The Role of Knowledge in Design Problems

Zdenek Zdrahal

Knowledge Media Institute, The Open University, UK

Abstract. The paper presents design as an example of ill-structured problems. Properties of ill-structured problems are discussed. French's model of design processes is analysed. The role of domain knowledge as a means for structuring the problem space is explained. Design process can be viewed as a sequence of problem re-representations gradually reducing problem indeterminacy. The results are demonstrated on pilot applications developed as a part of the Clockwork project.

1 Introduction

In this paper we discuss two issues related to knowledge sharing and reuse in engineering design. First, we show why domain knowledge is an essential component of problem solving in design. We will argue that knowledge provides the necessary structure for problem spaces. Then we will interpret the process of designing tasks as a sequence of problem re-representation where each subsequent problem specification is better structured. The described topics motivated the CEC funded Clockwork project whose objectives included the support for reuse of design knowledge.

2 Design as an Ill-Structured Problem

In mid sixties, when studying planning tasks, Rittel and Webber [18] noticed that most problems do not follow the linear "waterfall" model consisting of data analysis, problem specification and problem solving. In particular, problem specification and problem solving are mutually intertwined - problem cannot be completely specified without committing solution to the concrete problem solving method and the problem solving method cannot be selected without a complete problem specification. Similar characteristics were also observed in other domains such as architecture, urban planning or design. The problematic domains very often included social context. Solving this class of problems is difficult because at least a conceptual or tentative solution must be assumed as a part of problem specification. Rittel and Webber call these problems "wicked" to distinguish from standard problems which they call "tame". According to Rittel and Webber [18] p.161, the class of wicked problems is characterised by the following ten properties:

1. There is no definitive formulation of a wicked problem. The information needed for understanding the problem and formulating its specification depends on the idea of how to solve it.

R. Wagner, N. Revell, and G. Pernul (Eds.): DEXA 2007, LNCS 4653, pp. 884–894, 2007.

2. Wicked problems have no stopping rules. There is no criterion or test proving that the solution has been found.
3. Solutions to wicked problems are not true-or-false, but good-or-bad. When solving mathematical equations, the result is either correct or incorrect i.e. true or false. Such a dichotomy does not apply to wicked problems. Solutions could be good, bad, better, worse or perhaps only good enough.
4. There is no immediate and no ultimate test of a solution to a wicked problem. Solutions may produce unexpected consequences that are not obvious when the solution attempt is made. These even might not be immediate but they would emerge sometime in the future.
5. Every solution to a wicked problem is a "one-shot operation". There is no opportunity to undo already implemented solutions.
6. Wicked problems do not have an exhaustive set of potential solutions nor is there a well-described set of permissible operations that may be included into the problem solving plan.
7. Every wicked problem unique.
8. Every wicked problem can be considered as a symptom of another problem. Wicked problems may create a causal chain.
9. The existence of a discrepancy representing a wicked problem can be explained in numerous ways. The explanation depends on the "world view" of the problem solver.
10. The wicked problem solver has no right to be wrong. This point is especially important in social policy planning where the ultimate objective is not to find the truth but to improve the current state of public affairs.

These characteristics of wicked problems were formulated for social policy planning, but similar properties were also identified in other areas including design [2], [3]. For example, Schön [20], p.79 describes how the "designer shapes the situation and the situation 'talks back'", or that problem "naming, framing and, if needed, re-framing" is a way of dealing with the lack of definitive problem formulation.

For engineering design not all characteristics described above are equally important. The concept of "wicked problem" emerged from studying problems with social connotation where social criteria play important part of the problem domain. Planning city infrastructure, proposing solutions to the problems of poverty, education, crime or health care are typical examples that manifest wickedness. Conklin even explains that social complexity of design problems relates to the "measure" of their wickedness [3]. However, we claim that social context does not have the same impact in every domain. While it is certainly the key issue in policy planning, there are areas, where it is less important. This becomes obvious when we try to apply the above mentioned characteristics in the context of engineering design. When a planner tries to resolve a societal problem it is usually a one-shot operation. The solution cannot be withdrawn because it creates permanent or long lasting changes. Also, the solution must not be completely wrong because there is too much at stake. However, in engineering design unsuccessful trials that fail to solve the problem frequently exist. Thomas Alva

Edison was a pioneer of designing by trial and error and yet he patented over 1000 original designs addressing problems that certainly satisfied multiple properties of wickedness. Trials and errors are acceptable if the situation is reversible and return is not too expensive. The cost depends on the feedback loop through which the solution is evaluated. It is probably impossible to completely rebuild already constructed road system, because it would be too costly. It is certainly impossible to correct a medical therapy if the patient has died. But it is easy to redesign breaks for the bicycle, especially if the bicycle has not yet been put on the market. Conklin [3] demonstrates the properties of wicked problems on designing a new car with the aims of improving side-impact safety. In his example, the success of a car is measured by the response of the market. When a new model is launched, the customers will either accept or reject the product, but the innovative project cannot be turned back and the preparation of manufacturing technology cannot be undone. In this sense car design is a one-shot operation. However, there is a number of counterexamples when manufacturers had to recall hundreds of thousands of cars to correct design errors. The "second shot" was certainly very expensive, but it has been done. If the evaluation loop is short and cheap, the problem solving allows proposing, evaluating and rejecting alternatives. Returning to the car example, there are various legislations, national and international regulations that constrain possible designs before they reach the market, even before a prototype hits the road. Within these regulations, design by educated trial and error is a common practice. Yet these design problems retain the majority of characteristics of wickedness. Though design problems are presented as unique, similarities with other problems play an important role. How would otherwise designers develop their expertise if problems were always the "universe of one" [19]? It is well known that designers often acquire their inspiration from already existing cases. Schön [20] argues that in professions, unique problems are common, but they are often approached by adapting and combining known solutions.

Simon analyses similar class of problems but without emphasising the social context [21], [22]. He introduces the concepts of ill-structured and well-structured problems with a meaning similar to wicked and tame problems. In analogy to the way used by Alan Turing who formalised the intuitive concept of algorithm in terms of Turing machine, Simon associated well-structured problems with the General Problem Solver (GPS) [15]. This formalisation makes it possible to introduce the concepts like state space, states, state transitions and structures. Well-structured problems are defined by the following conditions (adapted from [21]):

1. There is a definite criterion for testing proposed solutions
2. There is a problem space in which it is possible to represent the initial problem state, the goal state and all other reachable states.
3. Attainable state transitions can be represented in a problem space.
4. Any knowledge that the problem solver can acquire can be represented in a problem space

5. If the problem depends on input from the external world, then the state transitions reflect accurately the laws of nature that govern the external world.
6. All these conditions are practically computable.

The problems that do not satisfy one or more of these conditions are called ill-structured. The correspondence with many of characteristics of wicked problems is straightforward. It makes sense to consider the measure to which the real problem satisfies these conditions. We can then position the problem on a scale with ill-structured problems at one end and well-structured on the other one.

When solving a problem each method has associated preconditions of its applicability. Since ill-structured problems are not fully specified these preconditions are impossible to evaluate. The preconditions specify what kind of information about the problem space is needed and, consequently what is the quality of the solution delivered by the method. For example, solutions can be optimal globally, locally or only "good enough", methods can guarantee that if a solution exists it will be found, they may search a large state space or progress directly to the solution, they may convergence slowly or quickly. There are many other measures that assess the quality of solution. In general, the more a priori information about the problem space is available, the better results are delivered by the method [16]. If the problem is well structured, i.e. the problem space is well described and the quality criterion is known, it is possible to apply "strong" problem solving methods, such as optimisation algorithms. However, since design is an ill-structured problem, generally applicable strong problem solving methods do not exist. But why is design an ill-structured problem? Buchanan [2], p.15 argues that the reason is that "design has no specific subject matter of its own apart from what the designer conceives it to be". It means that designer has to bring his/her domain knowledge to structure design tasks. This observation has important consequences for knowledge management in design processes.

Imagine an engineering problem of designing a bridge over a river. The problem can be certainly regarded as ill-structured. Possible solutions include a beam bridge, an arch bridge, a suspension bridge or perhaps even a pontoon bridge. There are many other bridge types, each of them having its pros and cons. By making the decision and selecting one type, the designer brings into the design process the corresponding part of physics and provides structure for the problem space. If the designer opts for a suspension bridge, he/she will have to express the problem in terms of concepts and laws of theory of elasticity. If he/she decides to construct a pontoon bridge, he/she will use theory of hydrostatics and hydrodynamics. The progress in problem solving makes it possible to improve the problem specification (e.g. in the case of the suspension bridge the task of calculating the strength of cables is a part of the additional problem specification). Problem solving is intertwined with problem specification as the problem acquires structure and moves from ill-structured towards well-structured end of the scale.

For solving wicked problems Kunz and Rittel [13] proposed the method known as Issue Based Information Systems (IBIS), that allows the designers to uncover

conflicting criteria in the reasoning process, capture the argumentation, and arrive at an acceptable solution. A number of variations of IBIS-based methods focused on design problem solving are discussed in [14]. Though these methods successfully capture design rationale, they provide only limited support to converting the ill-structured design problem into a well-structured one because they not offer any means for integrating new domain knowledge with the existing problem specification. Mismatch of expectations and real benefits of IBIS-like methods are analysed in [12].

3 Models of Design Processes

In engineering, various models of design processes have been proposed. Typically, design starts with a problem analysis and ends with a detailed specification for the artefact. At the beginning, the problem is indefinite, its specification is incomplete and therefore, the problem space is ill-structured. At the end the problem space is already well-structured and the artefact is fully specified at the required level of detail. In order to structure the problem space, new knowledge need to be brought into the design process. The model of design process elaborated by French [9] is frequently used in engineering applications. This model is shown in figure 1 (a). French's model divides the design process into a number of distinct stages. They are: problem analysis, conceptual design, embodiment of schemes and detailed design. In the first stage, the designer analyses and sets the problem. The remaining three stages synthesize the solution. French admits that this model is one of many possible, however since it captures well our intuitive understanding of the design process, it has been adopted by many other authors, e.g. [5], [11], [23], [8]. We argue that French's model also well describes how the designer gradually commits the solution to the selected engineering domain. This can be manifested by the increasing use of domain knowledge and the decreasing indefiniteness of problem specification as the design process progresses through consecutive stages. This is highlighted in figure 1(b). Model in figure 1(a) describes the design process as a sequence of steps with possible feedback (progress in vertical direction). However, each step is also a design process on its own. These "horizontal" processes are shown in figure 1(b). For example, conceptual design is a process which produces a conceptual solution to the problem in terms of schemes or other conceptual objects. Similarly, embodiment of schemes and detailing are design processes that provide the conceptual solution with more detailed structures and elaborate final details.

Each stage produces a complete solution to the design problem, but the solutions differ by the coarseness of detail due to different indeterminacy of problem specification. This does not imply that the solution at the higher design stage must be finalised before the lower stage starts. The most important decisions are made in early stages. In final stages, the problem is already well-structured and strong problem solving methods, such as optimisation procedures are likely to be found. Conceptual design can be very creative but provides only limited guidance for problem solving (Newell's weak methods) because the problem is

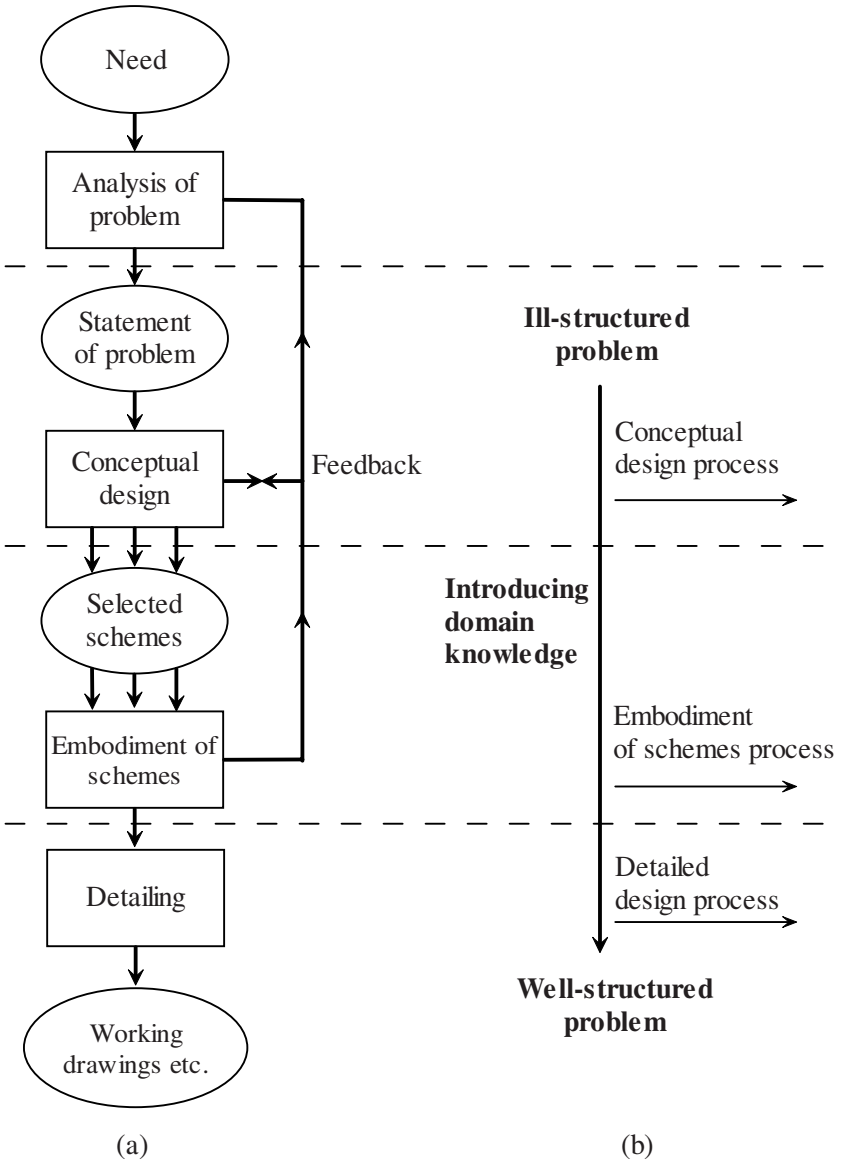

Fig. 1. (a)Model of design process according to French, (b) Introducing domain knowledge into process

ill-structure and there is no commitment to a specific domain. Goel [11], p. 131 characterises conceptual design as follows: " Generation and exploration of alternatives is facilitated by the abstract nature of information being considered, a low degree of commitment to generated ideas, the coarseness of detail and a large number of lateral transformation. A lateral transformation is one where

movement is from one idea to a slightly different idea, rather than a more detailed version of the same idea. Lateral transformations are necessary for widening the problem space and the exploration and development of kernel ideas". In general, design decisions made during early design stages direct the solution towards a specific domain and make the corresponding domain knowledge available for the later design stages. This principle applies throughout the design process. Domain knowledge available at each stage affects the repertoire of applicable problem solving methods. Problem solving methods typically used in early design stages differ from those applicable later. In engineering practice, creativity is often restricted to selecting and exploring a few most promising alternatives using case-based or IBIS-like problem solving methods. This was reported by Ball, Ormerod and Morley [1] who observed that designers solve new problems by looking for analogies with old ones. Moreover, they showed how designers' experience changes their reasoning processes: Experts develop and use more abstract knowledge schemas while novices tend to draw the analogies from specific cases. Cross [7] arrives at similar conclusions claiming that the experienced designers tend to keep design ill-structured in order to have more opportunities for creative thinking while novices quickly moves towards the first possible solution. Goel [11] demonstrates that different parts of brain are specialised in solving different design stages. Right hemisphere is predominantly used for ill-structured problems i.e. for conceptual design, while left hemisphere supports mental activities needed for solving well-structured problems, i.e. methods used in detailed design.

4 Solving Design by Problem Re-representation

Design process based on French's model can be viewed as problem re-representation. First, a tentative conceptual model is built from the initial problem specification . The choice of conceptual objects and relations is based on designer's assumptions about the expected solution. Conceptual solution is further elaborated by specifying the structure and components to be used for implementing. A new set of assumptions is needed, now at a different level of abstraction. Finally, the parameters of selected components are specified. Their calculation may require a few additional assumptions. In general, design decisions have the form of assumptions.

Problem re-representation is also used in a compilation and linking of computer programs. The algorithm represented by a source code, say in the C language, is re-represented step by step until the absolute machine code is achieved. However, there is a difference: when compiling a C program, all necessary information is already contained in the source code. The problem is well-structured from the very beginning and there is no need for supplying additional information into the compilation/linking process.

In design, on the other hand, the problem is initially ill-structured. Each re-representation step converts the results of the previous stage but require also additional assumptions about the solution in next step. These assumptions add

new information into the design process. For example, when constructing a model of car suspension, the designer may assume that the elasticity of the body can be omitted. At the current stage of problem solving, this cannot be proved or disproved but the design must make tentative decision, otherwise the process cannot continue. Design decisions are based on formal and experiential knowledge of the designer. Capturing designer's most important and innovative decision and making them available in the future was one of the major objectives of the Clockwork project described in the following section.

5 The Clockwork Project

Structuring design as problem re-representation was the basis of the CEC funded project "Creating Learning Organisations with Contextualised Knowledge-Rich Work Artifacts" (Clockwork). The project objectives included the development of methodology and support tools for sharing and reuse of design knowledge. The support tools were integrated in a configurable web-based toolkit, called the Clockwork Knowledge Manager (CKM). The main focus of the project was on modelling and simulation models of dynamic systems and therefore, the first pilot application was from this area. Since the development of simulation models has all features of design, the Clockwork approach can be applied to other areas of engineering design. This was tested by the second pilot application in the area of designing industrial thermal technologies. Both pilot applications were developed as instantiations of CKM. In Clockwork, design stages are called *worlds*. Each world has associated world objects and relations specified in terms of domain ontologies.

Clockwork methodology provides a number of extensions on top of French's model. For example, the world representation can be also used for stages and environments that are not part of the French's model. Since it might be convenient to collect post-design data about the designed artefact, the CKM application can be extended by Production, Installation or Maintenance worlds. The collected data may serve to inform future designs. Similarly, in modelling and simulation applications, the task can be to simulate a real object. In CKM, all objects to be simulated could be described in a Real world. Example of four worlds used in the modelling and simulation application is shown in figure 2. In this appli-

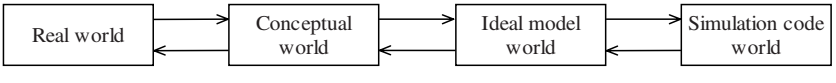

Fig. 2. Four world model of modelling and simulation

cation the problem to be investigated by computer simulation is introduced in Real world. For example, imagine that the task of designing a car suspension that guarantees passenger comfort within certain speed range on standard types of road. Though concepts like passenger comfort and road type are likely to

be defined by international, national or manufacturer's norms, the problem is clearly ill-defined. There is a number of possible conceptualisations of the problem. For example, the suspension system can be modelled using the concept of a "quartercar" (model of a single wheel), quartercar models can be connected by a set of rigid or elastic bodies, the whole suspension can be modelled using finite elements etc. Each conceptual solution can be further elaborated, for example models can be built using ideal physical models, bond graphs, etc. These will be eventually converted into simulation code. Most important decisions for the final outcome are the assumptions justifying conceptualisation and modelling approaches. Re-representation from model world to simulation world usually does not require any additional design-related assumptions because the problem is already well structured. For this reason, simulation packages often allow the user to build the model in a graphical environment and the simulation code is uploaded automatically. CKM toolkit not only supports the integration of domain ontologies with the design process, but also capturing design decisions, their assumptions and consequences at more detailed level. In order to minimise additional tedious work of designer, the Clockwork methodology assumes that only important or non-trivial knowledge will be recorded. The Clockwork therefore does not build a complete mapping between adjacent worlds. An example of four world environment for modelling and simulation applications is shown in figure 3. CKM provides two mechanisms for representing case specific knowledge. Formal knowledge is represented by instances of ontology classes. In Clockwork, these instances are called *semantic indexes*. The relations between semantic indexes of adjacent worlds are parts of problem re-representation. Informal knowledge has the form of textual annotations and can be associated with transitions between whole models or with individual semantic indexes. CKM in-

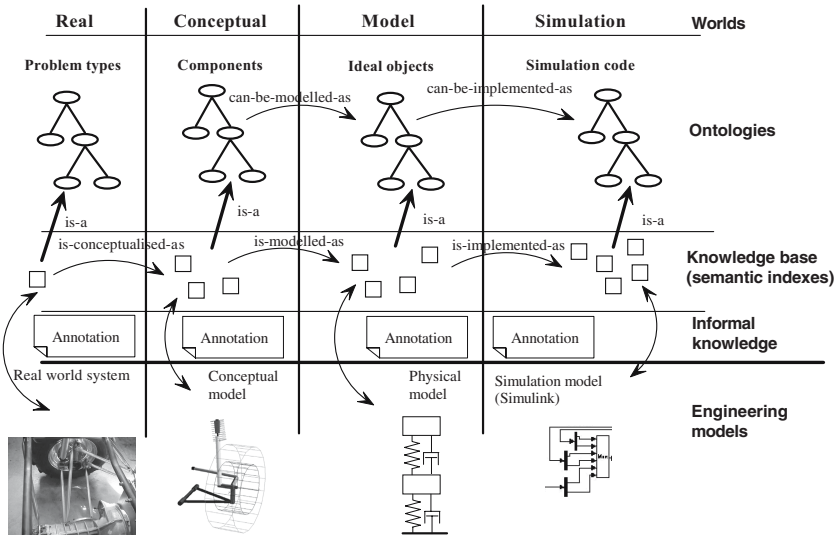

Fig. 3. Modelling and Simulation application

tegrates formal and informal knowledge with standard work representation such as technical drawings, engineering diagrams, conceptual graphs or graphical environments. The application to thermal machinery design and production was also developed by instantiating CKM. The configuration includes worlds shown in figure 4. The design-related worlds follow from the French's model. The post

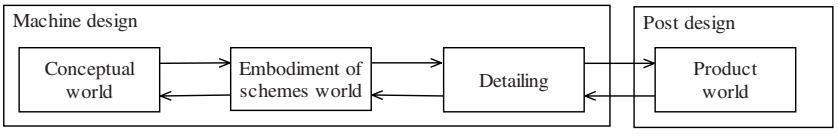

Fig. 4. Product design and post design worlds

design worlds collect data related to the machine installation and maintenance. The company design team uses these data for informing future design of machines with a similar specification.

6 Conclusions

Design is a human activity which aims at converting the current state of the world into a new, preferred one. Since we usually do not specify in advance how to reached the desirable outcome, the problem is ill-structured. Designers need their expertise to overcome the problem of indeterminacy and to interpret the problem. We have presented a methodology and a tool that supports knowledge reuse and sharing between designers. This overview could not present the problem in a full complexity. For example, we did not discuss the role of tacit knowledge in design, see [4], [17] or sharing design knowledge in settings where participants must collaborate but also compete on the same market (cautious knowledge sharing). These issues were also addressed by the Clockwork project The following persons significantly contributed to the Clockwork project: Paul Mulholland of The Open University, Michael Valasek of the Czech Technical University and Ansgar Bernardi of DFKI.

References

1. Ball, L.J., Ormerod, T.C., Morley, N.J.: Spontaneous analogising in engineering design: a comparative analysis of experts and novices. Design Studies 25, 495–508 (2004)
2. Buchanan, R.: Wicked Problems in Design Thinking. Design Issues 8(2), 5–12 (1992)
3. Conklin, J.: Wicked Problems and Social Complexity. In: Book Conklin J. Dialogue Mapping: Building Shared Understanding of Wicked Problems, John Wiley and Sons Ltd, Chichester (2005)

4. Cook, S.D.N., Brown, J.S.: Bridging Epistemology: The Generative Dance Between Organizational Knowledge and Organizational Knowing. Organization Science 10(4), 381–400 (1999)
5. Cross, N.: Engineering Design Methods. In: Strategies for Product Design, 2nd edn., John Wiley & Sons, Chichester (1994)
6. Cross, N.: Descriptive model of creative design: application to an example. Design Studies 18, 427–440 (1997)
7. Cross, N.: Expertise in design: an overview. Design Studies 25, 427–441 (2004)
8. Dym, C.L.: Engineering Design, A Synthesis of Views. Cambridge University Press, Cambridge (1994)
9. French, M.J.: Engineering Design: The Conceptual Stage. Heinemann Educational Books, London (1971)
10. Goel, V., Pirolli, P.: The Structure of Design Spaces. Cognitive Science 16, 395–429 (1992)
11. Goel, V.: Cognitive Role of Ill-Structured Representations in Preliminary Design. In: Gero, J.S., Tversky, B. (eds.) Visual and Spatial Reasoning in Design, Key Centre of Design Computing and Cognition, pp. 131–145. University of Sydney, Sydney (1999)
12. Isenmann, S., Reuter, W.D.: IBIS - a convincing concept . . . but a lousy instrument? In: Proceedings of the Conference on Designing interactive Systems: Processes, Practices, Methods, and Techniques, pp. 163–172. ACM Press, New York (1997)
13. Kunz, W., Rittel, H.W.J.: Issues as Elements of Information Systems. Working Paper WP131. July 1970, reprinted May 1979. Inst. Urban and Regional Development., Univ. Calif., Berkeley (1970)
14. Louritas, P., Loucopoulos, P.: A Generic Model for Reflective Design. ACM Transactions on Software Engineering and Methodology 9(2), 199–237 (2000)
15. Newell, A., Simon, H.A.: GPS, a program that simulates human thought. In: Computers and Thought, pp. 279–293. MIT Press, Cambridge (1995)
16. Newell, A.: Artificial Intelligence and the Concept of Mind. In: Schank, R.C., Colby, K.M. (eds.) Computer Models of Thought and Language, pp. 1–60. W.H. reeman and Comp., San Francisco (1973)
17. Nonaka, I.: The Knowledge-Creating Company. In: Harvard Business Review on Knowledge Management, pp. 21–45. Harvard Business School Press, Boston (1998)
18. Rittel, H.W.J., Webber, M.M.: Dilemmas in a General Theory of Planning. Policy Sciences 4, 155–169 (1973)
19. Schön, D.A.: Designing: Rules, types and worlds. Design Studies. 9(3), 181–190 (1988)
20. Schön, D.A.: The Reflective Practitioner. Ashgate Publishing Ltd., Aldershot, England (1991)
21. Simon, H.A.: The Structure of Ill Structured Problems. Artificial Intelligence 4, 181–201 (1973)
22. Simon, H.A.: The Science of the Artificial, 3rd edn. MIT Press, Cambridge (1996)
23. Tansley, D.S.W., Hayball, C.C.: Knowledge-Based System Analysis and Design. Prentice-Hall, Englewood Cliffs (1993)