviews the related research, then introduces knowledge modelling as a technology, and finally gives an overview of knowledge-based design support as a problem of knowledge (re-)use. Further parts of the document present a new approach to the knowledge-based design support based on the common ontologies and analogous reasoning. Section IV gives a brief overview of the proposed approach, followed by Section V presenting preliminary design guidelines and Section VI presenting a scenario of their application. Document concludes with a discussion of issues raised in the text and suggests expected outcomes of the research.

Note: This technical report is based on the work in progress, and thus some strands of the research as proposed here, may be further developed, changed, or omitted in the future materials that will refer to this document.

November 1999

Table of Contents

I. INTRODUCTION

The aim of this document is to present a new view on the knowledge-based systems supporting design and designers especially in the early phases. Although the design support systems may be described and compared on many different levels, the knowledge-centred view in this document is emphasised by the three facets that describe the proposed approach: design knowledge representation, design process control, and analogous design cases retrieval and presentation. The document begins by introducing design and design support in Section I., followed by a brief review of the nature of design in Section II. Two important features of design that are relevant from the knowledge-centred point of view are discussed more deeply – creativity and reflection, and the implications toward design support are drawn. Section III discusses the knowledge-based support for the design: first related research, then knowledge modelling, and finally knowledge-based design support. Further parts of the document present a new approach to the knowledge-based design support based on the common ontologies and analogous reasoning. Sections IV and V give a brief overview of the proposed approach followed by Section VI presenting design guidelines and Section VII presenting a scenario.

Smithers, Conkie *et al.* (1990) suggest that a design task may occur when an agent determines to change the status of the surrounding world. The purpose of design is then to give a specification how these changes may be done. The process that leads towards such specification is called *design*. Archer (1970) sees design as a goal-oriented, problem solving activity; and Simon (1973) understands it as an example of an ill-structured problem to which a solution may not be found until significant effort to understand the structure of the problem has been made. It is thus possible to agree with MacCallum (1990) who claims that design must be viewed as an intellectual and knowledge-rich activity of an agent. These are only some of the reasons justifying the research activities in the area of knowledge-centred or knowledge-based *support* for the design.

There are many different definitions for a design task (Bhatta, Goel *et al.* 1994; Tang 1997), however, many of them agree that (engineering) design involves the use of scientific principles, technical information, and designers' imagination to define systems that perform desired functions. Schön (1983) points out that in practice designers are rarely presented with a detailed specification of design problems. Descriptions must be built up from uncertain design situations. Engineering design typically begins with initial requirements that are often vague and incomplete. These must be transformed to a consistent and complete description of a system. Designers must perform a certain amount of work to convert such uncertain situation into a set of soluble design problems. However, designers often know what they should do to achieve their goals, they might have acquired some experience that is helpful in tackling current design problems (Dominowski and Dallob 1995). Engineering design thus depends on both types of knowledge – *empirical*, experiential knowledge about previous design cases, and *theoretical* domain foundations together with design guidelines.

In engineering design several mutually related activities may be recognised: description of a problem, solution, evaluation etc. All of them can be supported in one way or another. Traditional CAD systems were mostly concerned with the support for the solution phase with the description phase being undervalued (MacCallum 1990). Finding a relevant description of requirements to a design problem is equally important part of design process as the solution (construction of system specification). Generation of requirements and solutions are interactive activities in terms of knowledge they use and provide (Smithers, Conkie *et al.* 1990; Nidamarthi, Chakrabarti *et al.* 1997). Many computer-based design systems developed in so far were mostly concerned with a one-directional use of available knowledge; e.g. to check whether current solution satisfis current requirement without suggesting the next step. Some systems supported case retrieval and simple adaptation, but were able to work only in a very specific and limited domain (Bhatta and Goel 1994; Bhatta, Goel *et al.* 1994; Tang 1997; Watson and Perera 1997).

In my research I would like to investigate, how designers might be supported during one particular design phase, namely *problem formalisation*. This phase may be defined as an iterative, goal-oriented process transforming the initial requirements to a set of consistent and complete descriptions of a design problem at hand. Basically, two activities will occur throughout the problem formalisation: *recognition* and *description* of design problem in form of requirements and solutions (Dzbor 1999a). I presume in the following parts that the above-defined design phase may be formally supported although it is a highly creative activity. The approach discussed further is based upon knowledge-level models that can be designed to support different types of knowledge, 'in-domain' and 'across-domain' experience, theoretical knowledge, etc. A knowledge-level model as used further is a representation of an agent's knowledge that is relevant to a particular subject matter and may explain agent's actions (during the design process) (Newell 1982).

First, let us start with a brief description of the nature of problems in design as viewed from the cognitive perspective, including some illustrative examples and implications toward design support.

II. NATURE OF PROBLEMS IN DESIGN

As mentioned above, design is a knowledge-rich activity and designers often draw upon their previous experience from solved design tasks when they tackle initial, uncertain design situations. Uncertain situations must be converted into soluble problems, i.e. complete descriptions of the requirements and of the designed system. The conversion will be an iterative process. Schön's (1983) understanding of a design phase that corresponds to the problem formalisation sees design as a process, in which designers interactively name objects, to which they will attend, and frame the context, in which they will work. Before we say anything about design systems and techniques several selected features of the design process will be discussed and implications for the knowledge-based design support systems will be set out in the following sections.

A. Creativity and its role in design

Design tasks are often divided into several distinguished categories: (i) *routine*, (ii) *innovative*, and (iii) *creative* (Qian and Gero 1996). The difference between the different design tasks is in what is known about the task in advance, what is given, and what must be found. Routine design tends to instantiate a given set of variables, whereas creative design typically expands such a set. Innovative design tasks are located between these two extremes – they use the existing variables in a novel way. Altshuller (1984) distinguishes as many as five levels or categories of invention within the design tasks. His categories range from the simple changes and improvements to a technical system (levels 1, 2), through the innovative applications of the existing knowledge to new areas (levels 3) to the discovery of new phenomena (levels 4, 5). Altshuller also claims that a significant majority of design tasks belongs to the three lower levels.

However, both categorisations reveal that there are at least two contradictory understandings of what creativity and innovation are. It may be a design process and technique that qualify as innovative, or it may be a final product and its deployment in a particular domain. As further paragraphs show, in my research I am more inclined towards the latter understanding of creativity and innovation. It follows from this understanding that an innovative product does not have to be produced by an explicitly innovative method or technique. An existing method when applied in a novel way may be perceived at the end of the day as producing an innovative product. As Watson and Perera (1997) point out, creativity in this sense is much more a social than a technical issue and it is the society that judges what is creative and what is not.

Cognitive scientists studying the problem solving activities in design warn that designers may often fail to solve given problems due to a fixation on a familiar interpretation of the situation at hand and literal re-use of a previously applied approach (Dominowski and Dallob 1995). Because of the undesired fixation on familiar interpretations, they are often unable to approach the problem in an innovative or creative way. Fixation, however, might be avoided by an *insightful design* that is based on a sudden understanding of a problem, especially by designers' intuition. Cross (1997) in his study justifies an idea of 'creative leaps' based on the observation that many innovative design concepts emerge suddenly, almost in a form of 'enlightenment'. Creative leap is defined as a sudden perception of (or a shift to) a new perspective of viewing the situation.

Cross describes designers working on a bicycle rack and reports that they were guided by the initial requirements and recognition of the problem together with their experience. When they attempted to solve the problem as specified initially, they soon recognised new issues that had to be included into requirement description and problem recognition. They proposed a solution in early phases of design but subsequent refinements of this approach did not remove an important problem of mud spray thrown by the rear wheel. They 'played' with the suggested alternative solutions and the essential breakpoint came when they looked at the rack as 'a tray' or 'something like a bag'. This newly identified concept caused a shift in their perspective and helped them to arrive at a sound solution.

According to Cross, the familiar situation, which initially suggested a routine approach, eventually gave rise to a creative and innovative design. Cross has suggested several models to describe how such innovation may occur; the most common models are by *combination*, *modification* and *analogy*. All models require good knowledge of the existing design domain. As further sections show, these models of creative (or at least innovative) thinking may be supported by a well-structured knowledge base. This assumption is also supported by another study of creativity in design that reveals that the key element in the creative design is designer's act of insight by which certain focuses are chosen and combined in an innovative way (Candy and Edmonds 1996). Design process is a unique case representing an *extension* to knowledge currently possessed by a designer. It is the designer's ability to make analogies with parts and products in other areas that may cause the extension of otherwise traditional design space that typically leads towards routine solutions. Candy and Edmonds in their study emphasise the importance of deep domain knowledge and immersion into this knowledge. As shown further, deep knowledge and analogy finding are considered to be significant features of design tasks. Further, the means providing support for these features by a set of hierarchically organised knowledge models that build a framework for design and domain knowledge bases, is discussed in details.

B. Reflection and its role in design

Reflection is a term introduced by Schön (1983) to explain the nature of non-trivial problem solving. As mentioned above, Schön sees problem formalisation as an iterative process. It means that one may change the current perspective (frame) when this does not suit the design situation satisfactorily. This change will cause new objects to be identified within the situation, which may further lead to another change in perspective. The need for changing the current perspective (frame) is usually caused by an unexpected result in the current perception of a problem. Schön refers to this need as 'a surprise' and claims that in theory any result inconsistent with designers' theoretical and/or empirical knowledge might be perceived as 'a surprise'. When designers find something surprising (either positively or negatively) in the design problem description based on their current perspective (frame), they *reflect* on the description made so far.

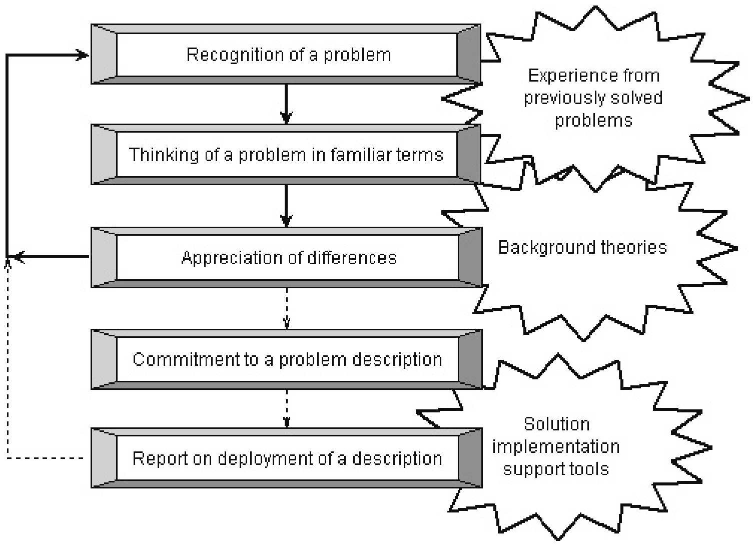

Recognition of a problem
Thinking of a problem in familiar terms
Appreciation of differences
Commitment to a problem description
Report on deployment of a description
Experience from previously solved problems
Background theories
Solution implementation support tools

Fig. 1. **Problem formalisation and reflection**

Reflection process applied on the design problem formalisation is illustrated in Fig. 1. First, the problem is recognised from given initial requirements and using designers' domain knowledge. Knowledge and experience re-use might be performed through thinking about the current problem in terms of a familiar case solved in the past. Such familiar and similar cases may be retrieved from the design repository, and differences between the current and previous cases might be perceived. Any such difference means that current description is probably inconsistent with existing knowledge or is incomplete. Designers may reflect on their current understanding of a problem, identify the causes of inconsistencies, possibly change their current frame, or suggest modifications to achieve consistency. The process of frame modification is called re-framing, whereas the process of minor modifications to the current description will be referred to as re-formulation.

Reflection during problem formalisation is shown as a full-arrow loop in Fig. 1. Modification and re-use of a familiar case is depicted as a commitment to a particular description. Later it may show that even the modified description did not help and reflection must occur again to change the initial requirements and frames. Such an approach mirrors well the philosophy of co-evolution of requirements and solutions (system descriptions) as mentioned in the introductory section (Smithers, Conkie *et al.* 1990; Nidamarthi, Chakrabarti *et al.* 1997). The evolution of both, requirements and solutions is also implied by the ill structure of the design problems especially in their initial phases when the initial requirements are often incomplete, ambiguous and contradictory. The role of design process is then to not only propose a solution satisfying these requirements, but also amend and complete the requirements.

C. Implications towards design support

The section on creativity in design might have implied that successful innovative design is often a matter of designer's mastery and artistry when it comes to converting the initial requirements into final description of requirements and solutions. However, as it was shown this 'artistry' highly depends on knowledge that is available during design and is affected by the structure of available knowledge bases. Knowledge bases (KB) may be present explicitly, for instance in a form of computational KB or implicitly, in a form of designer's knowledge and experience that is 'stored' in the designer's head. The role of knowledge bases (KB) for the design is thus twofold:

1. KB serves as a source of knowledge for the problem description; a *vocabulary* of terms that may be used to communicate design problem description
2. KB serves as a *reference* vocabulary supporting and enabling retrieval of familiar/similar cases and their modification (knowledge re-use)

These roles correspond clearly to various activities that typically occur in design, particularly in case-based design (CBD). A typical model of CBD (Watson and Perera 1997) contains the following phases: (i) retrieval, (ii) re-use, (iii) revision, and (iv) retention. The role of KB as a vocabulary suits very well to phase (i) – to describe the current case and retrieve the similar one, as well as phase (iii) – to re-use the previous case appropriately and consistently. The role of KB as a reference addresses all four phases in CBD; all four phases must somehow refer to a shared 'index' to achieve their goals.

To summarise the cases reviewed in previous sections and draw some implications toward design support see Table 1 (Dzbor 1999b). It identifies the corresponding steps within the different researchers' understanding of design process. The comparison is based on the functional similarity of design procedures recognised in their specific case studies. Table 1 shows possible mappings between the design steps as different researchers define them. From the observations Cross, and Candy & Edmonds did in their particular case studies, it is possible to conclude that *the most promising concept selection and development* (in Cross's terms) is very similar activity to *the use and iteration* (in Candy and Edmonds's terms). Especially, the further development of concepts and iteration can be associated with Schön's reflection on design actions. The research activities reviewed in the table do not have much in common on the first glance; however, it is possible to find some pattern here. As it is visible, it is even possible to include Altshuller's work into the pattern that was built by the other researchers. Table 1 attempts to associate different understandings of design process. Guidelines discussed further in this document are based on the design steps associated in the table below.

Table 1. **Steps of engineering design processes (Comparison)**

	Schön	**Cross**	**Candy, Edmonds**	**Altshuller**
Step 1	*Recognise a problem*	*Explore the problem*	*Make*	*Analyse problem*
Step 2	*Name concepts*	*Write performance specification*		*Build problem model*
Step 3	*Frame problem in familiar context*	*Generate range of concepts*	*Design*	*Formulate contradictions*
Step 4	*Appreciate differences*	*Evaluate*	–	*Use table of contradictions and basic (inventive) principles*
Step 5	*Experiment in virtual worlds*	*Select the most promising concept*	*Use*	*Apply principles in the current problem*
Step 6	*Reflect on action*	*Develop concept using creative models*	*Iterate*	*If necessary, re-formulate problem*
Step 7	*Re-frame the problem*			

Another conclusion implied by the reviewed studies (Altshuller 1984; Candy and Edmonds 1996; Cross 1997; Schön 1983) shows that the design is a knowledge-rich activity. Various researchers were using different terms to express their opinions that designers must have a deep knowledge and understanding of their particular domain and must be really immersed in this knowledge. Then they may expect to be able to see analogies and similarities among products in different areas. And such analogies, as Candy and Edmonds claim, can extend the otherwise traditional (routine) design space and foster creative and innovative solution of design tasks. Moreover, this is the reason why the knowledge-based approach as proposed later in this document is described around three facets:

- what knowledge is useful for a design support system and how to represent different knowledge,
- how to present this knowledge to the designers,
- and how to support the interactions between the computational support system and the designer.

In accordance with the design steps in Table 1 and the roles of KB mentioned at the beginning of this section the design task may be formulated as a triple $F^k = \text{á}R, S; H\text{ñ}$ Symbol S stands for a description of the designed product (solution); R is a set of requirements on the product; and H is the history of a particular design case (a sequence of formulation leading toward the current state). Formulation F^k representing the current step of a design problem DP^{k} is derived from the initial requirements R^0, actual designer's assumptions A, and expectations E. Problem formalisation may thus be formally defined as a mapping ρ as follows:

$$DP^{k} = \text{á}F, A, E\text{ñ}^{k} = \text{á}R, S, H; A, E\text{ñ}^{k}, \qquad (1)$$

$$DP^{k+1} = \rho\,(DP^{k}) = \rho\,(\text{á}R, S, H; A, E\text{ñ}^{k}). \qquad (2)$$

Equations (1) and (2) partially reflect the nature of design problems as they were introduced in section II. They relate requirements, solutions, and design case history in a single structure called *formulation,* and define problem formalisation as a mapping ρ that changes current understanding of a problem *DP*. Mapping ρ might be thus understood as a *re-formulation operator* (Choueiry, McIlraith *et al.* 1998). However, in the previous paragraphs a broader term 'mapping' is used because ρ might serve also as a re-framing operator[1].

An important part that was not mentioned in the equations above are knowledge bases that can be used to find and apply an appropriate operator ρ. There have been several attempts to construct knowledge bases that will help to tackle a design problem. Many approaches were essentially rule-based. For instance, Altshuller (1984) in his theory of inventive problems solving (TRIZ) claims that any solution to an inventive problem is determined by objective laws. He further suggests that it is be possible to solve any such problem with technical systems using a combination of well-defined rules – *inventive principles*[2]. He developed a base of several such principles that are invoked when certain contradictions between selected design variables and parameters occur. Although we do not have to agree with all his opinions, his main achievement is in the introduction of a limited set of generic design rules. The 'inventive principles' can be perceived as a generic, task-oriented, 'across-domain' knowledge base that contains *heuristic rules* or best practices. These may (i) improve the navigation in the available design space; (ii) explain the designers' actions and decisions; and finally (iii) help the designers to understand the relationship between (possibly contradictory) requirements and solutions (necessarily sound and consistent). Various perspectives and issues concerning knowledge bases for design and their construction will be discussed more in-depth in the following section.

III. KNOWLEDGE-BASED DESIGN SUPPORT

Design systems for the engineering and technical problems have been developed basically in two parallel directions:

1. The automated design systems, also called intelligent CAD (MacCallum 1990) whose aim was full or partial automation of the design process, and where a role of human designer was to type initial requirements, evaluate solutions, build prototypes etc.

[1] Choueiry *et al.* use only re-formulation operator for their purposes of explanation of engineering systems; however, the same theory applied to design requires some additional operators to generate new formulations. Details are discussed further on.

[2] Altshuller's general inventive principles include, for instance, those of segmentation or merging of objects, preliminary actions, addition of a dimension, phase transition, preference of short-lived cheap objects and many more. To solve an inventive design task several principles may be applied and combined. A sample implementation of a principle of segmentation may be the use of Venetian blinds instead of solid curtains or shades. The individual slats in Venetian blinds are connected together by a string so that it is possible to open or close them all at the same time (principle of merging).

2. The design support systems that aimed at assisting human designers in their tasks either by recalling past cases (Watson and Perera 1997), critiquing and navigating (Fischer 1992), reasoning and consistency maintenance (Smithers, Conkie *et al.* 1990; Tang 1997), etc.

Engineering design has been identified as a goal-oriented intellectual and knowledge-rich activity (MacCallum 1990; Smithers, Conkie *et al.* 1990). There have been several attempts to develop a computer-based system supporting in one way or another designers and the design process. For a further reference see, for instance the following works (Smithers, Conkie *et al.* 1990; Tang 1997; Watson and Perera 1997). Knowledge-based design support system (KBDSS) may be basically defined as "*a decision support system to enable designers to explore the structures of design problems and their solutions by combining human design expertise with domain and design knowledge that might be stored in a computational system*" (Tang 1997). As seen from the definition, one of the aims of KBDSS is to assist designers in the exploration of structure of both design process and domain knowledge. KBDSS thus need means that would clarify the structure and simplify the exploration process. The next subsection justifies the application of knowledge models as a core technology for the knowledge base construction in a KBDSS.

A. Related research

In this section, different approaches towards design support will be reviewed, including their underlying architectures and philosophies. This review is rather informative but not exhaustive; the aim is to address alternative research directions within the field of intelligent support for the engineering design.

1. Review of design support philosophies

The *Edinburgh Designer System* (EDS) is a knowledge-based design support system developed at the University of Edinburgh as a part of their 'AI in Design' Programme (Smithers, Conkie *et al.* 1990). EDS supports the work in multiple design contexts in mechanical engineering domain. EDS is based upon two basic pillars: module class definitions and assumptions-based truth maintenance system (ATMS). Module class definition represents a generic model of a particular device (e.g. a turbine) that includes all descriptive variables, constants, parameters, constraints, etc. The design process control is performed through ATMS and initial designer's assumptions. EDS has a typical blackboard architecture where current design case is described in form of assumptions and inferences made by both – the designer in order to pursue particular direction in design, and EDS in order to initialise module class definitions and propagate them. EDS is able to utilise different knowledge sources to infer new datum as a consequence of given assumptions.

In EDS crucial decisions (assumptions) are made by human designers, EDS supports their work by propagating and inferring new knowledge. Human designers are also in control of the entire design process and are allowed to explore multiple alternative approaches. ATMS maintains contradictory assumptions in separate contexts that enable designers' exploration of a complex design space. Through assumptions assertion and maintenance EDS is partially able to address the issue of co-evolution of requirements and product descriptions that together with a sequence of decisions in a particular design case construct a Design Description Document.

On similar principles is built a system called *Integrated Functional Modelling* (IFM) originating at the University of Cambridge. IFM is a further enhancement of the above mentioned architecture with some additional support tools (Tang 1997). Design knowledge base contains a hierarchy of components and relationships among them, sets of constraints and a detailed description of the components. The IFM starts with a formation of solution concepts that lead towards conceptual product model. Such model contains abstract information in terms of functionality and topology that satisfies given requirements. Conceptual model is further transformed into constrain-based model to search for a satisfactory embodiment solution. Search is performed through constraint satisfaction and propagation techniques.

IFM architecture thus combines rule-based and object-oriented knowledge representation into a system supporting work in multiple contexts and concurrent design environments. Designer's actions are treated as assumptions upon which system may perform some inferences.

Slightly different approach towards design support is visible in Goel *et al.*'s work on KRITIK, IDEAL and E-KRITIK systems, e.g. (Bhatta and Goel 1994; Prabhakar and Goel 1998). Goel *et al.* built their design support system on case-based reasoning principles and developed a specific scheme for design problem representation called Structure–Behaviour–Function (SBF) model. Both latter mentioned systems (supporting also 'across-domain' analogies) were constructed upon experiences from the former support system that was working with 'in-domain' analogies. For all systems, a design case contains (i) functions that must be delivered by the product, (ii) structure of the product, and (iii) causal behavioural (SBF) model of the product. Similar cases are retrieved on a function-to-structure basis where functions serve as indexes in a knowledge base. Previous SBF models may be adapted for a new design case based on the functional similarity. IDEAL is also able to learn abstractions from several similar design cases attaining the same functionality.

The underlying technology of all mentioned systems are common ontologies developed by Chandrasekaran *et al.*, functional modelling and case-based retrieval and adaptation of previous design cases. Especially IDEAL combines very efficiently case-based and model-based approaches.

Fischer and his team from the University of Colorado at Boulder pursued another alternative direction in design support. Their efforts were aimed at the *Domain-Oriented Design Environments* (DODE) in various design domains from user interfaces design to architectural layout planning (Fischer 1992; Nakakoji, Sumner *et al.* 1994). An underlying principle of most DODE-s was critique of the designers and design argumentation. A generic architecture consisted of a catalogue of components and design cases, argumentative system, construction and simulation tools that were integrated through links satisfying various functional aims. Among them was a catalogue explorer to browse and retrieve appropriate cases, an analyser serving as a critic checking the compliance of current design with the given guidelines and tactics, and finally an argumentation illustrator tool helping to understand critic's arguments.

DODE-s promote not only support for the design task, but also deeper understanding of design, underlying principles, related arguments and rationale. Domain knowledge is in the focus in this approach because, as Fischer (1992) claims, the lack of domain orientation renders the amount of support that might be provided and also decreases the shared understanding of problems among different parties involved.

2. State-of-the-art in KBDSS research

Several major strands of current research activities exist in the development of knowledge-based systems for design. On one end of the scale are efforts to develop an *intelligent CAD* (MacCallum 1990) as a heuristic extension of traditional CAD. A large group of activities consists of *case-based design* systems that employ various CBR-related techniques to adapt previous solutions to suit current needs (Watson and Perera 1997). Another group of researchers pursues the usage of *prototypes* in design (Gero 1990), where a prototype is a model of design product or component to be adapted and instantiated for a specific case. When we move further toward the more abstract end of the scale we arrive to the re-usable and sharable *model-based design* systems (van Heijst, Schreiber *et al.* 1997), where design task and domain components are modelled using hierarchical structures. Different models may be valid inside a single domain or across several domains, for a single task or a group of several tasks (Motta and Zdrahal 1998).

There have been several attempts to combine some of the above-mentioned principles and develop *hybrid* support systems. For instance: case-based and model-based support in IDEAL (Bhatta, Goel *et al.* 1994), rule-based and case-based approach in DODE-s (Fischer 1992), case-based and constraint satisfaction techniques of EDS and IFM (Smithers, Conkie *et al.* 1990; Tang 1997), etc. However, most of the current approaches address only some issues of design. They deal very well with case adaptation, while some of them pay less attention to other important issues of an intelligent support, such as learning, 'across-domain' knowledge transfer, and so on. Research as proposed further in this document can be also categorised as a hybrid design support system. More details on these issues shall be addressed in further sections.

As it was stated earlier, there are many research activities devoted to design. However, most of the current CAD or design support systems address only a few issues occurring within the design. There is a lack of support given to designers in early phases of the design, which are often crucial for successful designs. Prescriptive design support significantly renders designers' creativity; on the other hand pure case-based approaches are better in dealing with creativity and innovation. However, CBR-related techniques are not a general remedy, especially because it typically works in narrow domains and requires quite thorough description of the current problem to be able to retrieve a similar case. Also there is clearly a lack of understanding of the importance of design process history and its role for analogous reasoning. All mentioned issues are partially responsible for the gap in the design-related research. The research as detailed further should 'build a significant portion of the bridge' to overcome the identified gap. It is understood that to achieve a complex and generally accepted solution to these issues requires a lot of research work to be performed in many different sciences. Therefore, the aim of the current research is to show possible ways how the gap and related issues might be overcome using hybrid technology of knowledge modelling and analogous reasoning.

B. Introduction to knowledge models

Newell (1982) introduced the ‘knowledge level’ assuming that an agent’s actions during problem solving are determined by its knowledge about the problem. Thus it is possible to describe (or model) agent’s actions on a level of knowledge it has about the problem at hand. One approach to the construction of knowledge models[3] (KM) is based on the entities known as common ontologies. Gruber (1993) understands ontology as “*an explicit specification of conceptualisation; in AI systems, what ‘exists’ is what can be ‘represented’*.” Van Heijst, Schreiber *et al.* (1997) add it is a theory of “*what can exist in the mind of knowledgeable agent*;” and finally Chandrasekaran, Josephson *et al.* (1999) see ontology as “*a representation vocabulary typically specialised to some domain or subject matter*.” Ontologies because of their nature may significantly clarify the structure of knowledge within a domain and provide means supporting knowledge communication, sharing, re-use, and transfer.

For a single subject several ontologies may be developed. They will differ mainly in their specificity and viewing perspective. In the literature we can find examples of ‘upper’ (generic), ‘middle’ (prototypic), and ‘lower’ (problem specific) ontologies (Gero 1990; Chandrasekaran, Josephson *et al.* 1999). Upper ontologies are typically more abstract and are often suitable for more domains. On top of them are built middle and lower ontologies. Upper ontologies are often task oriented, whereas lower ones are more problem oriented (Motta and Zdrahal 1998). It is obvious that both types may have their specific roles in the design process. These roles will be reviewed and discussed with regard to the demands on KB as they were set in Section II.C earlier. The first demand was to have a KB as a source of domain and design knowledge. This demand is directly satisfied by the definitions of KM and ontologies as representation vocabularies of subject specific terms that clarify and structure knowledge within KB[4]. KM is a hierarchical structure itself, so it is possible to find an appropriate level of description depending on the design phase and the designer’s experience.

The second demand was to use a KB as a reference for the purposes of easier navigation and retrieval. This can be satisfied through the hierarchical relationships between upper and lower ontologies and their elements. Similarity between previous design cases and the current problem might be assessed in the network of hierarchically arranged knowledge models containing generic terms and terms specific for the different domains. As an example we may have *Instrument, Agent* and *Object* as generic, task oriented terms of an upper ontology. Then, *Missile*, *Launcher* and *Enemy-Plane* will be instantiations for the air defence KM; whereas *γRay*, *γGenerator* and *Tumour* will be instantiations of the same generic terms for the cancer treatment KM. Air defence and cancer treatment are ‘parallel’ KM that are related through a generic KM. Thus it is possible to perceive the concepts of *Missile* and *γRay* as equivalent in a certain sense because they address similar actions and instantiate the same generic terms (e.g. *Instrument*).

C. ‘Ideal’ knowledge-based design support system

Suggested approach to the knowledge-based design support is built mainly on the basic assumption in the ‘design process’ definition, namely that the design is a *goal-oriented* activity. The design problem is being described with particular goals and expectations in mind. Typically, design goals and designers’ expectations are expressed in form of what functional requirements shall final system attain. On the other hand, designers’ description of the designed system reflects what structures might be applied to achieve desired functionality. Thus, different ‘languages’ are available for designers to address different needs and phases of the design process (Fischer 1992). It is quite difficult to separate them into independent design phases because, as we mentioned already, both languages are used almost simultaneously to express designers’ current understanding of the design problem requirements and solutions.

The structure of ‘design languages’ plays an important role in the design process. As observed by Smithers, Conkie *et al.* (1990), carrying out a design task does not result only in a single solution, it also results in knowledge extension and better understanding of a certain type of design tasks. Thus knowledge acquired in

[3] Although the full, correct term is ‘knowledge-level models’, it is usually possible to abbreviate it to ‘knowledge models’; but we must keep in mind that we model an object or a problem on a knowledge level, not knowledge itself!

[4] KB is understood as a functional unit of KBDSS, whereas KM is a means how KB can be constructed. Often one term might be used instead the other; the meaning is given by context.

tackling a particular design problem is transferred back to the design and domain knowledge bases for future re-use. In this sense it is obvious that design process is understood as an *exploration task* rather than *search task* (Smithers, Conkie *et al.* 1990). Search is more or less unidirectional application of available knowledge to find a desired solution in a limited design space. Heuristic and domain specific knowledge might be used to improve search performance and further limit the design space but the existing knowledge is rarely extended by a single design task. On the contrary, when we assume design process as an exploration, from the very meaning of this term it is visible that new knowledge acquisition is 'expected' to extend the current sources.

As it was stated earlier, analogies in design and knowledge extension are basically two crucial features of a creative, professional design. Therefore, any prescriptive model of design that is rigid and does not support knowledge transfer cannot be considered as a feasible *knowledge-based* design support system. Tang (1997) reviews various approaches to knowledge-based design systems and claims that most tools available cannot satisfactorily scale within and across domains, adapt to new contexts, or acquire new knowledge. Partial remedy to these drawbacks lies in the case-based design (CBD) paradigm. Watson and Perera (1997) show that CBD systems may facilitate not only analogous thinking and experience-based knowledge re-use but also are able to learn by introducing new cases into knowledge repositories.

From what was mentioned so far we may summarise that a viable KBDSS should be able to use all different categories of knowledge – theoretical and empirical knowledge, problem domain and design task oriented knowledge, as well as historical knowledge. The criteria satisfied by a successful KBDSS (see also similar breakdown in Tang 1997) include among other issues the provision of:

- Support for the construction and extension of design knowledge bases;
- Support for the solution derivation reflecting human actions in design processes;
- Support for the explicit explanations and justifications of design decision steps;
- Support for the multiple-context design, to the exploration of a problem from different perspectives;
- Support for the knowledge transfer from individual design tasks to a shared knowledge base.

IV. HOW IT IS INTENDED TO WORK

This section summarises and further discusses the implications and demands set on the knowledge-based design support systems in Section II.C. As it was mentioned earlier, essential features of the design process are the designers' reflection on their actions and the existence of strong links between the co-evolving requirements and solutions. Therefore, the iterative nature of design processes is presented below as a problem of re-formulation and re-framing. Using these two operations, the designers are able to 'generate' and refine both, the current requirements on the designed product and the solutions satisfying the requirements. After formal definition of these operations, the discussion attempts to answer also the other questions raised in Section II.C regarding knowledge needed in the support for design. To refresh the memory, the questions are:

- What knowledge is useful for a design support system and how to represent different knowledge?
- How to present different types of knowledge to the designers?
- How to support the interactions between the computational support system and the designer?

The 'what knowledge' question is focused on the knowledge representation in KBDSS; the 'interactions between designers and computational system' are briefly discussed in the consequent part; and finally the issues with the 'retrieval and presentation' are discussed at the end of this section. Although these three facets do not provide an exhaustive description of issues with KBDSS, they address satisfactorily the main topic of this document – the knowledge-based support for the design.

A. Re-formulation and re-framing in design

Section on the nature of design concluded with several implications and formally described design process as an iterative application of a mapping ρ. The mapping transforms the current understanding of a problem PB^k to a new understanding PB^{k+1}. According to the theory proposed by Choueiry, McIlraith *et al.* (1998), such mapping fulfils the role of a re-formulation operator that in general has a form:

$$\rho = \langle \mathit{Condition};\ \mathit{Transformation} \rangle \qquad (3)$$

where *Condition* denotes when particular operator may be applied and *Transformation* denotes procedure that can change the original problem PB^k into a re-formulated one PB^{k+1}. In their theory Choueiry *et al.* use only operators for re-formulation to reason about physical systems; however, the same theory applied to design requires some additional operators to generate new formulations. This inconsistency might be removed by the introduction of another mapping type. Let us use ρ_Π as a symbol denoting *re-formulation* as defined in (Choueiry, McIlraith *et al.* 1998) that is triggered by the availability of some information with current problem description as well as appropriate problem solving technique. Let us introduce a design specific operator called *re-framing*, denoted as ρ_Φ, and define it as a mapping that may change current problem solving technique, current requirements, and expectations, thus shifting the design process into a new context. Equation (2) might be updated for these two operators as follows:

$$S^{k+1} \text{ »} DP^{k+1} = \rho_\Pi (DP^k) = \rho_\Pi (\text{á}F; A, E\text{ñ}^k) \text{ »} \rho_\Pi (R^k \text{ ® } S^k) \qquad (4)$$

for problem re-formulation and

$$E, R^{k+1} \text{ »} DP^{k+1} = \rho_\Phi (DP^k) = \rho_\Phi (\text{á}F; A, E\text{ñ}^k) \text{ »} \rho_\Phi (R^k \neg S^k, E) \qquad (5)$$

for problem re-framing.

In other words, problem re-formulation deals mainly with the modifications of solutions based on the current set of requirements and corresponds, more or less, to the process described by Altshuller (1984) that is directed from the (contradictory) requirements towards the solutions. Problem re-framing affects mainly designer's expectations and consequently requirements that determine a contextual frame for the current understanding of a design problem; these are determined through reflection on the current solutions as observed by Schön (1983). Design task history is affected by any operation upon design problem description as implied by its definition. Schematic model of design process as described here is depicted in Fig. 2 below; including both mapping operators – re-formulation (ρ_Π) and re-framing (ρ_Φ), as well as knowledge transfer from design task and its current 'formulation' (F) to the design/domain knowledge bases (KB).

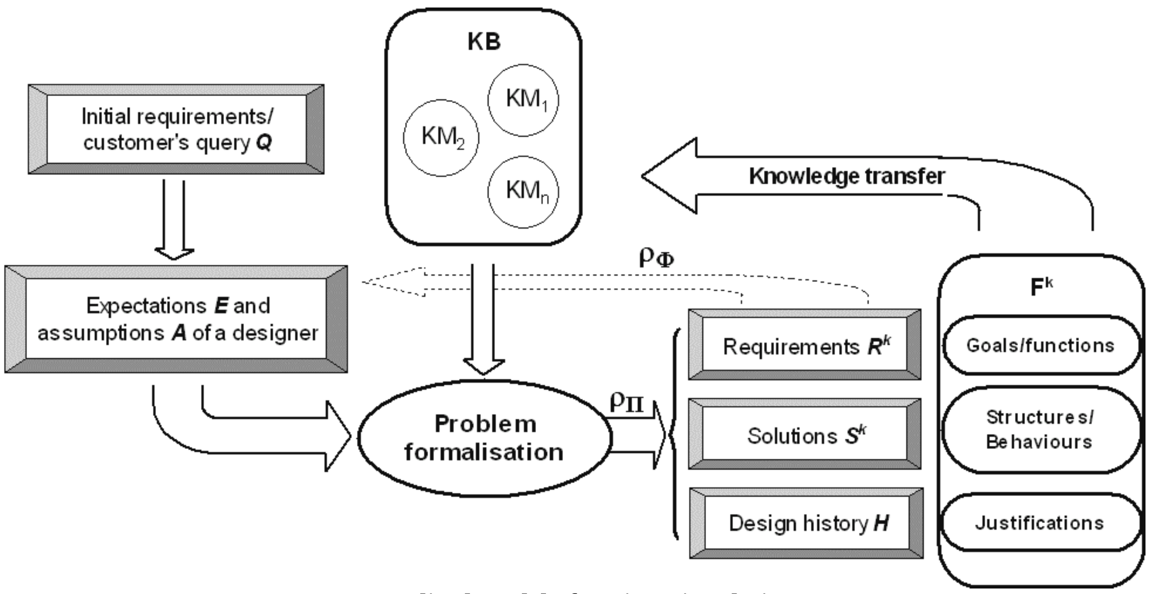


Fig. 2. **Generalised model of engineering design process**

B. Knowledge representation

Design is a knowledge-rich activity that might well benefit from the systematically represented knowledge. In this section the issues with the structure of knowledge bases and the different types of knowledge useful for the design support will be briefly presented. To help designers exploring the complex knowledge bases (KB) in design their knowledge might be classified around several 'dimensions': (1) problem domain knowledge; (2) design task knowledge; (3) indexing knowledge; and (4) design history knowledge.

(1) *Problem domain knowledge* consists of common concepts used in a particular domain including theoretical foundations of a domain (at least that part that is formally expressible) and general relations among concepts. This knowledge has a much more static character, might be at least partially prepared in advance, and eventually might be available as external knowledge sources or plug-in modules that are specialised for a narrow domain. For example, cancer treatment domain knowledge might include terms as *tumour, malign, benign, healthy tissue* etc., then different parameters relevant for a domain – *temperature, radiation resistance*

of tissues, thermal and electric conductivity of tissues etc. Also various formulae to calculate some of these parameters might be available.

(2) *Design task knowledge* reflects good practices how to perform a design task in general or what are specialities of a particular domain. This knowledge model may describe ways how designers usually approach their problems; knowledge includes design guidelines, successful designs from the past, various heuristics etc. Design task knowledge is used to manipulate domain specific knowledge in order to generate new data for the history of the current design task.

(3) Purpose of *knowledge models for indexing* is to maintain an unambiguous structure of knowledge bases, to relate similar design tasks through general reference ontologies. For instance, design tasks may be indexed according to functions they eventually attain, behaviour they show, possibly goals they were expected to satisfy, or roles different components may play in the design task. To illustrate the use of this knowledge we may take a *car* design task and a *plane* design task as both satisfying the goal of transporting an object or both attaining a functionality of moving, changing positions. Also, the following components may be perceived as similar based on their roles: *car electric engine* and *plane jet engine* are both *Instruments* causing the same *Effect* (move); *electricity* and *aircraft spirit* are both *Agents* needed to use the instruments; car *drive* and plane *flight* are similar *Actions* of movement and so on.

(4) And last but not least important dimension of knowledge in KBDSS is *design history knowledge*. From the points made earlier it is possible to conceive history knowledge as ‘a folder’ putting all other types of knowledge together into what can be marked as ‘a design process’. Design task history is case specific for an individual task and may be understood as a flow diagram describing that particular design process. It will contain individual decision steps, their descriptions and justifications, performed reasoning of both human designers and KBDSS, role assignments to particular concepts from domain specific KB, location and re-use of similar design tasks (called design stories) etc.

C. Design process ‘control’

Issue of controlling the design process has much to do with the interaction between designers and KBDSS. Basically, two distinct approaches are present in current research activities – automated design (‘get humans out of the loop’) and intelligent design support (‘get a computer into the loop’) (Fischer 1992). Continuously the latter alternative is becoming more popular among AI researchers. It has a significant implication towards KBDSS that we should not force human designers to comply with a strategy selected by and for a computer, rather the activity and control should remain with human who decides when to include a computer ‘into the loop’. People do not like ‘automated assistants’ approving every action of a designer (Smithers, Conkie *et al.* 1990; Fischer 1992). Much more feasible is an approach where KBDSS suggests general strategies to satisfy designers’ assumptions and expectations, assesses various aspects of available design alternatives, maintains multiple design contexts, and justifies (prompts for justification) of major decision steps.

This document does not focus on various different modes of interaction between the designers and computers; the major control strategy in a proposed KBDSS should be opportunistic in the sense that a system follows designer’s decisions. Much more important is the discussion on the retrieval and presentation of the relevant knowledge. The idea is that the KBDSS will present the relevant knowledge (especially previous design stories) based on the current assumptions, requirements and/or described components, features, and roles.

D. Location and presentation of relevant knowledge

Finding the relevant case, design story or simply knowledge is a crucial activity in design support. As mentioned earlier, designers rely heavily on their previous experience and are able to use it creatively. To access the repository of design stories this must be indexed suitably and efficiently (Watson and Perera 1997). One indexing approach is based on associative recall using a relationship-based indexing scheme (Watson and Perera 1997). KM and underlying ontologies might provide such a scheme. We have mentioned that one of the roles ontologies may play in KM and KB construction is that of a reference vocabulary. Assuming there is an ontology containing basic indexing concepts and relations among them, we have a scheme suitable for the associative recall. From the four types of knowledge sources the most suitable one for the purposes of indexing and associative recalling is the third one – indexing knowledge base containing concepts as *Goal, Function, Behaviour, Instrument, Agent, Object, Action* and so on. Different domain, design task, and history specific concepts may be linked to the concepts of an indexing KM. Such arrangement will give designers an opportunity to see similarities between the design problems that seem far apart at the first glance.

After locating a similar story in the repository, this must be presented to the designer in a user-friendly way. An ontology-based KM provides an advantage of a reference vocabulary that might be used to mediate and structure retrieved results. KBDSS may suggest possible mappings between current and retrieved design problems; however, it is on the designer to pursue the indicated direction, agree or discard the suggested alternative, refine, change or generate the mappings as needed.

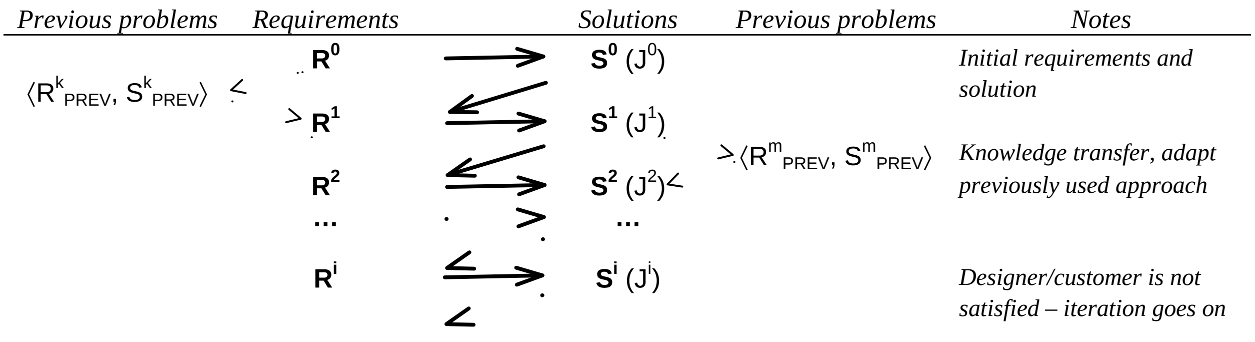
In a design story presentation, emphasis must be given on its easy comprehension and understanding by the designers. Stories should be presented in such way as to support innovative and creative design (for details see also Section II). It means that various 'rich' formats are more suitable for the story presentation including drawings, sketches, or models. In the current approach to KBDSS, we assume that system will use justifications and rationale as available in histories of particular cases to present a story as well as some reasoning using problem domain knowledge. Nevertheless, it is possible to foresee a philosophy where KBDSS will interact with other design support tools (such as CAD, databases of standards and available components etc.) to simulate, assess, and present current alternative in entirely novel way. Future KBDSS may be well able to support entire lifecycle of a product from requirement specification, through qualitative modelling, design, simulation, prototyping and ending with deployment, maintenance and further improvements.

V. THEORETICAL FOUNDATIONS OF DESIGN SUPPORT

In this section, I will propose a theoretical model of design process that incorporates knowledge from the previous sections and further extends the issues raised in Section II.C and discussed in Section IV. Included are the concepts of co-evolving requirements and solutions in design, reflection on action, and analogous design. First, the basic terms will be defined, and then the overall design process will be described graphically as well as explained in words. For illustration and reference see also diagrams in Fig. 2 and Fig. 3.

According to Section II.C two types of a design problem descriptions are distinguished: *requirements* (R) and *solutions* (S). Requirement R is understood as a condition that must be satisfied by the final artefact. It may be either a condition requiring that a certain property is fulfilled or a condition that must not be violated by the solution (S). Solution S is understood as a set of variables, parameters, values etc. that satisfies the current set of design requirements. Any statements generated by either human designers or a computational support system may qualify as requirements or solutions. In the further discussion, they will be referred to as *assumptions* (A). Depending on the nature of an assumption, it is classified to one of the above-mentioned types of design problem description. Some requirements, functional or structural relationships will have a broader impact and possibly will be problem- and/or domain-independent. These might be included in domain-oriented knowledge bases. Domain-oriented knowledge base (KB) or simply *domain knowledge* (DK) represents persistently stored data that describe the knowledge valid in a broader scope because of:

a) general applicability (e.g. physical, chemical, thermodynamic laws),
b) axiomatic nature (e.g. definitions of physical quantities, diseases and their symptoms), or
c) empirical/common sense nature proved by experiments or everyday experience (e.g. observable symptoms, diseases, and medical remedies to cure them).



Previous problems
Requirements
Solutions
Previous problems
Notes
R^0
S^0 (J^0)
Initial requirements and solution
$\langle R^k_{PREV}, S^k_{PREV} \rangle$
R^1
S^1 (J^1)
$\langle R^m_{PREV}, S^m_{PREV} \rangle$
Knowledge transfer, adapt previously used approach
R^2
S^2 (J^2)
…
…
R^i
S^i (J^i)
Designer/customer is not satisfied – iteration goes on

Fig. 3. **Spiral of requirements–solutions co-evolution**

As mentioned in Section IV in addition to the domain knowledge designers often use the knowledge from previously solved design problems that is not formally included in the domain knowledge base. However, it represents a valuable source of information that may significantly influence the entire design process. Eventually, this experiential knowledge may be 'promoted' and included in a domain knowledge base when it satisfies condition c) above. In the scheme depicted in Fig. 3 this knowledge source is represented by the previous cases. They are described using the same structure as the current problems, i.e. requirements, solutions, and relationships among them. Formally they are distinguished by an additional keyword '$_{PREV}$' in a form R_{PREV}, S_{PREV}. See Fig. 3 for the details about the possible relationships between defined concepts.

Let us assume that the designer is given some initial information from the customer asking him/her to design some artefact. Customer's information is translated by the designer to the language that is specific for his/her area of expertise. The result of the translation is a pair of initial requirements and solutions; in Fig. 3 it is depicted as $\langle R^0 \rightarrow S^0 \rangle$. R^0 represents an initial perspective to understand a problem and S^0 an initial solution proposal that may satisfy given requirements. However, S^0 is far from the solution that would satisfy the customer's and designer's expectations. To refine and better understand the problem, designer draws on his/her previous experience, i.e. the support system retrieves a similar design situation from the design repository and suggests possible mappings between the current and previous problems. Retrieved design story (situation) is also represented by pairs of requirements and solutions – $\langle R^k_{PREV} \rightarrow S^k_{PREV} \rangle$.

Designer may adapt or transfer the previously used approach to (i) modify the set of requirements on the current problem and/or (ii) generate a new solution that would satisfy the existing requirements. Anyway, the result of the adaptation is depicted as a pair $\langle R^1 \rightarrow S^1 \rangle$. Although in the figure indexes of both the requirements and solutions are incremented, in the real design situation designer may decide to modify only one of the sets, e.g. requirements, without attending to solutions. Even if a solution exists in the space of possible solutions it may be impractical to explicate it every time when requirements were changed. Vice-versa, designer may decide to modify an existing solution, to derive new alternative without changing the requirements. Formally these situations may be represented by equalities $R^{n+1} = R^n$ or $S^{n+1} = S^n$

Having a set of requirements and corresponding solutions in the iteration step 'i', makes it possible to retrieve other similar design situations, reason using domain models, or apply one of the generic rules to shift one step further – to the pair $\langle R^{i+1} \rightarrow S^{i+1} \rangle$. The whole design procedure ends when the designer is satisfied with both requirements and current solution. At this point s/he may want to simulate the current solution or present it to the customer. Both actions may be understood as an 'evaluation phase' and may lead to the further reflection and identification of new requirements. After the evaluation stage, the existing requirements may be modified, their priority and importance may be changed, or completely new demands may be presented the parties involved were unaware of before. And the whole procedure may begin again – previous situations may be retrieved, adapted, new requirement modification or solution alternatives introduced etc.

A sequence of design decisions – requirements-solutions pairs, constructed during a particular design process, represents the historical knowledge. As stated earlier, design history knowledge is very important for understanding the design process in the context of similar design situations, designer's decisions and justifications. This knowledge is built automatically in a form of logging designer's actions, decisions, explanations, and justifications. In equations (1) to (5) design history knowledge was described (using a letter H) as an independent 'parameter' that is evolving together with requirements and solutions[5].

Designer's interventions during the design process can be done in a form of introducing new assumptions. Depending on the goal of the intervention assumption A may be understood and stored as a new/modified requirement (R^{i+1}), as a new/modified solution (S^{i+1}), or as a *justification* of performed decision step (J^i). However, in all earlier mentioned cases assumptions are stored as case-specific knowledge sources valid for a particular design situation and/or design step. In addition, it is possible for the designer to introduce new datum that will have a broader impact. Such 'global' information is stored persistently in a design or domain knowledge base (KB and consequently KM^k) depending on the designer's choice and nature of the datum.

[5] This is the reason why the equations (1) to (5) mentioned problem description as a triple $\langle R^i \rightarrow S^i; H^i \rangle$. R^i and S^i are represented explicitly and introduced by designers, whereas H^i is logged automatically and implicitly associated with a particular design step.

VI. FORMALISATION GUIDELINES – 'ALGORITHM'

In the previous sections, general requirements on the KBDSS were raised and theoretical foundations were briefly discussed. Table 1 in Section II.C compared and associated the steps in the design process as different researchers observed them. To summarise – the design process typically contains the following phases:

(1) *recognition* of a problem;
(2) identification of some key features, problem *framing and formulation*;
(3) analogous thinking, adaptation and *knowledge transfer*;
(4) *reflection* and assessment; and finally
(5) *re-formulation and re-framing*.

The aim of this section is to introduce the *guidelines* that may serve as a framework for supporting the designer's efforts during the design problem formalisation. The guidelines are presented in a form of recommendations that may but do not have to lead towards a guaranteed solution of the design process. It is a 'heuristic' rather than an 'algorithm'. The term 'algorithm' is defined as a program that is deterministic, robust, etc. whereas the proposed framework does not satisfy all those requirements. The philosophy of design problem formalisation and support for it will be presented as a sequence of numbered steps denoting particular actions of a designer including the computational support provided and explanations of individual steps. The symbols introduced in the previous section is used also here to provide better navigation in the text.

Step 1. Recognition of a problem

1.1 What do we know about problem at hand? ® *Q, R^0, S^0*

a) What is the customer's query (Q) to be answered by the solution?
b) What requirements (R^0) are given for a problem/product in advance?
c) Do we know a scenario in scope of which the problem occurs (S^0)?
d) Are there given any evaluators to assess designer's satisfaction with a particular solution?

1.2 What is expected from the solution to the problem? ® *E*

a) In the current scenario or solution (S^0), are there any flaws in performance to be removed?
b) What is the expected performance or result (E)?

1.3 Formulate the problem within the design situation at hand

a) First, try to formulate the difference between the existing and expected performances.
b) Is it possible to perceive the problem as 'remove the identified difference'?

The first set of guidelines is derived from the first step of the design process (see also Table 1). The main output of the first phase of design process should be the provision of information on the initial problem description and query related to the task to be performed; denoted by letters R^0, S^0 and Q, respectively. All pieces of information will be used in further steps to identify other important features of the design problem. Customer's query Q and both R^0, S^0 can be incomplete and contradictory in the very beginning, however, they should be able to show the direction in which the designer will move during the next steps of the design process.

Having an initial description of a problem and a query, designers may formulate their expectation, the expected solution or performance (E). Often E will be an answer to the query Q as proposed in the previous step. Often it may be possible to compare the current position in design with the expected one. Typically, if there is a difference between these positions, it is this particular difference (~ compare to Altshuller's contradiction, Section II.C) that causes a problem; and it is this difference that must be removed during the design process.

Initial design information is introduced by the designer in co-operation with the customer. In this early phase the designers might be supported by the system presenting them various 'design stories' from the repository an showing how a particular initial description was derived from a customer's query in those 'stories'. The aim is to give the designers 'a feeling' that any information contained in the query might be suitable to begin with and that the initial description does not have to be necessarily sound and complete.

Step 2. Formalisation of a problem – problem model construction

2.1 Map the current problem on the design ontology.

a) Summarise the problem description using a problem-specific terminology.

b) What are the relationships between specific terms and general concepts existing in the domain and design ontology?
c) Specify the current problem using the existing concepts, or use available links to include the problem description in the appropriate position in the hierarchy.

2.2 Create an abstract model of a problem, a frame ® *A* Þ *DK*

a) Specify and instantiate the roles the problem components may take (e.g. Object, Instrument, Agent, …); they may help to find relevant parts of the domain knowledge (*DK*).
b) Formulate the expected result in general terms of Object/Instrument ontology; specification of roles leads to the formulation of assumptions that will guide the current approach to the problem solution and design process.

2.3 Find and retrieve similar design situations ® R^{k}_{PREV}, S^{k}_{PREV} (J^{k}_{PREV})

a) Find and show a design story about a problem that was described by the designer with the same or close expected result in mind (map the descriptions on the level of Actions between Object and Instrument)
b) Show mappings between problem-specific and generic terminology, i.e. what were the relationships between Object/Instrument ontology and the description of a problem in case-specific terms.

The recommendations in the second set are based on the following steps in the design process – *identification* of key features of the problem, problem *framing and formulation* (see also Table 1). The aim of the second step is to describe the significant features of the current problem in order to recognise what could be done, and relate the problem to the previously solved design situations. As mentioned earlier, ontologies provide excellent means to organise subject-specific concepts that are applicable for the problem representation. First, designers should note down a few significant features or parts of an initial problem description. Then, the identified features should be related to the existing concepts of generic ontologies. Such operation creates a seed for a problem domain definition; in other words, we may see it as designer's assumptions that draw boundaries to constrain otherwise vast problem space.

A combination of generic (Object/Instrument) ontologies and domain ontologies may bring new components, features, or parameters in designer's focus. Vertical and horizontal relationships in the ontology hierarchy enable the designers to explore the design repository containing theoretical foundations of that particular domain, as well as experiential knowledge of previous problems. Previous problem $\langle R^{k}_{PREV}, S^{k}_{PREV} (J^{k}_{PREV})\rangle$ – also called 'design story', may be retrieved based on the similarity of roles that were assigned to different elements of the previous problem description. Designers are thus allowed to 'think about' the current problem in the familiar terms of a previous design situation (story) and re-use the relevant descriptions.

The support for this particular set of actions will be as follows. Support system may show how previous design cases were mapped onto common ontologies. If designer does not provide any hints on the preferred problem domain, the system may show 'design stories' randomly. If designer identifies some features of the current problem and gives some hint about the problem domain, the system will be able to retrieve more relevant and more similar stories from the repository. Presentation of an existing story gives the designer a mechanism of familiar and explained terms that might be re-usable for the current problem description. The role of the support system is to retrieve similar stories and suggest how they can be mapped on the current problem description. The decision whether to accept the mapping or not is upon the designer. If a mapping is accepted, the designer introduces a new assumption (requirement or solution) into the current problem history.

Step 3. Knowledge transfer and re-use based on retrieved cases

3.1 Justify mappings from step 2.3b ® R^{i}, S^{i} (J^{i})

a) Show comments associated with the retrieved case and mapping
b) Show a reasoning process that may explain certain features in a retrieved case
c) Suggest a possible use of the identified mapping as a frame for the current problem.

3.2 Derivation of consequences

a) Assuming that the coarse-grained mapping from step 2.3 is correct, show other Object(s), Instrument(s), Action(s), Agent(s), etc. that are present in the retrieved case together with their relations
b) Show the effects the additional components had on the previous description and solution
c) Imagine we transpose our current query Q on the retrieved example, would it be addressed and answered?

3.3 Problem model update ® $R^{i}, S^{i} (J^{i})$ & $R^{k+1}, S^{k+1} (J^{k+1})$

a) Try to introduce corresponding actions to the current problem (use identified *DK*).
b) Re-formulate the effects and features from a similar story in the current problem terminology.
c) Find out corresponding Agent(s), Instrument(s) etc. in the current problem domain.

The third step in a typical design process as identified earlier was *knowledge transfer*, analogous thinking, knowledge adaptation, and re-use. Designers' 'thinking about the current problem in term of familiar situations' was identified as a major professional strategy in tackling uncertain design problems (Dzbor 1999b). Section II.C also included the implication on the support for analogous 'thinking' in design. The current set of guidelines suggests how the analogy that was found, retrieved, and consequently mapped onto the current problem, can be used to update the current problem description.

Once a familiar design story had been retrieved and mapped on the current problem (see step 2.3), designers should justify its relevance with the current problem. The justification can be done in form of existing comments and rationale as well as in form of reasoning in a hierarchy of ontologies. The operation of justification proves the validity of the coarse-grained mappings and leads towards the derivation of further consequences. In general in the justification procedure both, the designer and the support system may take part. However, their contributions will be different, almost complementary. The support system may reason in a hierarchy of ontologies, whereas the designer will contribute with textual comments and rationale.

Consequences derived from the reasoning process can be again justified by an experience expressed as a rationale in the design story or a relationship between the concepts in ontology. During the process both, the analogy mappings and the current problem description are continuously refined. Designer introduces new effects, features, or elements to the current problem model (as R^{i+1} or S^{i+1}) using derived terms reasoned by the support system from the domain or design knowledge base.

Step 4. Analogy application – solution model construction

4.1 Apply the identified Action(s) using Instrument(s) and Agent(s) on Object(s)

a) Are effects from step 3.3b when transposed to the current problem domain harmful, required, or might be ignored?
b) If harmful, how was the harm tackled in the retrieved story?
c) Go back to 3.3 and include additional Agent(s), Instrument(s), and Action(s) so that the harmful effect might be removed similarly as in the retrieved case.

4.2 Re-formulation of the current description

a) Formulate the solution from abstract terms to the current problem-specific terms.
b) Perform backward mapping from a generic, task-oriented ontology to a problem-oriented ontology.

4.3 Remove side effects

a) Add a local goal of removing a particular (harmful) side effect that was identified.
b) Treat it as a separate problem, and go back to step 2.2.

As we already stated above, the literal re-use of experience is rarely possible, and often may lead to an undesired fixation on familiar approaches. Therefore the elements from a retrieved design story must be transformed or adapted to suit the needs of the current problem. Two operators were introduced that perform the transformation – re-formulation and re-framing. A problem might re-formulated when we take the current assumptions, problem domains and mappings as 'granted and unchangeable' and attempt to derive further consequences from them. What we do in this case is re-using a previous design story in a context of new assumptions. Mostly the re-formulation will deal with issues how to describe a particular effect in a particular domain, how to tackle this effect in a new domain, how to change the requirements and descriptions to implement the chosen strategy, etc.

Step 4 is in the algorithm as a detailed continuation of step 3.3 and its aim is to provide a ground for reflection on the current understanding of a problem. Reflection often leads to a better perception of a design situation, discovery of new objects or features in the situation, and eventually to the modification of the problem description. In step 4 iteration to one of the previous steps may occur, so that new goals are set up and these are treated as separate sub-problems. The description of the sub-problems may extend our knowledge about the design problem itself, some requirements may be altered, and some new elements described.

Step 5. Solution and problem model evaluation

5.1 Evaluate the model of a solution to the current problem using existing criteria.

5.2 Does the model solution (S^i) answer the query Q from the step 1 and all current requirements (R^i) satisfactorily? If yes, go to step 5.3.

5.3 Is the designer/customer satisfied with the current set of requirements (R^i) and no new requirements are found? If so, the problem can be considered as solved…

5.4 Solution (S^i) still does not satisfy all desired requirements (R^i), no further improvements are possible within the current perspective (A, E, DK). Try to amend existing expectations E or eventually customer's query Q (from step 1).

When no further amendments can be performed in the loop containing steps 3 and 4, it may be necessary to change the foundations of the current approach. Compared to what was mentioned earlier, the loop between steps 3 and 4 corresponds to the internal feedback as depicted in Fig. 1. The outer loop in the same figure corresponds to the more significant changes in the designer's perspective and occurs between steps 1 and 5 in the design-guiding framework. This operation was defined as a problem re-framing. During re-framing, the designers may change their assumptions that have led to the current problem understanding or they may alter their expectations regarding the product of the design.

VII. DESIGN SUPPORT SCENARIO – 'EXAMPLE'

In this section, an existing design scenario well described in the literature will be used as a pilot study for the verification of the design guidelines presented earlier. The initial expectation is that we would be able to describe the design process and justify the designer's steps similarly as in the solution described in the literature. We decided to use such example because of several reasons, just to mention a few of them:

a) traditional approaches are well described, together with their failures, pluses, and minuses;
b) novel approach using an existing design scenario can be directly compared to the previous approaches;
c) stronger implications can be drawn from the study, because the same scenario was several times repeated

The design of Electric Power System in Satellite was chosen as a design situation that will be further analysed in a form of pilot study. The same scenario is described in several research reports from authors working at the Knowledge Systems Lab, Stanford University; see for example (Iwasaki, Fikes *et al.* 1993; Vescovi, Iwasaki *et al.* 1993). The problem is given as follows: "*The device to be designed is the electrical power system (EPS) aboard an Earth-orbiting satellite. The main purpose of EPS is to supply constant source of electricity to the satellite's other sub-systems (such as communication, navigation, etc.).*" Also, let us assume that a designer has in the design repository situations, in which different power supply systems were designed in the past. In addition, we may assume that design knowledge model is not empty but already contains some common concepts and relationships regarding the domain of power supply devices, physics, etc.

In accordance with step 1 of the design guidelines (see Section V), the designer attempts to learn more about the design situation to be tackled, to recognise the problem, and eventually identify initial requirements and possibly solutions. What is specified above implies the following goals, requirements, and solutions[6]:

■ the aim is to design a power supply system of satellite orbiting Earth

Assumption A1 (necessary requirement): EPS has function 'to supply' electric energy.

[6] In further transcript symbol ' ' denotes designer's actions, ' ' denotes the reasoning and hints from KBDSS, '■' is a general remark.

Assumption A2 (necessary requirement): load has function 'to consume' electric energy.
Assumption A3 (requirement): power supply is constant, i.e. its life span is infinite ($T_\Omega \rightarrow \infty$)
Assumption A4 ('solution'): power supply is a technical system supplying electricity to load

- There is no existing scenario that could be improved, so the problem may be formulated as how to achieve the desired functionality, what do we need, what can supply power …
- Designer's expectation is to design an electric system that will satisfy all current requirements.

According to step 2 from the guidelines, designer tries to map the current situation on the existing ontology and domain theory. For instance s/he assumes that different components will have the following roles:
Assumption A5 ('solution'): consider EPS and load as some Objects;
Assumption A6 ('solution'): among these Objects exist two Actions (Supply-power, Consume-power);
Assumption A7 ('solution'): Electricity acts as a mediator or Instrument to attain desired functionality.

In this moment, design support system may retrieve similar design situations from the repository. Since very little is known so far, retrieval can be based for instance, on one of the desired Actions (supply power) and the Instrument (electricity). The result of retrieval process is for instance, a car battery that was used in some previous problem to supply electricity to car sub-systems. The following mappings are proposed by KBDSS:

- Mapping 1: car battery has function 'to supply' electric energy to electric sub-systems of a car; associate it with EPS, i.e. EPS $\cong$ battery;
- Mapping 2: car sub-systems have function 'to consume' electric energy; associate them with satellite's load, i.e. satellite load $\cong$ car sub-systems load
  Assumption A8 ('solution'): consider battery as EPS;

Since the designer agreed with the suggested mappings, KBDSS attempts to derive other interesting facts from this association. Consequently, the following information appears:

- Reasoning result 1: battery is described by a quantitative physical property called *output* that is expressed in Watts (W, kW, MW, etc.) – this quantity may be relevant to the satellite's EPS;
- Reasoning result 2: load is also described by a quantitative physical property called *power consumption* that is expressed in Watts or Joules (J, kJ, …) – this quantity may be relevant to the satellite's load;

Designer may assess the suggestions of the KBDSS and commit him/herself to the following assumptions:
Refine assumption A8 ('solution'): consider battery as EPS, and include descriptive quantities like capacity, express battery capacity and power consumption of the satellite's sub-systems in particular units

- Reasoning result 3 & Consistency check 1: car battery has such behaviour that it is getting weaker, losing its capacity quite quickly and must be often re-charged. Therefore the requirement A3 might be violated.
- Hint 1 (justified by an existing heuristic rule): Try to revise the violated requirement, it may be possible to modify it without loosing too much in the solution quality.

After reading this reasoning, consistency check, and attached justification/explanation designer returns to the mentioned assumption and revises it. S/he finds that it is practically impossible to design a technical system with an infinite life span and tries to define a 'reasonable' value for this parameter. Therefore the requirements is refined as follows:
Refine assumption A3 ('requirement'): the life span of EPS should be at least 20 years ($T_\Omega > 20$ years).

But such amendment did not solve all the problems, another problem is that initial capacity of the battery must have certain value so that it lasts 20 years to discharge it; the system picks up another issue with such approach:
Assumption A9 ('requirement'): to achieve required T_Ω the initial value of battery capacity must be greater than 200 Farrads {?!?} ($C_0 > 200$ F);
Assumption A10 ('solution'): Let us try the initial capacity $C_0 = 250$ F, which implies $T_\Omega = 22$ years.

Here designer tries to assess current solution against current requirements and it seems that no violation occurs. However, when studying the retrieved examples and their descriptions, designer becomes aware of a fact that the larger initial capacity makes battery more robust and thus heavier. S/he consults this issue with the customer (see step 5.3 above) and they find that the weight of EPS is an important requirement that was forgotten initially. The result is a new requirement regarding EPS weight:

Assumption A11 (‘requirement’): Maximum weight of EPS in a satellite is 50 kg ($M_{EPS} \leq 50$ kg).

After some calculations designer realises that assumptions A9, A10, and A11 are contradictory, that the original solution does not satisfy this new requirement, and thus the current approach is not leading toward a solution. There is a need to retrieve something new from the design repository:

Discard assumptions A8 (‘solution’) and A10 (‘solution’).
Decrease priority and importance of assumption A9 (‘requirement’).
Discard mappings 1, 2, and associated reasoning process.

- Mapping 3: petrol generator has function ‘to supply’ electric energy to electric sub-systems of a house; associate it with EPS, i.e. EPS $\cong$ generator;
  Modify assumption A8 (‘solution’): consider petrol generator as EPS;

System tries to reason further details relevant for the current choice of a generator as a power supply in satellite similarly as it did with a battery. The reasoning results may include:

- Reasoning result 1: generator is described by a quantitative physical property called *output* that is expressed in Watts (W, kW, MW, etc.) – this quantity may be relevant to the satellite’s EPS;
- Reasoning result 2: load is also described by a quantitative physical property called *power consumption* that is expressed in Watts or Joules (J, kJ, …) – this quantity may be relevant to the satellite’s load;
- Reasoning result 3 & Completeness check 1: when a generator is used as a power source fuel and air must be present in the Environment so that combustion process can occur.

Designer must thus refines the concept of Environment that was not yet described. System cannot give any hints since there are no relevant data in repository but designer knows that the satellite is orbiting the Earth. S/he can assume that it will orbit in certain altitude greater than 30 km. In addition, s/he is acquainted with an empirical knowledge that in such altitude above the surface, the air is very thin, it may be considered as vacuum. Therefore a new pair of requirement and solution appears:

Assumption A12 (‘requirement’): Petrol generator needs air as its outer Environment and presence of petrol as an Agent participating in the reaction.
Assumption A13 (‘solution’): Satellite orbits the Earth in the altitude above 30 km.
Assumption A14 (‘solution’ + domain knowledge): In the altitude of 30 km above the Earth, there is no air – we consider it as vacuum.

Unfortunately a new contradiction appears almost simultaneously and the designer is forced to give up the fuel generator as a power supply for a satellite orbiting the Earth:

- Reasoning result 4 & Consistency check 1: when the satellite orbits the Earth in the altitude above 30 km above the surface its Environment must be vacuum because of domain knowledge A14, which is inconsistent with the requirement A12.
  Discard assumptions A8 (‘solution’) and A12 (‘requirement’).

A new alternative is retrieved from the repository – for instance, a solar plant and similar mappings are developed to consider solar panel as EPS. Using domain knowledge and associated comments with the problem of solar power plant, the system reasons that the solar panel may be dependent on the presence of the Sun:

- Reasoning result 4 & Completeness check 1: when the solar panel is used as EPS it requires a presence of the Sun to generate power. Refine your knowledge about the Environment of the satellite.

Designer checks his/her common sense knowledge and defines new assumption that the sunshine is present in satellite’s Environment, but only for about 70 minutes out of 100.

Assumption A8 (‘solution’): Consider solar panel as an EPS.
Assumption A15 (‘requirement’): Solar panel needs the presence of sunshine and the Sun in its outer Environment.
Assumption A16 (‘solution’): One satellite’s orbit around the Earth lasts 100 minutes, sunshine is present 70 minutes.

System makes consistency check and finds out that assumptions A16 and A3 are inconsistent. During some periods, EPS may fail to supply electric power. Designer decides to discard also this alternative:

- Reasoning result 4 & Consistency check 1: when the Sun is not present for 30 minutes, the assumption A15 is violated, solar panel cannot work properly and supply electricity. Thus, assumption A3 requiring a constant power supply will be also violated.
  Discard assumptions A8 ('solution') and associated mappings.

In similar way designer may go through all available alternatives in the repository and discard all of them. S/he is not satisfied with the current approach and feels that a more radical shift of perspective must be made. The system provides a hint from a base of heuristic rules and best practices:

- Hint 2 (justified by an existing heuristic rule): Try to combine several simple alternatives to use their strengths and eliminate their weaknesses. You may begin with two best performing alternatives.
- Hint 3 (justified by an existing heuristic rule and common sense): You may combine things in parallel, in series, or both ways. Parallel is good when one thing is complementing the other, serial to amplify the performance of one thing.

In this case, the best performing alternatives so far were the battery and solar panel as EPS. Moreover, since their functions should be mutually complementary, a better combination seems to be in parallel. The question arises whether battery can be re-charged when solar panel is supplying electricity and supply energy when the solar panel is turned off.

Assumption A8 ('solution'): Combine battery and solar panel in parallel as EPS;

Assumption A17 ('solution'): When the sunshine is present $\Rightarrow$ solar panel supplies electricity; when the sunshine is not present $\Rightarrow$ battery supplies electricity;

Assumption A18 ('solution'): When the solar panel supplies electricity $\Rightarrow$ battery must re-charge to the initial capacity; when the solar panel does not supply electricity$\Rightarrow$ battery supplies it and discharges;

Assumption A19 ('requirement'): The initial capacity of the battery should be such that battery can supply power for at least 30 minutes (sunshine note present) – $C_0 > 20$ F.

Assumption A20 ('solution'): Let us try the initial capacity of a battery C0 = 21 F; the weight of a battery will be $M_{BAT} = 30$ kg.

Assumption A21 ('solution'): Let us check the weight of the solar panel assembly and the total weight of EPS – $M_{EPS} = M_{BAT} + M_{SOL} \cong 45$ kg (satisfies requirement A11).

Another question arises how to switch between two different power sources. The system may retrieve an example where a switch is used to turn one source of supply on and the other off. It would be possible to go on and complete the design problem. However, the purpose of this section is to serve as a pilot study assessing the viability of proposed design guidelines. It is possible to see that the selected approach to the design support is good and worth further investigation and development. More detailed study of the deployment of the support system will be done in a later stage of the project using an implemented prototype.

VIII. DISCUSSION AND OUTCOMES

This document tackled some issues regarding intelligent knowledge-based support for the engineering design. Firstly, it reviewed the nature of problems in design and formulated some important implications for the design support. These findings were used to justify an approach seeing design as a knowledge-rich activity that depends heavily on the ability of designers to organise their knowledge, use different knowledge sources, and draw analogies to the previously solved design situations. It was shown that a deep immersion and understanding of the existing domain knowledge might eventually lead to the emergence of innovative or creative solutions. Although creativity and innovation are important in the design tasks, they are not sufficiently explored; not mentioning providing support for them.

The document described one particular methodology that is applicable as a core for the knowledge-based support system in design, namely knowledge modelling and common ontologies. Common ontologies can be developed from many different points of view; the knowledge bases constructed on top of them can cover a broad area of designers' expertise and incorporate the rich structure of design knowledge. Eventually, ontologies may simplify the designers' work. However, the process of formalising a design situation using ontologies is not straightforward. It is essentially an iterative process, where one of the important features causing iterations, is the designer's reflection on the actions performed. Iteration and designers' ability to draw upon their previous experience in tackling similar situations in the past were identified as key issues that must be born in mind when 'designing' an intelligent support system for the design.

Based on the sequence of common actions during a design process, design guidelines were suggested and justified. The guidelines were consequently verified using a practical design scenario from numerous design literature. Such example served as a 'mini-pilot study' and it was proved that the chosen track in pursuing the issue of design support was viable and worth further investigation. Although human creativity is a very complex phenomenon, there is a hope that research activities like this may shed some light on the possible ways of its support and further development.

A. Expected outcomes

Case-based reasoning and analogous thinking are powerful techniques that are able to tackle 'real world' problems more efficiently than traditional 'rule-based' systems. However, there appears to be a gap in the research activities directed at the use of analogous thinking in design and there is lack of support for the early phases of the design process where analogy plays an important role. Designers start their work from vague descriptions of both, requirements and solutions; most of the available analogous methods are practically useless until there is enough information available. Therefore the first strand of further research will be oriented toward the support for the mapping a design situation on a common ontology based on a typical professional strategy of analogous thinking.

Objective 1:
Provide a set of guidelines and support tools helping designers with mapping their current design situations on existing ontologies, which involves the following activities:

a) Retrieval of similar design situations in form of design stories from the repository;
b) Re-use of patterns successfully applied in the previous solutions;
c) Iterative refinement of the current solution pattern and retrieved stories.

Iterative nature of the design process will be reflected in the development of techniques and methods for an efficient navigation of designers during the exploratory process of designing an artefact from initial requirements toward final specifications of both, requirements and solutions.

Objective 2:
Develop a feasible methodology incorporating the co-evolution of requirements and solutions during the design process, and relate it clearly with the existing theories of design.

Objective 3:
Research the possibilities how a co-evolution of requirements and solutions may be achieved, and eventually supported by a knowledge-based design support system, which includes also the following activities:

a) Applicability of completeness/consistency checking for the introduction of new assumptions (requirements, solutions, or domain knowledge);
b) Influence of analogous design stories on the generation/modification of assumptions;
c) Creation and re-use of justifications and reasoning process during design.

The interaction between collaborating human designers, or a human designer and a computational support tool are usually very complex. They may serve as a topic of a stand-alone research in general. However, the last objective aims at identifying possible issues with the deployment and use of such systems, and sketching a methodology providing the guidelines to avoid most visible issues.

Objective 4:
Implement the identified activities and produce a methodology for the deployment and position of this kind of knowledge-based design support systems.

As a base language for the support tool development will serve Common Lisp together with the extensions and plug-ins that were developed at the KMi in the past. Lisp module will be responsible for the 'intelligent' and reasoning part of the entire support tool. As it was identified we need quite a powerful engine for the representation of cases and retrieval of previous cases based on their similarity to the current one. Significant role has also the presentation of knowledge about the current and previous design problems. A user-friendly interface will be developed; probably an object-oriented programming language such as Java will be used. Use of Java (Java-like) language is advantageous also from other point of view – there will be directly provided support for working with remote knowledge sources using Internet/Intranet. From 'remote knowledge sources' there is a little step towards collaboration and group work in order to tackle more complex design tasks.

B. Research methods

To achieve the objectives identified above the future research will be carried out in several parallel strands. First of all, the guidelines and methodology for their application will be extended. Analysis and comparative studies of existing approaches will serve as a base for both – definition and modification of the design guidelines, and their rigorous evaluation. Eventually, a framework unifying reflective (Schön) and inventive (Altshuller) approaches toward engineering design is expected as a main outcome of the research. An important portion of rigorous assessment will be the comparison of the developed methodology with several psychological models fostering application of creative principles and methods (such as synectics, brainstorming etc.) in problem solving activities.

To test and evaluate the suggested methodology a toolkit providing support for design will be implemented. It will use the existing in-house technologies for user-friendly and powerful knowledge-level modelling of tasks and domains, as well as in-house expertise in the development of client interfaces and reasoning tools. The viability of such toolkit will be assessed by at least one study, where the toolkit will be deployed in its native environment and used by designers in tackling real-world design situations. Direct observation of designers' actions and consequently semi-structured interviews should help to surface the remaining issues as well as the usability of the entire approach to the knowledge-based support for knowledge-intensive human activities such as engineering design.

IX. REFERENCES

Altshuller, G.S. (1984). Creativity as an Exact Science. USA, Gordon & Breach Science Publishers.

Archer, L.B. (1970). An overview of the structure of design process. Emerging Methods in Environmental Design and Planning. G.T. Moore (Ed). Cambridge, MA, MIT Press: 285-307.

Bhatta, S. and Goel, A. (1994). "Discovery of Physical Principles from Design Experiences." Artificial Intelligence for Engineering, Design, Analysis and Manufacturing **8**(2).

Bhatta, S., Goel, A. and Prabhakar, S. (1994). Innovation in analogical design: A model-based approach. 3rd International Conference on AI in Design (AID'94), Lausanne, Switzerland.

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., Josephson, J.R. and Benjamins, V.R. (1999). "What Are Ontologies, and Why Do We Need Them." IEEE Intelligent Systems & their applications **14**(1): 20-26.

Choueiry, B.Y., McIlraith, S., Iwasaki, Y., *et al.* (1998). Thoughts Towards a Practical Theory of Reformulation for Reasoning about Physical Systems. Stanford, USA, Knowledge Systems Laboratory.

Cross, N. (1997). "Descriptive models of creative design: application to an example." Design Studies **18**: 427-440.

Dominowski, R.L. and Dallob, P. (1995). Insight and Problem Solving. The Nature of Insight. R.J. Sternberg and J.E. Davidson (Eds.). USA, MIT Press**:** 33-62.

Dzbor, M. (1999a). Intelligent Support to Problem Formalisation in Design. 3rd IEEE Conference on Intelligent Engineering Systems (INES'99), Stara Lesna, Slovakia.

Dzbor, M. (1999b). Knowledge-intensive design support: An enquiry into the role of ontologies and analogous thinking in the design support. (Literature Review), KMi, The Open University, UK.

Dzbor, M. (1999c). Support for Problem Formalisation in Engineering Design: An Enquiry into the Role of Knowledge-level Models. 10th International DAAAM Symposium, Vienna, Austria.

Fischer, G. (1992). Domain-Oriented Design Environments. 7th Knowledge-Based Software Engineering Conference (KBSE'92), IEEE Computer Society.

Gero, J.S. (1990). "Design prototypes: A knowledge representation schema for design." AI Magazine **11**(4): 26-36.

Gruber, T.R. (1993). "A Translation approach to Portable Ontology Specifications." Knowledge Acquisition **5**(2): 199-221.

Iwasaki, Y., Fikes, R., Vescovi, M. and Chandrasekaran, B. (1993). How Things Are Intended to Work: Capturing Functional Knowledge in Device Design. International Joint Conference on Artificial Intelligence (IJCAI'93).

MacCallum, K.J. (1990). "Does Intelligent CAD exist?" Artificial Intelligence in Engineering **5**(2): 55-64.

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.

Nakakoji, K., Sumner, T. and Harstad, B. (1994). Perspective-based critiquing: helping designers cope with conflicts among design intentions. AI in Design'94.

Newell, A. (1982). "The knowledge level." Artificial Intelligence **18**(1): 87-127.

Nidamarthi, S., Chakrabarti, A. and Bligh, T.P. (1997). The significance of co-evolving requirements and solutions in the design process. 11th International Conference on Engineering Design (ICED'97), Finland.

Prabhakar, S. and Goel, A. (1998). "Functional modeling for enabling adaptive design for new environments." Artificial Intelligence in Engineering **12**: 417-444.

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.

Schön, D.A. (1983). Reflective Practitioner - How professionals think in action. USA, Basic Books, Inc.

Simon, H.A. (1973). "The structure of ill-structured problems." Artificial Intelligence **4**: 181-201.

Smithers, T., Conkie, A., Doheny, J., *et al.* (1990). "Design as intelligent behaviour: an AI in design research programme." Artificial Intelligence in Engineering **5**(2): 78-109.

Tang, M. (1997). "A knowledge-based architecture for intelligent design support." Knowledge Engineering Review **12**(4): 387-406.

van Heijst, G., Schreiber, A.T. and Wielinga, B.J. (1997). "Using explicit ontologies in KBS development." International Journal of Human-Computer Studies **46**(2/3): 183-292.

Vescovi, M., Iwasaki, Y., Fikes, R. and Chandrasekaran, B. (1993). CFRL: A Language for Specifying the Causal Functionality of Engineered Devices. 11th National Conference on AI (AAAI'93).

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**: 59-87.