Engineering with Computers (2000) 16: 209–235

Engineering with Computers

A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval

W. C. Regli[1], X. Hu[2], M. Atwood[3] and W. Sun[2]

[1]Geometric and Intelligent Computing Laboratory, Department of Mathematics and Computer Science, Drexel University. Philadelphia, PA, USA; [2]Rapid Product Development Center, Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA, USA; [3]College of Information Science and Technology, Drexel University, Philadelphia, PA, USA

Abstract. *This paper provides a survey on recent research in the area of design rationale. The study of design rationale spans a number of diverse disciplines, touching on concepts from research communities in mechanical design, software engineering, artificial intelligence, civil engineering, computer-supported cooperative work, and human-factors and human-computer interaction research. We focus this survey on prototype design rationale systems for these application domains, and put forward several major criteria with which to describe and classify design rationale systems, including argumentation-based, descriptive, process-based approaches. Further, we attempt to abstract the place of systems and tools for design rationale capture and retrieval in the context of contemporary knowledge-based engineering and Computer-Aided Design (CAD) tools. This survey is structured around classes of fundamentally different approaches, their representation schema, their capture methods and retrieval techniques. A number of recent design rationale systems, including JANUS, COMET, ADD. REMAP, HOS, PHIDIAS, DRIVE and IBIS are analysed. We conclude with an assessment of current state-of-the-art and a discussion of critical open research issues.*

Keywords. Design history; Design intent; Design rationale capture; Engineering design process; Knowledge management; Software design; User interface design

1. Introduction

Design rationale is an explanation of why an artifact, or some part of an artifact, is designed the way it is [1]. Design rationale includes all the background knowledge such as deliberating, reasoning, trade-off and decision-making in the design process of an artifact – information that can be valuable, even critical, to various people who deal with the artifact [2–6]. These views are consistent across the whole spectrum of engineering design disciplines, from mechanical design and architecture-engineering-construction to software and user-interface design.

Usually a developed artifact is defined in terms of specifications and parameters to describe the way it works, but does not include a description of *why* it is designed the way it is. Design rationale is usually not documented completely, and the collaborating work teams often need considerable amounts of communication to understand others' work [7]. Activities to maintain and redesign the artifact require much effort to understand the previous work. Many believe that keeping track of the design rationale will provide a great aid to designers: it helps to structure design problems, and provides a basis for designers to explore more design options. Design rationale systems can be used as a basis to discuss and reason among collaborating designers: to record the history of design process; to modify and maintain existing designs, or design similar artifacts [8,9].

Much progress has been made on the development of design rationale approaches and tools since the early 1980s. The research has ranged from basic observations about the design process [10] to different approaches to capturing design rationale. In this previous work, basic concepts were discussed and frameworks for design rationale were proposed. A number of important prototypes have been developed, but few design rationale systems have made it into practical use in industry. To this end, the fundamental goal of this survey paper is to answer the questions:

- If design rationale is useful (most agree on this

Correspondence and offprint requests to: Professor W. Regli, Department of Mathematics and Computer Science, Drexel University, Korman Center, Rm 238, Philadelphia, PA 19104, USA. E.mail: regl:@drexel.edu

point), why are design rationale systems not in widespread use in industry?
- How can design rationale systems better support engineering design?
- What are the major obstacles to the creation of truly useful and usable design rationale systems?

For a more comprehensive background reference of the design rationale field, interested readers are referred to the excellent 1996 edited collection by Moran and Carroll [11], as well as discipline-specific overview papers [12,13]. Recent research has a tendency to combine design rationale systems with other forms of design support tools [14,15]. Lee's [12] recent article describes some of the major open issues in design rationale research. We present this paper as a system-centred overview, with the goal of helping current and future builders of design rationale support environments sort through the many issues of system development. Furthermore, we put forth presently outstanding issues for design rationale research and practical system deployment and use.

This survey is organised as follows. Section 2 gives the framework of the survey, basic concepts of design rationale are introduced and design rationale systems or prototypes are reviewed in chronological order and summarised. Section 3 describes basic approaches to design rationale. Section 4 describes the representation schema of design rationale systems. Sections 5 and 6 describes the capture and retrieval of design rationale, respectively. Section 7 touches on some of the related research areas that support the study of design rationale. Section 8 provides brief examples of several different uses and applications for design rationale systems. In Section 9, we summarise our review, and discuss the open research issues that must be addressed to advance the state-of-the-art in design rationale systems.

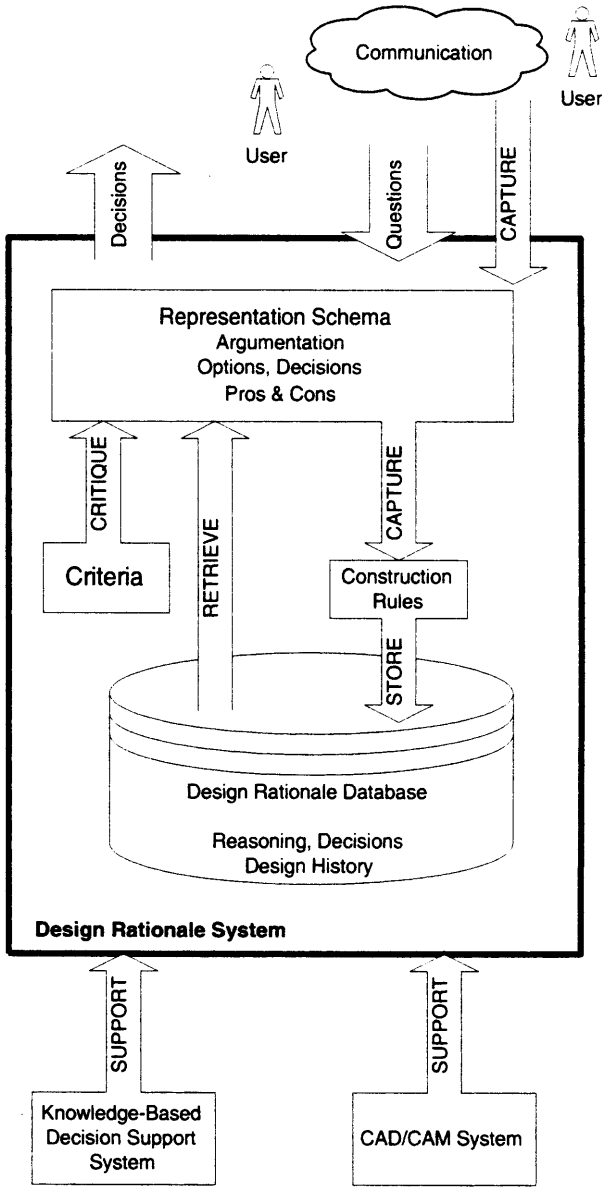

Fig. 1. Flow of information in a design rationale system framework.

2. A Framework for Design Rationale

The research on design rationale covers a large range of topics; this survey focuses on the review and analysis of existing systems and prototypes. In this context, *a design rationale system intends to let designers think and discuss design within a certain knowledge representation framework.*

Figure 1 illustrates how a prototypical design rationale system works. Based on a *representation schema,* the *decisions and reasoning* are either *recorded* by designers themselves using documents, or *captured* by a design rationale system's monitoring module. The capture of design rationale can occur from the design teams' communications via Computer-Supported Cooperative Work (CSCW) tools, such as electronic mail, phone conversions, and archived design meetings and designers' notebooks. The design knowledge captured is organised by *rules,* which are primarily determined by a *representation schema,* and stored into the *design rationale sytem* knowledge-base – to then be used in later design activities. If a design is similar to an existing design, the design rationale for the for-

mer design can be retrieved from the design rationale database, providing the suggestions and ideas relevant for the current design task. Some systems (e.g. the CRACK system [16]) also include critiquing tools, which are based on domain knowledge and can justify design decisions. Generally, a design rationale system is supported by knowledge-based systems and Computer-Aided Design and Manufacturing (CAD/CAM) systems, opening up a wide range of possibilities in supporting design activities.

We break the core of our survey down based on major systems and prototypes, and examine the four following issues.

Approaches to developing design rationale systems. Since the first approach to design rationale was proposed in 1970 [17], various other approaches have been developed. The main approaches are *process-oriented* and *feature-oriented*. In fields with a relatively high degree of standardisation, the feature-oriented approach is frequently used. Feature-oriented approaches focus on the representation of artifacts and the body of established rules governing the design process. In dynamic design domains, in which accepted design principles may not be well established, the process-oriented approach is often used to create a historical representation of the design process [2,3]. The integration of design rationale systems with other design support tools (knowledge-based, CAD and collaborative work) also influences the approach.

Representation schema for design rationale. A design rationale representation explicitly documents the reasoning and argumentation occurring in design [5]. The representation determines the methods used to capture and retrieve the design rationale, so it is important to select an appropriate representation schema.

Argumentation-based design rationale is a mainly *representational approach* which uses a semi-formal graphical format [2] for laying out the structure of arguments. It uses a node-and-link representation, which means it uses typed links to interconnect typed nodes [18]. A node represents a component, while a link represents a relationship. With argumentation, designers can easily maintain consistency in decision-making, keep track of decisions, and communicate about design reasoning with each other. The most common argument structures for selecting and organising information are IBIS, PHI, QOC and DRL.

Some systems use a *descriptive approach* [3] to represent design rationale, which records the history of design activities, work flow and communication between designers [18]. More specifically, it records what decisions are made, when they are made, who made them, and why [18]. This approach is used in dynamic design domains, in which the problems are vague and the solution technology is poorly understood, and therefore little or no standardisation of designed artifacts exists. The descriptive approach emphasises minimising the intrusion to designers by making design rationale capture as transparent as possible. This makes re-usability of the design rationale incidental [3].

Capture of design rationale. In a design process, design rationale is captured by recording reasoning, decisions, options, trade-offs, etc., and constructing a formal [9] or semi-formal structure [2] so that the design rationale can be used in the decision-making process during design.

Determining what information to capture during design, and how to capture it, is a fundamental problem many have tried to address. Gruber [19] proposed that rationale support tools should focus on capturing the information that is used to answer designers' questions, as well as data that might be used to infer answers to later questions. The dependency relations among the data and information are very useful, and should be captured as well. In this environment, rationale explanations are constructed in response to information requests, from background knowledge and information captured during design.

Applying this philosophy in a design environment using traditional CAD drawing tools, which are focused on detailed design and only capture data about the physical or logical aspects of an artifact, the design rationale support tools should capture information such as the intended behaviour or function of a device, and the justification for design decisions [4].

The design rationale capture process generally consists of two steps: *knowledge recording* and *design rationale construction*. The first involves capturing as much raw information as possible during the design process. The second step is the extraction, organisation and storage of rationale knowledge, which is based on the design rationale representation schema and constraints in the design domain. We observed that the recording and construction processes have been implemented in design rationale systems through both *automatic capture* approaches and those requiring *user-intervention* (in which designers are required to input

or record the design discussions, decisions and reasonings themselves).

Retrieval of design rationale. The retrieval of design rationale is determined by the representation schema and the requirements of the engineering domain for which the knowledge is being used. At each different design stage, there are various purposes for accessing design rationale: to answer a user query, to show the logical aspects of an important issue, to monitor design process, or to get a document about the designed artifact [31,27,33]. Different access strategies are needed to help users with different purposes. The retrieval of design rationale can be *active* or *passive*. It is desirable that some design rationale retrieval be triggered *automatically* according to the design context [24].

Given a representation schema, tools supporting functions such as *navigation by designers* [16] and *retrieval strategies* [27] can be implemented. The integration of design rationale systems with other design support systems can greatly improve the retrieval of design rationale. The retrieval of design rationale in such systems is usually involved in the design reasoning process, and supported by knowledge-bases in the design support systems.

A design rationale system is not effective as a standalone system. Together with other design support systems, such as CAD or Computer-Aided Software Engineering (CASE) tools, it contributes to the design process by providing designers with a knowledge representation framework, as well as tools to capture design rationale, design reasoning and communication during the design process. Ideally, such systems can transform raw design rationale and design history into knowledge for later reuse – providing facilities to retrieve design rationale when needed for review, maintenance or redesign. Figure 2 shows the architecture of a general design rationale system, generated from our survey by synthesising the functions of different systems. The design rationale systems or prototypes that we reviewed generally did not include all of the features indicated in the diagram.

To provide a general view of the development of the research area, we list the design rationale systems reviewed (in chronological order) according to the catalogue described above in Table 1. Each system is described in five dimensions: *knowledge representation, capture, retrieval, approach* and the *application domain*. The following sections will concentrate on a detailed discussion of the systems in the table.

3. Approaches to Building Design Rationale Systems

At different stages of the design process of an artifact, design can be more *process-oriented* or more *feature-oriented*. At the initial design stage, as the design progresses from the requirements to a conceptual design, there are many discussions concerning the requirements (which may not be well specified), and much exploration of options and trade-offs since there may be no fixed solution path [34]. This design process is more dynamic, with knowledge organised according to design progress, so the approach of design rationale at this stage is more process-oriented. At the detailed design stage or in routine design, design process is more constrained by the rules in the field or domain knowledge, since it is feature-oriented, [24,32], in which features could be function, performance, design, manufacture or implementation. These differences in the characteristics of the two different phases of design naturally lead to the two different approaches to design rationale.

3.1. Process-Oriented Approaches

Process-oriented design rationale systems emphasise the design rationale as a *history of the design process*. Process-oriented design rationale has been most often used in domains where the problems are vague, the solution technology is poorly understood, or both, and in which there is little or no standardisation of designed artifacts [3]. Unlike feature-oriented systems, which construct design rationale as a logical structure, design rationale is merely descriptive in a process-oriented system. Most existing design rationale systems are process-oriented, with issues, options and arguments captured and organised according to the design progress.

This approach originated from the Issue-Based Information System (IBIS) framework for argumentation [17]. A number of other frameworks have been developed since then, including DRL [1] and PHI [20].

The representation schema of this kind of rationale is generally graph-based, using nodes and links, with nodes indicating issues (questions), positions (options) and arguments, and links indicating the relationships among the nodes. This kind of representation schema provides a flexible structure and great convenience in recording design rationale from communications of design progress. This is especially true for multimedia communications, since

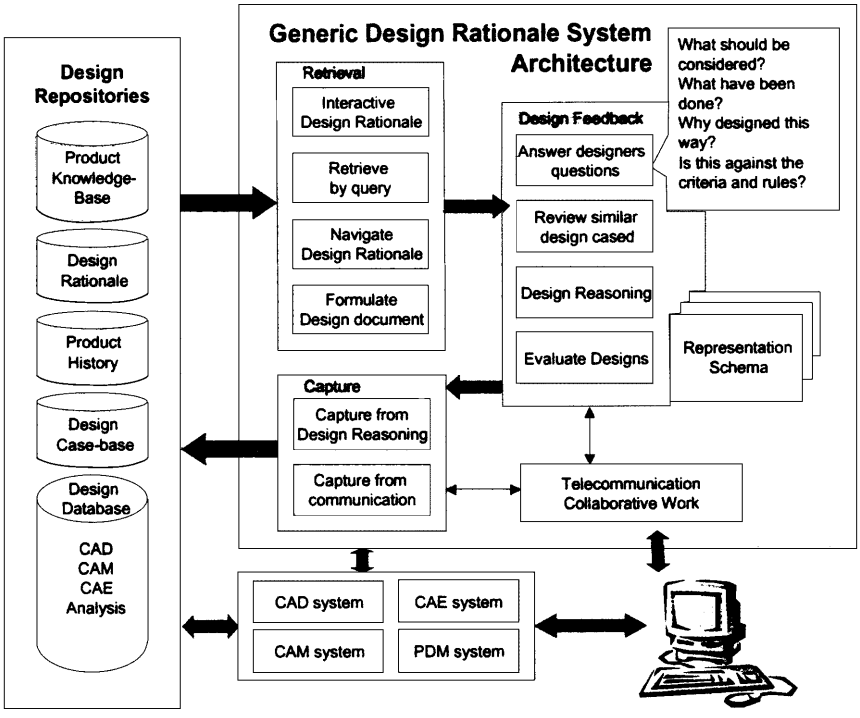

Fig. 2. The general architecture of a design rationale system.

multimedia nodes could be included as part of the design schema [26,18].

Figure 3 shows an example of the process-based approach to design rationale. In this example, a material handling system is being planned for a manufacturing plant in design. At this initial design stage, there is only a functional specification, with no restrictions on the step-by-step evolution of the design. With the process-based approach to design rationale, the step-by-step reasoning is captured and represented according to different tasks in the functional specification. An IBIS schema is used here for design rationale representation.

The challenge with this approach is to convert the captured information into *structured design rationale* – to create links among nodes – to make the information accessible. A number of approaches have been used to create links among nodes: PHIDIAS [18] links the nodes by authoring and indexing; REMAP/MM [26] supports hyper-links among design deliberation records and multimedia objects. This conversion process presents a large overhead to the design rationale maintainer (a smart computer or a patient human being). Another method to organize the recorded information is incremental formalisation, described in Shipman and McCall [35].

Once the nodes are linked, design rationale reuse becomes a matter of retrieval of the knowledge. The rationale could be viewed by traversing from one node to another by way of links, or be retrieved in response to queries. There are a number of techniques to deal with this; more detailed information on retrieval is given in Section 6.

The process-oriented approach to design rationale helps designers by providing descriptive history information, to answer questions such as who, why, what and when. Currently, it is not easy to translate this into representations that can be understood and processed by computers, so this approach provides support to the design process only when designers access and understand it [21].

3.2. Feature-Oriented Approaches

A *feature-oriented* approach starts from the design space of an artifact, where the rules and knowledge in the specific domain must be considered in design decision making. A design decision not supported by the rules of knowledge needs to be confirmed, so it will cause a revision of the rules and knowledge in the domain [24].

A feature-oriented system is usually developed in a *task specific context* using an *empirical study*. Some models are extended from generic *Design Decision Support* (DDS) systems by adding primitives to explicitly represent design process. Gener-

Table 1. Table of prototype design rationale systems.

System Name Acronym	Knowledge Representation	Knowledge Capture	Knowledge Retrieval	Approach	Design Domain	Year
IBIS [17]	Issue-based	UI	Navigate	PO	Generic	1970
PHI [20]	Extending IBIS	UI	Navigate	PO	Generic	1987
QOC [5]	Design Space Analysis	UI	Navigate	PO	Generic	1990
DRL [1]	Representing elements of decision making	UI	Navigate	PO	Generic	1991
CRACK [16]	N/A	Auto	Trigger	FO	Kitchen	1989
VIEWPOINTS [16]	IBIS	N/A	Navigate	FO	Kitchen	1989
JANUS [21]	PHI	Auto	Hybrid	FO	Kitchen	1989
IBIS-style browser [22]	IBIS	Auto	Navigate	PO	Generic	1991
COMET [23]	LOOM	UI	Navigate	FO	Sensor-based tracker software	1992
ADD [24]	Argumentation & Model-based	UI	Trigger	FO	HVAC	1992
REMAP [25]	IBIS	UI	Query	PO	Generic	1992
REMAP/MM [26]	IBIS	Auto	Query	PO	Generic	1995
ADD+ [27]	Rhetorical Structure	UI	Query	PO	HVAC	1997
HOS [18]	PHI	Auto	Trigger	PO	Generic	1997
PHIDIAS [18]	PHI	Auto	Trigger	FO PO	2D, 3D	1997
KBDS-IBIS [14,15]	IBIS	UI	Query & Navigate	FO PO	Chemical Plant	1997
DRIVE [28]	PDN	UI	Query	FO	Building	1997
DRARS [29]	QOC	N/A	N/A	FO	Building	1995
KRITIK [30,9]	SBF	UI	Query	FO	Mechanical	1993
IDIS [31]	IBIS	UI	Navigate	FO	Chemical Plant	1998
RCF [32]	N/A	Auto	N/A	PO	N/A	1999

Capture Method: User-Intervention (UI) or Automatic (Auto) *Representation Method*: Feature-Oriented (FO) or Process-Oriented (PO) *Retrieval Method*: Navigate, Query, Trigger or Hybrid.

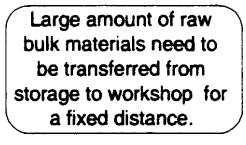

Fig. 3. An example of process-based approach: the planning of a material handling system for a manufacturing plant.

ally, these kinds of systems contain domain knowledge-bases, which can be used to support automated reasoning. CRACK is a typical feature-oriented system [16]; its detailed description can be found in the following sections.

In a feature-oriented design rationale system, existing knowledge-bases usually support the generation of design rationale, so representations of design rationale are usually more formal than in process-oriented systems. In some systems, the design rationale is represented with links to the existing knowledge-base. The retrieval and reuse of design rationale is very natural in the design process of later artifacts. An example model is GTMD [36], which is used to structure the acquisition process within DESIRE, a formal framework for compositional modelling.

With this approach, design rationale systems are usually included in the design systems. This helps designers by storing design rationale in a formal format which can be processed automatically, so it can provide active support to design or redesign processes such as the evaluation of design decisions and conflict resolution [28].

While the feature-oriented design rationale approach provides active support to design activities, it has the limitation that only part of design rationale (i.e. how the artifact designed satisfies the requirements) can be handled: other parts (i.e. option-exploration, trade-off, who, when, why, etc.) cannot be handled with this approach [30].

Combinations of both of these approaches have been proposed to help overcome their individual limitations. Systems with such a hybrid approach not only provide logical structure for design rationale, but also record the history of the design process. KBDS-IBIS is such an example [14].

3.3. Integration with Other Design Support Systems

A design rationale system is usually positioned as a technology to augment Computer Aided Design (CAD) and other engineering activities. In this way, it is not intended as a stand-alone system. Its smooth integration with other design support systems affects its effectiveness and efficiency. Figure 2 gives a general view of the integration of design rationale systems with other systems.

There is a trend towards tight integration of design rationale tools with other design representations [2], with the design rationale system being treated as an extension of the design system. Design rationale systems integrated with commercial CAD system have been extensively reported [14,31,32,27,24,37, 26,18]. In some of these systems, the integration was achieved by the combination of design rationale system with knowledge-based design decision support systems [31,27,37,26]. To integrate design rationale tools with design systems, there should be an integration of representation schemas. This improves the design system, makes the implementation of design rationale easier [9], and makes the capture and retrieval of design rationale driven by implementation concerns. The integration of design rationale with Computer Supported Cooperative Work (CSCW) and telecommunications was reported in PHIDIAS [26,18].

Sections 4–6 give a more detailed discussion on representation schema, capture and retrieval.

4. Representation Schemas

Due to the increasing complexity of computer systems and demand for higher reliability, there is a growing interest in developing design systems to help designers record and re-use design rationale. Especially for the design and maintenance of large systems (such as software systems), the capture and re-use of design rationale information system can be essential over the life-span of the product. Designing an artifact involves considering many alternatives – where one must address, evaluate, and ultimately accept or reject each alternative. A design rationale system needs to record the analysis of the various alternatives so that designers can easily make their decision; after designing, the rationale for a design should be kept for future use. How to organise this enormous amount of diverse material, and build it into a usable structure, is a critical issue [2]. A good *representation schema* is vital to enabling effective design and reuse. Much attention has been focused on developing methods, notations and tools for recording rationales, the space or history of arguments surrounding the actual decision made as development progress is represented. An overview of some of the most prominent techniques is given in the following sections.

4.1. The Issue-Based Information System (IBIS)

The Issue-Based Information System (IBIS) uses an *issue-based* approach that has been used in architectural design, city planning and public-policy discussion. The key issues of IBIS are usually articulated as questions, with each *issue* followed by one or more *positions* that respond to the issue. Each *position* can potentially *resolve* or be *rejected* from the issue. *Arguments* either *support* or *object* to a position. Figure 4 shows the relationship among three elements in IBIS. An IBIS-style browser [22] is the implementation of a merged issue-based and *truth-maintenance* [38] dependency structure. Such browsers use a graphical shorthand to represent node types and link types in a graph-based structure. Issues, positions and arguments are the main components captured in the graph. One characteristic of such a system is that it can provide immediate feedback to designers by indicating the belief status of various issues (via colour or other notations on the nodes).

The Representation and Maintenance of Process knowledge (REMAP) system [25] uses the *Telos* language to represent knowledge, and is also based on the Issue Based Information Systems (IBIS) method. The history about design decisions in the development life-cycle is called *process knowledge*. Much of this knowledge, involving the deliberation on alternative requirements and design decision, is

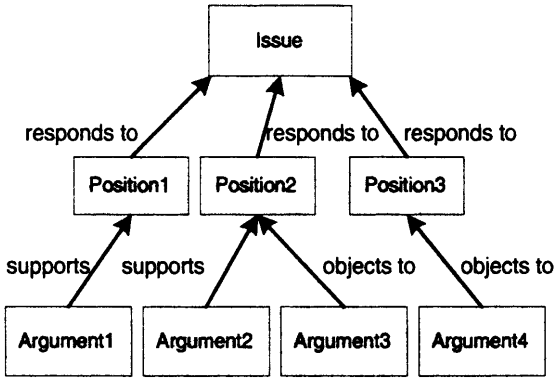

Fig. 4. The relationship of issue, position and argument.

lost in the course of designing and changing such systems. REMAP records the argumentation related to deliberations and solves the process knowledge loss problem. As the deliberation proceeds among designers during the design process. REMAP lists various issues, such as a problem, a concern or a question that requires discussion for the problem solving to progress; positions (alternatives) for solving each issue are responded using a process model window, each of these positions has different supporting arguments, each of which in turn is supported by assumptions. Issues are resolved by making a decision to select a position, thereby leading to a constraint that needs to be satisfied by design objects. Therefore, process knowledge is related to the objects that are created during the requirements engineering process.

Researchers have recognised that an explicit record of design rationale is an important requirement for effective design support. As computer systems supporting all aspects of the design process evolved, many have pursued integrating the record of rationale with the history of the evolution of the design artifact [39,40]. An integration of these two disjoint data models could bring many benefits for design, such as improved documentation of the design process and the verification of the design methodology and the resulting design artifact.

The Knowledge Based Design System-IBIS (KBDS-IBIS) [14] is the result of integrating a representation of design rationale (in the form of IBIS networks) to the design alternative history maintained by a design support system (KBDS). KBDS-IBIS has been used in chemical process plant design, and put forward some extensions to the structure of standard IBIS networks. KBDS-IBIS introduced three new classes of object – *artifacts, steps and tests*. These integrate the representation of *argumentation* with the *design process history*. KBDS-IBIS also can automatically generate two types of reports: *step reports*, which describe the design rationale for the evaluation of one design alternative relative to another; *issues reports*, describing the deliberation leading to a particular design. These record in a prescriptive fashion – allowing the design team to identify which parts must be re-designed in the light of a change in the internal assumptions, constraints or specification, or a change in any external factors affecting the design. In an integrated and prescriptive form, KBDS-IBIS explicitly maintains the history of the design goals, decisions, justifications and assumptions coupled with the evolving description of a chemical process plant during its conceptual design.

Chung [31] described an Integrated Design Information System (IDIS) that supports the design of chemical plants. This system also places particular emphasis on supporting the design process so that the recording of design rationale may be done easily. A commercial system from Enviros Software Solutions, DRAMA (Design RAtionale MAnagement), was based on the ideas of the KBDS prototype from the University of Edinburgh. It uses a graphical user interface and an object-oriented database to store and structure reasoning relating to design (http://www.enviros.com/drama/).

4.2. Procedural Hierarchy of Issues (PHI)

The Procedural Hierarchy of Issues (PHI) [20] extends IBIS by broadening the scope of the concept '*issue*' and by altering the structure that relates issues, answers and arguments. First, it simplifies relations among issues by using the 'serve' relationship only. Secondly it provides two methods to deal with design issues: deliberation and decomposition (i.e. to give answers to the issue or to break down the issue into a variety of subissues which in turn could be deliberated or decomposed). The prime issue is by definition the whole project. PHI avoids raising irrelevant and trivial issues [20]. Another advantage of PHI is that the issues, subissues, answers and arguments can be presented in the format of outlines, as shown in Fig. 5. Compared with IBIS, PHI provides dependency relationships between issue resolutions and considers the pros and cons of alternative answers [41]. In addition, it more completely and accurately models the task structure of the design process and the information useful for tasks [21].

VIEWPOINTS [16] is a hypertext system for argumentative kitchen design based on the PHI design methodology. It is a a tool which supports the argumentative approach within the design process. The elements of VIEWPOINTS are *issues, answers, arguments* and *graphics*. It provides a graphical interface to facilitate its use, enabling designers to do extensive browsing and make decisions.

JANUS [21] is the integration of the CRACK and VIEWPOINTS systems. CRACK [16], another computer-supported design rationale system, is a knowledge-based critic which has information about how kitchen appliances can be assembled into a functional kitchen. JANUS allows architectural and interior designers to graphically construct artifacts by direct manipulation, and at the same time receive

```
ISSUE:
What kind of conveyers should be used in material handling?
  SUBISSUE:
  1. What kind of conveyers should be used in bulk material handling?
  …
  2. What kind of conveyers should be used in unit material handling?
    ANSWERS:
      1. Gravity conveyers
        SUBANSWERS:
        1. chutes, skate wheel conveyers
        2. roller conveyers
          ARGUMENTS:
          1. Its advantage is low cost, relatively low maintenance, and negli-
          gible breakdown rate.
          2. Its requirement is the ability to provide the necessary gradient in
          the system configureation.
        χ…
        2. Powered conveyers
        …
        3. Chain-driven conveyor
        …
        4. Power-and-free conveyers
        …
```

Fig. 5. An example of PHI: select conveyers for material handling.

information useful to what they are doing from hypertext activated by knowledge-based agents. From JANUS and its predecessors CRACK and VIEWPOINTS, we can see that integrated support for construction and argumentation is necessary for full support of design.

PHIDIAS [18] is a hypermedia system which is based on PHI; like other rationale representation schemes, it uses a graph-based node-link structure. PHIDIAS represents all of its knowledge, including semi-formal rationale as well as formal representations of physical objects, facts and rules, in this graph-based format. In PHIDIAS's graphs, both nodes and links are first-class objects with prototype-based inheritance. Nodes can vary in size from a single letter to a multi-page document, with most rationale nodes corresponding to individual words, sentences, or paragraphs. The Hyper-Object Substrate (HOS) [18] is another hypermedia prototype. It includes a generic view containing an argumentation structure, which uses a piece of the design and nearby discussion as an example in this structure.

4.3. Design Space Analysis

Design space analysis places an artifact in the space of possibilities and seeks to explain why the particular artifact was chosen from these possibilities. The Question, Option and Criteria (QOC) [5] is one semi-formal notation of *design space analysis*, which focuses on the three basic concepts indicated in

its name. The QOC representation emphasises the systematic development of a space of design options structured by questions, which is different from the IBIS-derived systems and PHI, whose purpose is to capture the history of design deliberation. QOC represents the design space using three components: *questions* identify key issues for structuring the space of alternatives; *options* provide possible answers to the *questions; criteria* are the bases for evaluating and choosing from among the *options*.

Figure 6 presents an example of design space analysis for displaying a scroll bar. The rationale representation in QOC is created along with the descriptive representation (specification) or the artifact itself (prototype). In aspects of innovation and reuse, QOC has many advantages. It is relatively easy for a maintainer to create a QOC to 'reverse engineer' a part of a system and preserve it for future use. QOC records the exploratory activity of the design team, such as what alternatives were considered and what choices were made and why. QOC's graphical argumentation notation lets team users actively manage QOC recording during work, and supports the transition from expression to documentation, from informal, incomplete, private rationale to more formal, complete, and publicly intelligible rationale [42].

The Design Rationale Authoring and Retrieval System (DRARS) [29] is a system that uses a variation of QOC [29]. Views, goals, alternatives, claims, questions, answers and versions are the DRARS system's objects. The human user is responsible for giving descriptive and useful names to these objects.

Another way to record design rationale is by describing how an artifact serves or satisfies expected functionalities, using languages such as Decision Rationale Language (DRL) [1] or Function Representation (FR) [30]. DRL is an expressive language which represents the space around decisions, which can be maintained independently or integrated with traditional design representation. DRL was implemented in a system called SIBYL [43]. and is being used to explore various kinds of computational service over DRL structures.

Also related to QOC, [44] describes a language called TED used to represent the reasoning behind the implementation of particular features.

4.4. Functional Representations

Functional Representation (FR) [30], a representational scheme, describes how the device works (or is intended to work). In the functional representation scheme, design rationale is used as an account of how the designed artifact serves or satisfies expected functionality. One can use FR to capture the *causal* components of design rationale. FR takes a top-down approach to represent a device: the overall function is described first, and the behaviour of each component is described in the context of this function. FR encodes the designer's account of the causal processes in the device that culminate in achieving its functions. Tasks that design rationale should be able to support are: control of distributed design activity; reassessment of device functions; generation of diagnostic knowledge; simulation and design verification; redesign; and case-based design. FR provides a partial rationale for choices made about components and their configuration; its limitation is that FR only captures the causal knowledge about device operation.

The Structure, Behaviour and Function (SBF) model [37] is an approach for designing devices which explicitly represents the *functions* of the

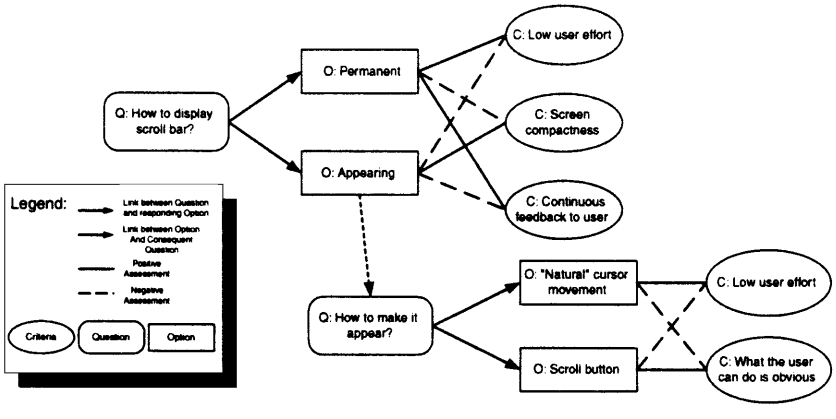

Fig. 6. A QOC representation of the design space for displaying scroll bar [5].

device (the problem), the *structure* of the device (the solution) and the internal causal *behaviours* of the device. Function can be defined as *what* (an object) *does*, behaviour as *how* (it) *does what* (it) *does*, and structure as *what* (an object) is. For example, a clock's function is to indicate to an observer the time, its behaviour is that its hands go around with a periodicity matching the elapsing of time, and its structure is its composition of metal, plastic, glass, and so forth [45]. The structure of a device in the SBF language is represented as a schema that specifies the input behavioural state of the device. The SBF model of a device also specifies the internal causal behaviours that compose the functions of device substructures into the functions of the device as a whole. The functional models represent *design cases* from which adaptation spaces are generated for solving a new design problems. The designed products are represented using qualitative models with the structure and behaviour of the artifact.

SBF models provide a powerful solution for adaptation problems and for performing case-based and variational design [46], in which old design cases are adapted to address new design challenges. KRITIK [37,47] is a system which uses a functional representation scheme called Environmentally-bound Structure-Behaviour-Function (EBSF) to represent and organise knowledge of the functioning of a device, including the role of its environmental interactions.

Rosenman [6] clarified the concepts of *purpose, function, behaviour and structure*. Purpose is the action or fact of intending or meaning to do something; function is 'the action of performing', 'the mode of action by which it fulfills its purpose': behaviour is defined as how something acts in response to its environment; and structure is the organisation of the constituents of the object. The relationship between them is: structure *exhibits* behaviour *effects* function *enables* purpose: or, purpose *enabled-by* function *achieved-by* behaviour *exhibited-by* structure. The process of interpreting required behaviour to arrive at a given structure is the process of analysis, the interpretation of structure to determine behaviour and function is the process of synthesis. These processes are causative processes within the physical environment. The process of interpreting function for purpose is the process of realisation (of possible utility), whereas the process of interpreting required purposes as desired functions is a process of problem formulation. These latter two processes provide the communication between the human value system and the physical environment, and teleological reasoning comes into play. Purpose, function, behaviour and structure are generally decomposed down to sub-functions, sub-behaviours and sub-structures, so the problem is formulated by the relationship among groups of related functions, groups of related behaviours and groups of related structure (see Fig. 7). Such a representation can be easily mapped to a relational database, with each predicate being mapped to a table in the Relational Database System (RDBMS), or into an object-oriented database, which has better capabilities of handling class-subclass relationships.

4.5. Other Representation Schema

In each different engineering domain, from mechanical design to software design, the design process has its own unique features. According to the specific requirements, various support systems have been developed. One major challenge for software engineers is judging how a change in one software module effects (and is affected by) the rest of the design. Mark [23] introduced the Comet system, which uses explicit representation and reasoning with commitments to aid the software engineering and development process. The design knowledge managed by Comet is in the form of *module descriptions*: structure and behaviour specifications of modules interrelated by commitment constraints. The underlying representation is LOOM [48], a language and environment for knowledge representation and reasoning. LOOM maintains a taxonomy of module descriptions based on the defined inter-relationship of their constituent terms, and can automatically determine modules' relationships. Developers and software engineers can examine the commitments

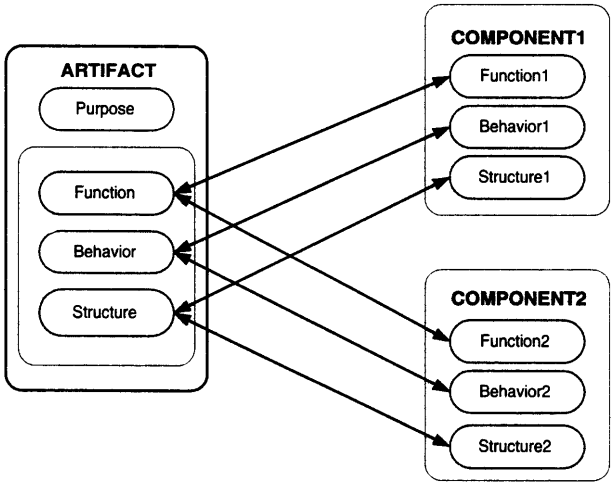

Fig. 7. Whole-component decomposition [6].

that must be met in order to include an existing module, and can explore how commitments change when modules are modified. Comet has been applied to the domain of sensor-based tracker software [23].

Augmenting Design Documentation (ADD) [24] is an integrated computational model for assisting designers in documenting projects at design time. The model was developed based on observations of designers developing Heating. Ventilation and Air Conditioning (HVAC) systems and on observations of documentation users accessing design documents. The ADD [24] model represents design rationale as a combination of argumentation-based and model-based rationale, and is good at both rationale acquisition and explanation. It works by documenting the complete design decision path associated with the artifact, as well as the rationale behind each decision presented by the user. This solution path represents the designers' strategy, in which each node is a sequentially linked decision. Users can explore the design rationale in several ways: through the history tree, the dependency tree, annotations, and (most importantly) by asking direct questions. However, the ADD system only provides a one-paragraph answer, without references to the relevant data in the knowledge base.

More recently, Garcia proposed a system, called ADD+ [27], which includes *rhetorical structures* in *active documents*. ADD+ uses the same basic model as ADD, but improves the system's interactions with the user. In ADD+, the wealth of knowledge kept in ADD's knowledge base is organised into high-level *Rhetorical Structure Theory* (RST) [49] schema, and mapped onto input and output screen configurations that gear the interaction between systems and users. Compared with ADD, ADD+ has an explicit communicative model to convey messages that reinforce the model's usability.

Garza [28] discussed a design rationale system, a path-finder computer program called Design Rationale for the Information phase of Value Engineering (DRIVE), used as part of a much larger Computer-Aided Value Engineering (CAVE) system. It consists of two modules: a domain-dependent Knowledge Representation Module (KRM), which contains objects and attributes representing building design information, and a domain-independent Rationale Storage Module (RSM), which contains all the design decisions made about the different performance parameters of various design objects in the KRM. The *depends-on* and *has-relationship* semantic net [50] links of of RSM generates the Parameter Dependency Network, which can determine how the designers arrived at a particular design decision. It also can determine how one object-parameter affects other object-parameters and further affect other object-parameters.

Figure 8 shows an example of the PDN. The *Room-Function* object-parameter pair affects the *Room-Fire Resistance Rating* object-parameter pair, which in turn affects the *Door-Fire Resistance Rating*, *Wall-Fire Resistance Rating* and *HVAC Equipment-Fire Resistance Rating* object-parameter pairs. It uses Kappa-PC (an object-oriented expert system prototyping environment and AutoCAD (a graphical computer-aided design application). Compositional modelling is one other approach that has been tried [51,36].

5. Design Rationale Capture

How design rationale is captured is a critical element in the development of a design rationale system: capture too little information (or the wrong information), and it will be impossible to create a representation of rationale; capture information too intrusively and designers will not use the system. The primary requirement of the design knowledge capture process is that it captures design descriptions in a form that supports the communication and reuse of design knowledge. In particular, these capture tools should operate on representations that are useful for constructing design rationale explanations [4].

From the designer's point of view, we can divide design rationale capture methods into two categories: those that require *user-intervention*, in which the designer must manually record the design information during the design process, and those that are *automatic*, in which the capture is performed automatically by the design rationale system.

5.1. User-Intervention-Based Capture

User-intervention-based rationale capture has often been approached by the *documentation* method. Documentation is intended to record the history of design activities. More specifically, it records what decisions designers made, when they were made, who made them and why [18].

The form of documentation is a *report*, which is written by individual designers. The creation of documentation tends to occur after the decision processes. Documentation therefore merely records decisions, without influencing the designer's thinking processes leading to the decisions. Documentation has some very useful functions [18]: it enables

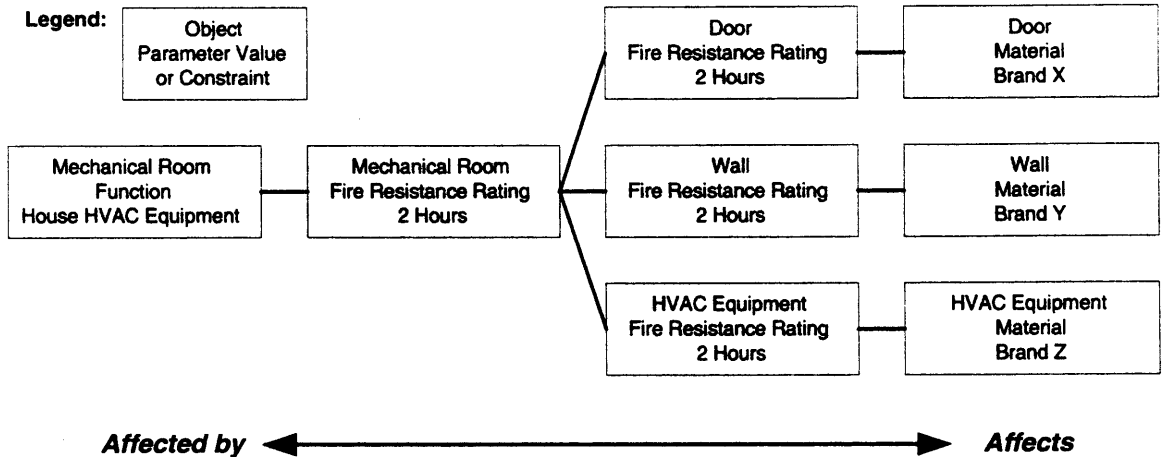

Fig. 8. Parameter dependency network example [28].

people outside the project group to understand, or supervise, the design process; and it can be used to identify the intellectual property generated in a project for the purpose of obtaining or defending patents.

Conklin [3] claimed that the documentation approach to rationale capture has the following drawbacks:

- documentation requires manually writing down a design rationale, which is not executable or even implementable;
- as documentation, a set of decisions is inherently unstable, especially if it is being written during an exploratory process;
- in order to be updated, the design rationale document can grow into a huge bundle. In this case, there are many people involved in the development process, and issues may be repeatedly recorded, which may result in inconsistent information in the design rationale document;
- design is often marked by breakthroughs of understanding, after which some previous decisions and assumptions may become incorrect or irrelevant after conceptual restructuring. When this happens, the design rationale document must be rewritten.

Most of the design rationale systems are using mechanisms that let designers provide their own expressions about design decisions and reasoning during the design process. The REMAP model [25] supports the capture of design rationale knowledge by providing a mechanism for the design team to conduct deliberations using the primitives (issues, positions and arguments) in the model. As introduced in Section 4.1, all these primitives are defined manually by the designers during the design process.

The Comet system [23] uses explicit representation and reasoning with commitments to aid software design. In terms of rationale capture, it requires the user to introduce new module descriptions as specialisations of existing module descriptions, which limits the design flexibility. However, a crucial area in which Comet is particularly useful is through its Design Memory window, which allows developers to trace the compatibility of the candidate modules with the current design. This notion is very important to the computational tractibility of the system's reasoning process.

In Garcia's [24] ADD system for HVAC, rationale is captured by using the computer as a 'designer's apprentice'. ADD contains a predefined set of relationship between the various design parameters. These relationships allow the system to expect certain values for the design parameters. If the designer proposes a value different from the expected value, the computer asks the designer for justification regarding the differences. The justification is input by the designer and recorded in the system's database.

DRIVE [28], compared with ADD, has a fixed and predefined set of relationships between its various model parameters. DRIVE builds relationships between its various model parameters in real-time, i.e. as the design develops. It helps designers express the rationale behind their design decisions in a computer-interpretable format. It also assists value engineers in formulating suitable design alternatives by presenting rationale about an existing design. However, the disadvantage of DRIVE is that it starts out with a totally empty rationale database and relies on the designers to create relationships among its various object-parameters. In contrast, ADD starts with a predefined but modifiable set of rationale information. ADD captures additional design ration-

ale when an expected value provided by the original rationale database differs from the current state of design. In other words, DRIVE needs user-intervention even though it is more flexible in terms of defining parameter dependency relationships.

KBDS-IBIS [14] introduced two new ideas. The first was to enable the designer to record a variety of complementary types of documents within the process design history, addressing the goal of improving the design information recorded. Design rationale is recorded in KBDS using an extension of IBIS structures. Secondly to extend the range of information recorded, KBDS allows the designer to annotate or associate a number of documents of different types to unit operations in a process flowsheet or to notes within an IBIS structure. The quality of design support depends on the quality and quantity of the design history records. It is necessary to have a full and accurate description of the design artifact on which to base the argumentation for design decisions.

Even though some design rationale systems need user-intervention to record design information, a system must naturally support the design process or engineers will find the use of the system a distraction. Chung's [31] IDIS has three main components for supporting natural capture of design rationale during the design process: a *viewpoint system*, an *issue-based system*, and a *rule-based system*. IDIS provides an integrated framework for recording three different aspects of design rationale: *exploration of design alternatives, reasons for design decisions* and *design constraints*. Recording the possible solutions and the argumentation explicitly helps other designers avoid considering the same unfruitful areas in the future, and can be useful in checking designs. By recording the design constraints as decisions are made, if changes are made to the design, the system can help designers check the design for violations of these constraints.

5.2. Automatic Rationale Capture

The automatic capture of design rationale assumes there is a method to capture the communication among the designers and design teams. The communication records can then be used to extract design rationale and decisions as they evolve during the design process [18]. Usually, communication employs Computer-Supported Co-operative Work Tools (CSCW) [52] or meeting technologies. This includes, for example, telephone, tape recorders, video camera, shared applications, or e-mail to capture oral discussions as well as writings and drawings exchanged between designers. The designers need not do anything more than pursue their usual design activities. When communications and meetings are archived digitally, design activities can be processed and design rationale determined.

One drawback is that what is recorded during communication and collaboration is likely to be free form, full of disorder and digressions [18]. Raw communication lacks structure; by using this approach to capture design information, the knowledge retrieval process may be ineffective as a result. While difficult to extract meaning from, the capture of raw CSCW sessions and project meetings is the most convenient approach for the raw archiving of design rationale.

The HOS system [18] provides an environment supporting computer network design with the combination of natural communication and design in an argumentation structure. HOS includes facilities for importing email and USENET/Internet news files – thus, the network designers can continue their existing practice of using e-mail to inform one another on the progress of tasks, and later include this information in their HOS design space. This design information capture process is done automatically by the HOS.

PHIDIAS's [18] integral, graph-based architecture facilitates relating the semi-formal knowledge contained in design rationale with more formal system knowledge. It implements this capture process in two steps. The first is by representing all of its knowledge, including semi-formal rationale as well as formal representations of physical objects, facts, and rules, in a common graph-based format. The second step is simply using hyper-links to interconnect knowledge items, regardless of their level of formality.

Ramesh [26] suggested that an ideal design rationale system should not only facilitate easy and non-intrusive capture, but also provide automated reasoning with related knowledge. Ramesh implemented a multimedia extension based on the REMAP model, called REMAP/MM, which is a prototype Decision Support System (DSS) environment that supports capture by providing a graphical interface for design teams to conduct their deliberations. It also supports hyperlinks among design deliberation records and multimedia objects. REMAP/MM explicitly identifies the dependencies among design rationale; when design is reviewed, dependency information can be used to identify relevant components of the process.

Fischer's [16] CRACK kitchen design environment consists of two components: a domain-oriented

construction kit for creating a kitchen floor plan layout and a knowledge-based critic for evaluation. The construction kit in CRACK is provided to give the designer the feeling of directly generating the design without the computer's being 'in the way'. The important abstract operations and objects in the kitchen design domain have already been built into the CRACK construction kit. Therefore, the design rationale used during the design is captured automatically by the system.

JANUS's [21] architectural design system integrates a CAD-like editor with a rule-based design critic and an argumentation-structured hypertext documentation environment; it is an integration of the CRACK and VIEWPOINTS systems. The JANUS system has demonstrated that hypertext can be used in conjunction with knowledge-based design environment to ease design knowledge capture.

An experimental system, the Rationale Construction Framework (RCF) [32] was designed to acquire rationale information from the detailed design process without disrupting a designer's normal activities. The underlying approach involves monitoring designer interactions with a commercial CAD tool to produce a rich process history. This history is subsequently structured and interpreted relative to a background theory of design metaphors that enable explanation of certain aspects of the design process.

Generic task model of design. DEsign and Specification of Interaction REasoning components (DESIRE) is a formal framework for multi-agent systems [53,51]. It explicitly models five types of knowledge: the task composition, the knowledge structure involved, information exchange between tasks, sequencing of (sub)tasks and goals, and the roles of the participating agents. It has been used for management of conflicts in design [53], (parametric) design of elevators [51]. Generic Task Model of Design (GTMD) [36] is one of the generic models available within DESIRE. GTMD is used to structure the acquisition process, clearly distinguishing the design tasks into three subtasks: reasoning about requirements and preferences (RQS); reasoning about the Design Object Description (DOD); and reasoning about (coordination of) the overall design process. Each of these three are composed of four subtasks: modification, update-of-modification history, deductive refinement, and update of current description. Knowledge used to generate design rationale is represented in the components of the GTMD. During the design process, the design rationale is generated and stored in the respective history components. Different types of design rationale are distinguished according to the functional role they play in the design process. GTMD has been used to structure the modelling process of an example aircraft design task [36] and an elevator design task [51].

Interactive Acquisition of Justifications. Justifications explain why something should be believed or done. Gruber [54] proposed an approach for acquiring justifications by transforming why-questions into what-questions. It changes the open-ended task of explaining why into the constrained task of selecting what is relevant. The author framed the problem of capturing design rationale as a knowledge acquisition problem, where the task is to elicit, from domain specialists (designers), knowledge that enables a program to generate explanations of how the designed artifact is intended to achieve its function. The ASK knowledge acquisition system was proposed. It first elicits an example specifying a situation and a choice; the user provides the example by running the interactive knowledge system until it comes to an interesting choice among actions. The user interrupts the knowledge system, and indicates which action the user would have taken in the situation and an example of an action that would not have been appropriate (the negative example). It then elicits justifications for the strategic decision, that is, reasons for choosing one action over another. The interface presents features of the current situation, and users select from among these features a set of relevant features for choosing the positive example. ASK computes the values of the specified features for the current state and the positive and negative actions, and presents these object-feature-value tuples as English sentences that are intended to explain why the chosen action is appropriate.

Capture of device behaviours. Stahovich [55,56] described a program called SketchIT that captures the behaviour of the device in a single sketch, represents them in *qualitative configuration space* (qc-space), and then synthesises them into multiple families of new designs. The input of SketchIT is a stylised sketch of a device and a state transition diagram describing the device's desired overall behaviour. The latter provides guidance in identifying what behaviour the individual parts of the device should provide. The overall behaviour of the device is achieved through a sequence of interactions between pairs of engagement faces, hence the behaviour of the device is captured and represented first by configuration space curves (cs-curves), which are formed by touched points of engaged surfaces in

configuration space (c-space). The axes of a c-space are the position parameters of the bodies; the dimension of the c-space for a set of bodies is the number of degrees of freedom of the set. The numerical cs-curve is further abstracted into a qualitative c-space, which represents cs-curves by their qualitative slopes and the locations of the curves relative to one another. In the synthesis process, the program turns each of the working qc-spaces into multiple families of new designs represented by a BEP-Model (Behaviour Ensuring Parametric Model). The synthesis process is comprised of two steps: selecting a motion type for each part and selecting a geometry for each pair of engagement faces. There are a variety of selections of motion type and geometry which are different from those of the original sketch, but have the same behaviour described by the working qc-spaces. This variety leads to a family of new designs which have the same function as that of the original sketch.

6. Design Rationale Retrieval

The reuse of design rationale is realised by its successful retrieval. In this context, design rationale research shares much in common with research in case-based reasoning and case-based/variational design. Case-based reasoning has been an active research area for the past 15 years [57–60]. This work represents a foundation of structures, algorithms and techniques for reasoning about and adapting archived knowledge.

Design rationale systems operate in a similar manner, retrieving past experience relevant to solving a new problem. Which cases are retrieved depends upon both the current objectives and the representation schema of the design rationale. There are different potential scenarios for retrieving design rationale: (1) to view similar design cases at the initial conceptual phase of design; (2) to retrieve criteria, rules and options to help make design decisions during the design process; or (3) to produce documents after a design process. Depending on the scenario, there are different strategies to retrieving related design rationale and avoiding unwanted material [14,27].

Lee [12] classified design rationale systems as *user-initiative* or *system-initiative*, depending on whether the user or the system initiates access. We consider the retrieval strategies of design rationale systems by considering their support of three basic retrieval approaches: navigating by designers, retrieval by queries, and automatic triggering during the design process.

6.1. Navigating Archived Design Rationale

Design rationale navigation involves permitting designers to explore design rationale by traversing from one node to another by existing links. Systems may provide facilities to help with such navigation [41]. For process-oriented approaches, this navigation often provides a backtracking of the design history. In a feature-oriented approach, with design knowledge being stored according to features of the designed artifact, navigation is done around the feature structure. For a complex artifact, a large quantity of information is recorded during the design process – looking up specific answers is often a difficult task.

The VIEWPOINTS [41] system uses PHI and extends IBIS by broadening the scope of the concepts of issues, answers, arguments and graphics, to aid designers in finding answers to specific problems. VIEWPOINTS creates an integrated multifaceted design environment with five components: a *construction kit*, an *argumentative hypertext system*, a *catalogue* with a collection of design examples, a *specification component* and a *simulation component* for exploring the design options. VIEWPOINTS is used as a look-up manual, where designers can find answers to specific problems and consider various arguments for and against them. For a complex problem, the discussion around it may be distributed in a wide range of archived activities. To navigate through all the related nodes and to make sense of it becomes a difficult task.

An IBIS-style browser [22] lets users browse design rationale as a map with nodes such as issues, positions and arguments, and links such as responds to, supports and objects. As a system combining issue-based and truth-maintenance approaches, it helps designers perform what-if analysis using colors to indicate the belief status of nodes.

ADD [24] proposed a read-only interface to allow users to navigate graphically through the decisions and reasonings connected to the designed artifact. ADD also allowed users to verify the knowledge required for justifying each decision, thus leading to retrieval of an intelligent design document. A controller contained domain knowledge for checking designers' decisions.

COMET [23], the software design support system for Computer-Aided Software Engineering, allows software developers to review and check any exist-

ing module descriptions in the Comet knowledge-base that are consistent with the descriptions created for a new module. There is a design memory window with a nodes-link diagram to help users with navigation.

In IDIS [31], listing and browsing facilities are provided by AutoCAD diagrams and the viewpoint linked to them. By clicking an item on an AutoCAD diagram, the specification of that item will be displayed, and users can find out all the issues and rules related to it by visiting the viewpoint and its parents and children. It also provides some supplemental facilities to support keeping track of design progress, and for reviewing a project when it is completed. It provides an option for listing unseen nodes in the issue base to answer the question 'what is new' in the design process, and an option for listing the outstanding issues and their corresponding deadlines to answer the question 'what are the outstanding issues'. Users can treat the issue system as a database of free text, so a keyword search could be used to find out 'what has been said about a certain topic'; it can display all the relevant issues that led a particular design alternative to answer 'how did we get here'. To answer the question 'what are the differences between two design alternatives', it can provide a summary of the viewpoints leading to the different design alternatives, process units that are in one design and not in another, and different parameter values associated with the same items in different designs.

6.2. Automatic Triggering

Several design rationale systems enable the capturing of design rationale by automatic triggers that detecting or monitor certain conditions according to the design context. This type of approach shares many similarities with the event-driven programming paradigm in the engineering of real-time and interactive software systems. Design conditions are monitored according to corresponding rules, criteria or constraints of design. The monitor is used to look over and check the design process, and compare the decisions made with the constraints, rules or criteria in a design rationale library or knowledge base; if differences are detected, the design rationale will be retrieved automatically.

The PHIDIAS [18] system uses issue-based indexing of design rationale as its hypertext-based retrieval strategy. The basic idea is to connect the design rationale with the design task. The connection is artifact-based indexing, which connects design rationale to the drawing of the artifact, critique-based indexing, which connects the critique to the design task, and operation-based indexing, which connects the design rationale to some specific operations on the designed artifact. Critics have been built according to the indexing scheme, which can be specified to be triggered automatically depending on design conditions or which may be requested to be executed by users.

CRACK [21] is a knowledge-based computer support system which can help designers solve constructive design problems in design activities. The critics in CRACK are intelligent support systems which detect and criticize partial solutions constructed by the designer based on knowledge of design principles. A critic can be triggered by state-driven condition-action rules, because it checks the knowledge -base to detect non-satisfying design decision [16].

ADD+ [27] acts as an apprentice and learns about the features that make a specific case different from the standard in the design process. The apprentice must be able to access the design knowledge so that a new design decision can be justified.

6.3. Query-Based Retrieval

To provide retrieval strategies according to designers' queries is more efficient than browsing the nodes of design rationale structures. The queries may be 'what-if' questions, which can be answered by exploring different options; or 'why' questions, which are answered by back-tracking in the network of nodes and links to find out the argumentation or reasoning behind a decision. Gruber [54] proposed a generative approach, in which design rationale explanations are generated in response to information requests from background knowledge and relevant information captured during design. The problem is how to provide a methodology of selecting and assembling knowledge from libraries of design rationale based on specifications of requirements [61].

REMAP/MM [26] uses a deductive query language to define various types of ad hoc queries, and provides a graphical interface for displaying queries and retrieving desired information. The queries may be recursive, which supports selective retrieval of process knowledge, allowing the design process to be replayed. Instead of replaying the entire process, dependency information can be used to identify relevant components of the process that need to be replayed. REMAP/MM made extensive use of multimedia and a combination of both informal and

formal representation to capture design rationale [26].

KRITIK [30,9] (Kritik in Sanskrit roughly means 'designer') is an early case-based design system. It uses a Structure-Behaviour-Function (SBF) model to explain how the structure of a device accomplishes its function, which is part of design rationale in showing why and how designers get the device to work as intended. The system searches the corresponding Causal Process Description (CPD), and follows state transitions to retrieve the desired knowledge.

DRIVE [28] is a rule-based system in which the captured design rationale has structure, and can be processed automatically by the system. It creates a rule base for all rationale hierarchy objects which is used to check the validity of the relationships every time a relevant design parameter changes, or to detect conflict and provide designers with various options for resolution.

6.4. Hybrid Retrieval Strategies

JANUS [21] uses the critics from CRACK to monitor the design process based on a knowledge base, and allows entering from a criticism point to the exact place in the hypertext network where the argumentation relevant to the current construction task lies. A Document Examiner provides functionality for on-line presentation and browsing of the issue base by users.

KBDS-IBIS [14] provides three ways in which design rationale can be used to the designer's advantage: *dependency-directed backtracking, automatic evaluation of position* and *automatic report generation*. All three need to review the design rationale according to certain requirements. To provide support in design requires prescriptive information to determine the validity of the argumentation stored within KBDS-IBIS, so it is more than just retrieval of what has been recorded.

7. Related Research Areas Supporting Design Rationale

Engineering design theory. Engineering design is the process of creating new products, processes, software and systems from an initial, incomplete and general set of goals, objectives, functional requirements and constraints, with the consideration of social and economic impacts pertaining to the use of product being designed. In general, engineering design involves the following four distinct aspects: *problem definition, conceptualisation, synthesis/analysis* and *detail design*.

Axiomatic design theory [62] is one of the successful methodologies used in the design field. It defines a design as the creation of the synthesised solutions that satisfy perceived needs through the mapping of the *Functional Requirements* (FRs) to the physical space, represented by *Design Parameters* (DPs) as shown in Fig. 9. Functional requirements are the minimum set of independent functions constructed to satisfy the design original needs. The design parameters are specified to satisfy the functional requirements.

Intelligent CAD. While CAD systems provide increasing levels of functionality to support the design process, they deal predominantly with detailed design. CAD systems are drafting and detailed modelling environments, performing best when users know what the design's physical form will be [63].

There have been a number of different approaches enabling intelligent CAD. *Case-based and Variational Design* environments enable previous design cases to be adapted to solve new problems [9,46,64]. Research in this area has built symbolic representations for indexing and retrieval of design cases.

Ullman [10] presents a detailed analysis of audio protocols used by five mechanical designers, observing that CAD tools should be designed to give cognitive support to the designer. He argues design rationale systems, combined with CAD systems, will provide a more complete toolbox to give better support for decision making at the initial design stage. A more recent project [65] examines communication among application programs in the Architecture, Engineering and Construction (AEC)

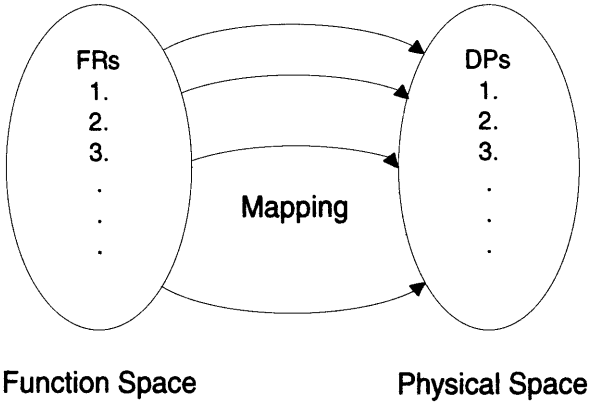

Fig. 9. Mapping process by axiomatic design.

domain. Internet-based project collaboration products are integrated with the AEC desktop, enabling a better understanding among team members in a design process. In this project, the commercial CAD tool is used to monitor designer interactions.

Klein [7] indicated that to achieve the benefits of using design rationale, representations must allow designers to express their design reasoning in a natural way. At the same time, these representations must be formal enough to support useful computational services and impose the minimum possible overhead on the design process. His Design Rationale Capturing System (DRCS) provides an integrated and generic framework for capturing rationale in team contexts. DRCS uses a vocabulary of assertions to capture design reasoning. The assertions consist of *entities* (e.g. modules, tasks, specifications and versions) and *claims* about these entities. The vocabulary of claims and entities that make up the DRCS rationale language fall into five categories: *synthesis* to capture the actions used to define artifacts and their plans; *evaluation* to capture not only design specifications but also how well they have been achieved; *intent* to capture when a designer takes an action; *versions* to capture how the designer creates and explores the space of design alternatives; and *argumentation* to capture the reasons for or against an action. DRCS was implemented in Common Lisp on Symbolics workstations.

Knowledge representation. The artificial intelligence community has produced a rich set of techniques for formal representation of knowledge [66–70,50,71]. Many of these techniques build on formal logics, and are often difficult to adapt to ill-defined and often informal engineering datatypes. This includes work on functional representation [30,72] and the creation of engineering ontologies [73–77]. There have been a number of successful efforts from the engineering community to apply formal AI representations to design and manufacturing problems [78–83,40,84].

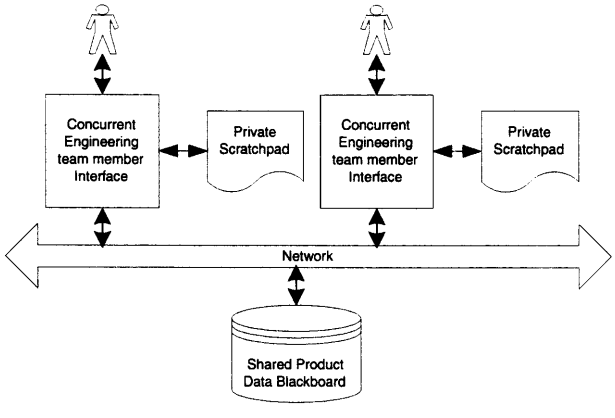

Fig. 10. DRCS architecture.

Knowledge-based design support. To model knowledge-bases and databases for intelligent design, design rationale must be captured and translated into these formalisms [85]. Many knowledge-based systems have been developed for design; several most relevant to design rationale systems include [15,14,31,86].

Gruber [87] analyses the role that a standard knowledge representation language can play in the sharing of Knowledge Bases (KB) among groups of people and programs that can make use of the knowledge and across research groups developing knowledge-based technology. Other work from Stanford on the DARPA MADE Program [88–92] repeatedly addressed knowledge sharing and knowledge representation as central issues in deploying collaborative engineering systems.

In a knowledge-based design system for conceptual design in chemical engineering [15,14], four inter-related networks are used to represent design process, and design rationale is represented using an extension of IBIS. IDIS (Integrated Design Information System) [31] is an issue-based system that supports the argumentation process during design; viewpoints are used to represent the design hierarchy and to support design exploration, and a rule-based system is used to represent design constraints. Knowledge can be represented in the expert system at different levels: the *knowledge-used* level is used 90% of the time, while the *knowledge level* is used to support the knowledge-used level [86].

For a comprehensive text on the subject of knowledge-based systems in design, readers are referred to the recent book by Sriram [93,94].

Machine learning. The research work in the area of machine learning has contributed many methods that have been applied to the acquisition of knowledge in design. Machine Learning in Design (MLinD) [95,96] examines the similarities between machine learning and design rationale capture. The main elements in machine learning are the input of knowledge, the output of knowledge, the transformation of knowledge, the learning triggers and the learning goals which are related to the record, the access, the construction, the capture and trigger of design rationale. The approaches of machine learning are basically analogical reasoning [97], induction-based

[98], knowledge compilation [99] and neural networks [98,100–102].

8. Other Applications of Design Rationale Systems

Design rationale systems enable a wide variety of other forms of knowledge reuse. For example, rationale environments can record not only the design decision, but also its justifications and the other design alternatives, trade-offs, and the argumentation that led to the decision. We present several other uses of design rationale environments.

Explanation, prediction and conflict management. Rationale systems can be used to provide justification for particular decisions [36]. In this way, people can understand which combinations of requirements and preferences designers have previously considered, which options and which partial designs were explored, and in which situation. They also can know the argumentation for the rejection and/or acceptance of options, choices and partial designs. Further, archives can help determine if specific design decisions were made previously in similar situations, using the results of past experience to help designers predict the probability of success for the new design problems [27]. Rationale systems have also been used to mitigate conflicts among design team members [36,15,18].

Design indexing, navigation and reuse. By associating formal representations of design rationale with their relevant design tasks, design experience can be retrieved. Such indexing is especially valuable when rationale needs to be rediscovered, reused or re-evaluated. The indexing can be issue-based, task-based, artifact-based, critique-based or operation-based [15,18,103,104].

Design rationale-based indexing can also be used to discover if parts of requirement descriptions, or object descriptions for past design cases can be reused for new design problems [36,105]. Some current research has focused on the role design rationale can play in redesign systems [105].

Design documentation. Rationale capture systems provide a form of design documentation, recording what decisions are made, when they are made, who made them and why. This documentation enables people outside the project team to understand, supervise and regulate what is done by the project team. Another use of this documentation is to identify and secure intellectual property generated during a project [18,27,15,105]

Tazi [106] proposed to use design rationale for the documentation of complex engineering systems. Intricate, multi-disciplinary design problems have numerous quantities and qualities of information, correspondingly complex document structures, and many intended users (such as engineers, technical writers, editors, graphics specialists and formatters) involved in the production of the product and its documentation. In the case of the documentation of a complex system (e.g. an aircraft or large software system), the authors need to be able to include not only the information content derived from the design, but also communicative intentions – the traces of various design choices by the designers and the authors.

Design rationale as a learning aid. Carey [107] proposed to develop a library of exemplar artifacts and design rationales as a Human-Computer Interaction (HCI) learning aid, to help train inexperienced graphical user-interface designers. For designers (both in training and on the job) the study of GUI designs is hindered by the lack of explicit rationales to explain how the design achieves its objectives, and why other alternatives were deemed less attractive. A prototype system built as an extension to a commercial user-interface toolkit showed that generic design rationale components could guide inexperienced designers.

Construction of design models. Design is viewed as a process of constructing and evaluating a succession of increasingly complete design models until a final design is produced. Templates [108] have been used to guide the development of useful parts of design rationale by creating an instantiation of an existing template, and supporting the design that evolves as a sequence of models towards a final result. Three kinds of templates are introduced [108] which support *software user interface* development, establishing *design goals* and *design process management.*

9. Discussion

This survey has discussed design rationale systems from five perspectives: *knowledge representation, rationale capture, rationale retrieval, technical approach* and *application domain*. While considerable effort has been put into developing design rationale systems, none of these systems has been

adopted for widespread industrial use. Although designers can benefit by using design rationale systems, most systems are still in the laboratory stage. Further research needs to focus on the advancements needed to take the science to the level at which it can be effectively deployed in industry.

9.1. Representational Challenges

The challenge of design rationale representation is to find the best method to assist designers in making decisions, which means this representation must possess three qualities: *ease of input, effective view* and *activeness* [3]. For current systems, there are still some significant problems. For example, in IBIS-based design rationale systems, arguments are represented as non-interpreted text – hence a system cannot really understand the design, which jeopardises the usefulness of the method [31,24].

There is a considerable amount of work needed in design rationale representation. A system component should be developed for articulating and representing the task at hand, for example, how to create a Design Space Analysis by using QOC representation [41,5]. Design rationale systems should have the capability to represent potentially relevant features and combine features of objects in specific contexts to form coherent explanations in justification elicitation [54], to explore the possibility of representing more generic clauses in the formal components of the rationales [109], and to encode the modeling knowledge in a form that can be shared and reused by several applications, i.e. the objective of developing some representation formalisms for sharing and reusing declarative knowledge [61].

For future systems, a formal language must be developed to represent and reason about the state of the design at all levels of abstraction [10]. Systems should have the ability to provide different views, e.g. selectively hiding unnecessary detail and reorganising the information according to different criteria (such as subject, creator, status) rather than only chronologically. For argumentation systems, an issue is how to integrate authoring with browsing, which would enable the designer to annotate the issue base, record decisions on issues and generally personalise the argumentation [14,21].

9.2. Capture Challenges

The basic concern is how to capture process knowledge with minimal overhead, with the least interference with the natural progression of design activities. Most importantly, how can this be done without shifting the focus of designers' work from creative design tasks to the more tedious documentation tasks [27,26].

Other problems worth noting include how to resolve the conflicts that arise when new knowledge captured may violate the knowledge previously captured, and how to construct the new design rationale and keep it consistent for later use [9]. These problems share issues in common with research from mainstream AI in areas such as non-monotonic logic and consistency management in knowledge-based systems. For most design rationale systems, the current implementation is only able to record the design rationale after decisions are made, but not *while* they are being made. Further work could contribute to better tools for real-time capture and processing of rationale [14]. The ideal would be to have design rationale systems that can bridge the gap between communication and argumentation by structuring rationale after it has been captured [18, 21].

For the engineers themselves, some problems need to be noticed. Considering the differences in the preferred evaluation criteria used by different designers, especially between experts and novices, to develop formal languages to represent and reason about the states of the design at all levels [10] is a way to keep consistency in the capture and construction of design rationale [34].

9.3. Retrieval Challenges

Since process-oriented design rationale is organised chronologically, research on retrieval strategies is needed to manage the enormous amount of chronologically organised design rationale [3] and increase human usability and computational tractability [1]. Effective facilities need to be provided for users to get required information without navigating throughout the whole rationale space, such as by selectively hiding unnecessary detail and reorganizing the information [14,27].

To capture deliberations and reasonings in design processes, it is sometimes desirable to provide a set of predefined queries which address various known user needs [25]. These queries can be configured using the design rationale recorded for previous designs or by extracting analogies from previous design situations via case-based reasoning [14].

9.4. New Approaches

To bridge the representation gulf between designer and design rationale representation and strengthen design rationale research, we need to modify the language of communication (notation) and develop the representation medium (tool support) [2].

From our survey, while there have been design rationale systems and prototypes deployed in lab settings, much effort is needed before they will find widespread acceptance in industry [28]. To extend the system to a wider range of tasks such as commissioning, maintenance, operational analysis and planning, in any area where decisions are made, not just in design process [14].

9.5. Conclusions

Organisations subsist on communication and coordination. Whether the organisation is a multi-national conglomerate with a centuries-old history or school children playing basketball at recess, success of the organisation depends on the ability of its members to communicate and co-ordinate. The promise of design rationale systems is to address this need – that is, to empower organisations to create and manage their knowledge assets. However, despite claims of the 'great potential' of design rationale systems, demonstrated successes are rare.

Design rationale is not a new concept for organisations. Organisations have long been concerned with capturing and preserving their intellectual capital [110]. However, the introduction of new technologies and concepts can potentially change the way in which knowledge must be managed. Currently, much of the information generated within an organisation is stored with the people in the organisation. That is, information about procedures, goals, practices and history is stored in the people's memory, desk drawers and bulletin boards. For decades, this informal method of managing information was, although not ideal, generally adequate. However, as artifacts increase in complexity the need to capture the underlying design rationale increases. The goal of design rationale systems is to alleviate this problem.

A successful solution to this problems is one that facilitates the generation, storage and retrieval of information and associates the information with the designed artifact. Although that is the intent of design rationale systems, the survey presented in this paper shows that this intent is largely unmet.

In the sections below, we summarise the challenges that successful design rationale systems must meet. We include design challenges which, we believe, will be more difficult to overcome than the technical challenges.

9.5.1. Technical Challenges

First, many systems require a large number of knowledge workers to help organise and develop the content. This presents a problem for organisations with limited resources and a long list of other priorities. These organisations need to take advantage of intellectual capital [110] that is developed as part of the work process. One such example would be the communication and coordination of distributed work teams that many managers are now required to do.

Secondly, even when the organisation as a whole and individual groups are well supported, and content is generated without an unreasonable cost to the organisation, another challenge arises. This challenge is in making members of the organisation aware of all the relevant resources available to them. This awareness mechanism must be based on each individual's needs. It must also give the individual the freedom to tailor it to his/her preferences (e.g. push versus pull, delivery frequency, and individual interest profile).

Thirdly, design rationale technologies need to be designed specifically to suit the needs of the organisation in which they will be used [111]. It is virtually impossible for systems to be reused from one organisation to the next. Although platforms such as Lotus Notes and other intranet platforms have helped decrease the time it takes to develop individual applications, it is difficult to find a reusable architecture for a system which would support an entire organisation. Reuse is made more difficult because technology can only be introduced after work on the organisational issues discussed above has been initiated, and how these issues are resolved will certainly reflect how technological solutions should be designed.

9.5.2. Design Challenges

As Davenport and Prusak [112] warn in their book 'if you build it, they may not come'. Being able to build a system is only an initial step; the 'gold standard' against which success is measured, however, is whether people will accept and use it. As technologists, we did not have much control over the personal reward systems of the individual users and management mandate that many [112,111] recommend will enhance usage of the technology, and therefore we could not motivate our users as such.

However, it has been shown that other factors are also involved with designing and deploying such a system that contribute to its success [113,111]. By using a 'human-centred' approach [114,115], the level of use will be strongly affected by the usefulness and usability of the system deployed.

Following Grudin's suggestion [113], we need to design systems so that there are identifiable benefits to the people who use them. When an individual uses a system, the benefit gained from this experience should encourage him to continue using the system. We must strive to design systems in such a way that there are benefits to the current users, not just the future users. In doing so, the system will have a better chance of sustaining continued use.

Design rationale tools must support both formal and informal knowledge, making the system flexible enough so that broad content types were supported [114]. They must support multiple levels of organisation of content and design systems so that knowledge can be structured at any time after it is entered [35]. We do not want to force the content to be too structured, but need to provide structuring mechanisms so that it can be automatically structured or restructured at a later time.

As Grudin suggested [113], it is best to build upon an already successful application. The luck, of course, is in finding such an application, and in appropriately defining 'successful'. Building on an application that the user population is already familiar with reduces the overhead of learning to use a new system. Providing a totally new application for storing and retrieving information increases overhead and correspondingly decreases the probability of a successful system introduction.

Finally, builders of design rationale systems should borrow ideas from the field of Participatory Design [116]. Evolutionary Growth [117], the Improvisational model [118], and the framework specified in Zimmermann and Selvin [119]. Joint design of a system with the intended users is much more likely to succeed than designing for them.

9.5.3. A Final Comment

The need for design rationale is a common problem, but successful design rationale systems are rare. The need to record and preserve intellectual capital drives organisations to manage knowledge. Our objective in this paper is to provide present builders of design rationale environments a guidebook into the systems design and implementation issues.

We believe that, as the technological sophistication of both the needed software components and systems integration tools increase, design rationale capture and reuse may soon become more widespread in business practice. We began this survey by asking some questions, we conclude by offering some tentative answers. The questions we sought to answer are:

- If design rationale is useful, why are design rationale systems not in widespread use in industry?
- How can design rationale systems better support engineering design?
- What are the major obstacles to the creation of truly useful and usable design rationale systems?

In conducting this survey, we found many things that surprised us. We expected to find many descriptions of design rationale systems and few successful applications in the 'real world'; but we were surprised that the number of descriptions was so large and the number of applications was so small. This finding underscores the importance of our initial questions.

Looking across all the descriptions of design rationale systems, we see many useful insights and many good ideas. We summarise these insights and ideas with respect to the four issues in our framework: approaches, representation schema, capture and retrieval. It is clear, for example, that there is not one right approach to building design rationale systems; a process-oriented approach fits better in a dynamic design domain and a feature-oriented approach fits better in a highly standardised design domain. Similarly, there is not one right approach to representation. A representational approach, for example, would aid consistency of decision making while a descriptive approach would aid reuse of design components. Further, different representations can be effectively combined in a single system. Automatic capture of design rationale certainly seems like the right goal, even though it will be a hard one to achieve. We are less certain about automatic retrieval. The idea is appealing, but finding the boundary across which systems become 'intrusive' rather than 'helpful' will, we expect, be a difficult problem.

We were also surprised at the breadth of literature relevant to design rationale. Looking through our reference list, you will find many citations to the fields of engineering, computer science and software engineering. Most readers, we expect, will not be surprised to find these citations. Some readers may be surprised, however, at the number of citations to fields such as cognitive psychology, sociology, business, management and human-computer interaction. The breadth of citations underscores, for us,

that design rationale is best viewed not as an independent area of investigation, but as a subproblem in the broader area of knowledge management. As we indicated earlier in this section, the design rationale community has much to learn from (and to teach to) their colleagues in the broader knowledge management community.

Our tentative answer to the three questions above is that we know a great deal about approaches, representation schema, capture, and retrieval of information in design rationale systems. There are many good ideas and insights in current systems that should be able to be combined to make more effective systems. The primary challenges that we see are the technical challenges of organising and managing knowledge and the design challenges of using a human-centred approach to building useful and usable systems.

Acknowledgements

This work was supported in part by National Science Foundation (NSF), Knowledge and Distributed Intelligence in the Information Age (KDI) Initiative Grant CISE/IIS-9873005; CAREER Award CISE/IRIS-9733545 and Grants ENG/DMI-9713718 and CISE/CDA-9729827. Additional support was provided by The National Institute of Standards and Technology (NIST) Grant 60NANB7D0092 and AT&T Labs.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s), and do not necessarily reflect the views of the National Science Foundation or the other supporting government and corporate organisations.

References

1. Lee, J., Lai, K. (1991) What's in design rationale. Human-Comput. Interaction, 6(3–4), 251–280
2. Buckingham-Shum, S. J., Hammond, N. (1994) Argumentation-based design rationale: What use at what cost? Human-Comput. Stud., 40(4), 603–652
3. Conklin, J. E., Burgess Yakemovic K. C. (1991) A process-oriented approach to design rationale. Human-Comput. Interaction, 6(3–4), 357–391
4. Gruber, T. R., Russell, D. M. (1992) Design knowledge and design rationale: A framework for representation, capture, and use. Technical Report KSL 90–45, Knowledge Systems Laboratory, Stanford University
5. MacLean, A., Young, R., Belloti, V., Moran, T. (1991) Questions, options, and criteria: Elements of design space analysis. Human-Comput. Interaction, 6(3–4), 201–250
6. Rosenman, M. A., Gero, J. S. (1994) The what, the how and the why in design. Appl. Artif. Intell., 8(2), 199–218
7. Klein, M. (1993) Capturing design rationale in concurrent engineering teams. IEEE Computer, 26(1), 39–47
8. Guindon, R., Krasner, H., Curtis, B. (1987) Breakdowns and processes during the early activities of software design by professionals. In: Olson, G. M. (ed.) Empirical Studies of Programmers: Second Workshop, Ablex, pp 65–82
9. Prabhakar, S., Goel, A. K. (1998) Functional modeling for enabling adaptive design of devices for new environments. Artif. Intell. in Eng., 12(4), 417–444
10. Ullman, D. G., Dietterich, P. G., Stauffer, L. A. (1988) A model of the mechanical design process based on empirical data. Artif. Intell. for Eng. Design, Analysis and Manuf., 2(1), 33–52
11. Moran, T. P., Carroll, J. M. (eds.) (1996) Design Rationale: Concepts, techniques, and use. Lawrence Erlbaum
12. Lee, J. (1997) Design rationale systems: Understanding the issues. IEEE Expert/Intelligent Systems and their Applic., 12(3), 78–85
13. Thompson, J. B., Lu, S. C. Y. (1989) Representing and using design rationale in concurrent product and process design. In: Chao, N. H., Lu, S. C. Y. (eds) Concurrent Product and Process Design, ASME Winter Annual Meeting, ASME, pp 109–115
14. Banares-Alcantara, R., King, J. M. P. (1997) Design support systems for process engineering iii – design rationale as a requirement for effective support. Comput. and Chem. Eng., 21(3), 263–276
15. King, J. M. P., Banares-Alcantara, R. (1997) Extending the scope and use of design rationale. Artif. Intell. for Eng. Design, Analysis and Manuf., 11(2), 155–167
16. Fischer, G., McCall, R., Morch, A. (1989) Design environments for constructive and argumentative design. In: Bice, K., Lewis, C. (eds.), Conference on Wings for the Mind, Addison-Wesley, pp 269–275
17. Kunz, W., Rittel, W. (1970) Issues as elements of information systems. Working paper 131, Center for Planning and Development Research, University of California, Berkeley
18. Shipman III, F. M., McCall, R. J. (1997) Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. Artif. Intell. for Eng. Design, Analysis and Manuf., 11(2), 141–154
19. Gruber, T. R., Russell, D. M. (1992) Generative design rationale: Beyond the record and replay paradigm. Technical Report KSL 92–59, Knowledge Systems Laboratory, Stanford University
20. McCall, R. J. (1991) PHI: A conceptual foundation for design hypermedia. Design Studies, 12(1), 30–41
21. Fischer, G., McCall, R. (1989) JANUS: Integrating hypertext with a knowledge-based design environment. In: Halasz, F., Meyrowitz, N. (eds.), Hypertext, Addison-Wesley, pp pages 105–117
22. Lubars, M. D. (1991) Representing design dependencies in an issue-based style. IEEE Software, 6(4), 81–89
23. Mark, W., Tyler, S., McGuire, J., Schlossberg, J. (1992) Commitment-based software development. IEEE Trans. Softw. Eng., 18(10), 870–886
24. Garcia, A. C. B., Howard, C. H. Acquiring design knowledge through design decision justification. Artif. Intell. for Eng. Design, Analysis and Manuf., 6(1), 59–71

25. Ramesh, B., Dhar, V. (1992) Supporting systems development using knowledge captured during requirements engineering. IEEE Trans. Softw. Eng., 18(6), 498–511
26. Ramesh, B., Sengupta, K. (1995) Multimedia in a design rationale decision support system. Decision Support Syst., 15(3), 181–196
27. Garcia, A. C. B., de Souza C. S. (1997) Add+: Including rhetorical structures in active documents. Artif. Intell. for Eng. Design, Analysis and Manuf., 11(2), 109–124
28. de la Garza, J. M., Alcantara Jr, P. T. (1997) Using parameter dependency network to represent design rationale. J. Comput. in Civil Eng., 11(2), 102–112
29. de la Garza, J. M., Ramakrishnan, S. (1995) A tool for designers to record design rationale of a constructed project. 10th International Conference on Applications of Artificial Intelligence in Engineering, Southampton, U.K. Computational Mechanics Publications, pp 533–540
30. Chandrasekaran, B., Iwasaki, Y. (1993) Functional representation as design rationale. IEEE Computer, 26(1), 48–56
31. Chung, P. W. H., Goodwin, R. (1998) An integrated approach to representing and accessing design rationale. Eng. Applic. Artif. Intell., 11(1), 149–159
32. Myers, K. L., Zumel, N. B., Garcia, P. (1999) Automated capture of rationale for the detailed design process. In: Uthurusamy, R: Hayes-Roth, B. (eds.), Eleventh Conference on Innovative Applications of Artificial Intelligence, AAAI Press, pp 876–883
33. Gruber, T. R., Russell, D. M. (1992) Derivation and use of design rationale information as expressed by designers. Technical Report KSL 92–64, Knowledge Systems Laboratory, Stanford University
34. Guindon, R. (1990) Knowledge exploited by experts during software system design. Int. J. Man-Machine Stud., 33, 279–304
35. Shipman III, F. M., McCall, R. J. (1994) Supporting knowledge-base evolution with incremental formalization. In: Adelson, B., Dumais, S., Olson, J. (eds.), Human Factors in Computing Systems: 'Celebrating Interdependence', Boston, MA: ACM/SIGCHI, pp 285–291
36. Brazier, F. M. T., Van Langen, P. H. G., Treur, J. (1997) A compositional approach to modelling design rationale. Artif. Intell. for Eng. Design, Analysis and Manuf., 11(2), 125–139
37. Goel, A. (1991) Model revision: A theory of incremental model learning. 8th International Conference on Machine Learning, Chicago, IL. Morgan Kaufman, pp 605–609
38. Doyle, J. (1979) A truth-maintenance system. Artif. Intell., 12(3), 231–272
39. Liang, J., Shah, J. J., D'Souza, R., Urban, S. D., Ayyaswamy K., Harter, E., Bluhm, T. (1999) Synthesis of consolidated data schema for engineering analysis from multiple STEP application protocols. Comput. Aided Design, 31(7), 429–447
40. Shah, J. J., Rangaswamy, S., Qureshi, S., Urban, S., (1999) Design history system: Data models and prototype implementation. Knowledge Intensive Computer Aided Design, Kluwer, pp 91–114
41. Fischer, G., Lemke, A. C., McCall, R. (1991) Making argumentation serve design. Human-Comput. Interaction, 6(3–4), 393–419
42. Buckingham-Shum, S. J., MacLean, A., Bellotti, V. M. E., Hammond, N. V. (1997) Graphical argumentation and design cognition. Human-Comput. Interaction, 12(3), 267–300
43. Lee J. (1990) SIBYL: A qualitative decision management system. In: Winston, P. H., Shellard, S. A. (eds.), Artifical Intelligence at MIT: Expanding Frontiers, Vol. 1, The MIT Press, pp 106–133
44. Franke, D. W. (1991) Deriving and using descriptions of purpose. IEEE Expert/Intelligent Systems and their Applic., 6(2), 41–47
45. Bobrow, D. G. (1984) Qualitative reasoning about physical systems: An introduction. Artif. Intell., 24(1–3), 1–5
46. Fowler, J. E. (1996) Variant design for mechanical artifacts: A state-of-the-art survey. Eng. with Comput., 12, 1–15
47. Goel, A. (1991) A model-based approach to case adaptation. 13th Annual Conference of the Cognitive Science Society, Cognitive Science Society, Lawrence Erlbaum, pp 143–148
48. MacGregor, R. (1990) The evolving technology of classification-based knowledge representation systems. In: Sowa, J. F. (ed.), Principles of Semantic Networks: Explorations in the Representation of Knowledge. Morgan Kaufman
49. Mann, W. C., Thompson, S. A. (1987) Rhetorical structure theory: A theory of the text organization. Technical Report ISURS-87-190, Information Science Institute
50. Sowa, J. F. (1984) Conceptual Structures: Information Processing in Mind and Machine. Addison Wesley
51. Brazier, F. M. T., Van Langen P. H. G., Treur, J., Wijngaards, N. J. E., Willems, M. (1996) Modelling an elevator design task in DESIRE: the vt example. Int. J. Human-Comput. Stud., 44(3–4), 469–520
52. Poltrock, S. E., Grudin, J. (1999) Cscw, groupware and workflow: Experiences, state of the art, and future trends. ACM SIGCHI Tutorial, Pittsburgh, PA
53. Brazier, F. M. T., Van Langen, P. H. G., Treur, J. (1995) Modelling conflict management in design: An explicit approach. Artif. Intell. for Eng. Design, Analysis and Manuf., 9(4), 355–366
54. Gruber, T. R., Russell, D. M. (1991) Interactive acquisition of justifications: Learning why by being told what. IEEE Expert/Intelligent Systems and their Applic., 6(4), 65–75
55. Stahovich, T. F. (1997) Interpreting the engineer's sketch: A picture is worth a thousand constraints. In: Anderson, M. (ed.), AAAI Symposium on Reasoning with Diagrammatic Representations II, Providence, Rhode Island. AAAI Press, pp 31–38
56. Stahovich, T. F., Davis, R., Shrobe, H. (1996) Generating multiple new designs from a sketch. The National Conference on Artificial Intelligence, Portland, Oregon. AAAI Press, pp 1022–29
57. Aamodt, A., Plazas, E. (1994) Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Comm., 7(1), 39–52
58. Bardasz, T., Zeid, I. (1991) Applying analogical problem solving to mechanical design. Int. J. Comput. Aided Design, 23(3), 202–212

59. Bardasz, T., Zeid, I. (1992) Cognitive models of memory for mechanical design problems. Int. J. Comput. Aided Design, 24(6), 327–342
60. Maher, M. L., Balachandran, M. B., Zhang, D. M. (1995) Case-Based Reasoning in Design. Lawrence Erlbaum
61. Gruber, T. R. (1992) Model formulation as a problem-solving task: Computer-assisted engineering modeling. Int. J. Intell. Syst., 8, 105–128 (Also available as technical report KSL-92-57 at Knowledge Systems Laboratory, Stanford University)
62. Suh, N. P. (1990) The Principles of Design, Vol. 1. Oxford University Press
63. Lakin, F., Wambaugh, H., Leifer, L., Cannon, D., Sivard, C. (1989) The electronic design notebook: Performing medium and processing medium. The Visual Computer: Int. J. Comput. Graphics, 5, 214–226
64. Sycara, K., Navinchandra, D. (1992) Retrieval strategies in a case-based design system. In: Tong, C., Sriram, D., (eds.), Artificial Intelligence in Engineering Design, Vol. II. Academic Press
65. Laiserin, J. (1999) A new kind of CAD–Communication-Aided Design. CADENCE, pp 18–26
66. Ginsberg, M. L. (1991) Knowledge interchange format: The KIF of death. AI Mag., 12(3), 57–63
67. Bobrow, B. G., Norman, D. (1977) An overview of KRL: A knowledge representation language. Cognitive Sci., 1, 3–46
68. Brachman, R. J., Schmolze, J. G. (1985) An overview of the KL-ONE representation system. Cognitive Sci., 9, 171–216
69. Evett, M. P., Hendler, J. A., Spector, L. (1994) Parallel knowledge representation on the Connection Machine. J. Parallel and Distributed Comput., 22, 168–184
70. Stein, L. A. (1996) Science and engineering in knowledge representation and reasoning. AAAI Mag., 17(4), 77–83
71. Brachman, R. J., Levesque, H. J. (eds) (1985) Readings in Knowledge Representation. Morgan Kaufmann
72. Chittaro, L., Kumar, A. N. (1998) Reasoning about function and its application to engineering. Artif. Intell. in Eng., 12(4), 331–226
73. Kumar, A. N., Upadhyaya, S. J. (1998) Component-ontological representation of function for reasoning about devices. Artif. Intell. in Eng., 12(4), 399–415
74. Kim, J., Ringo Ling, S., Will, P. (1997) Ontology engineering for active catalog. Technical report, The University of Southern California, Information Sciences Institute
75. Schlenoff, C., Denno, P., Ivester, R., Libes, D., Szykman, S. (1999) An analysis of existing ontological systems for applications in manufacturing. ASME Design Engineering Technical Conferences, 19th Computers and Information in Engineering Conference, New York, NY. ASME Press
76. Szykman, S., Racz, J. W., Sriram, R. D. (1999) The representation of function in computer-based design. ASME Design Engineering Technical Conferences, 11th International Conference on Design Theory and Methodology, New York, NY. ASME Press
77. Szykman, S., Senfaute, J., Sriram, R. D. Using XML to describe function and taxonomies in computer-based design. ASME Design Engineering Technical Conference, 19th Computers and Information in Engineering Conference, New York, NY. ASME Press
78. Chen, A., McGinnis, B., Ulman, D. G., Dietterich, T. G. (1990) Design history knowledge representation and its basic computer implementation. In: Rinderle, J. R. (ed.), Proceedings of the Design Theory and Methodology Conference, ASME, 175–184
79. Catron, B., Ray, S. (1991) Alps: A language for process specification. Int. J. Comput. Integrated Manuf. 4(2), 105–113
80. Schlenoff, C., Knutilla, A., Ray, S. (1996) Unified process specification language: Requirements for modeling process. Technical Report NISTIR 5910, National Institute of Standards and Technology, Gaithersburg, MD
81. Dong, A., Agogino, A. (1997) Text analysis for constructing design representations. Artif. Intell. in Eng., 11(2), 65–75
82. Thompson, J. B., Liu, S. C.-Y. (1990) Design evolution management: A methodology for representing and utilizing design rationale. Proceedings of the Second International ASME Conference on design Theory and Methodology
83. Urban, S. D., Ayyaswamy, K., Fu, L., Shah, J. J., Liang, J. (1999) Integrated product data environment: Data sharing across diverse engineering applications. Int. J. Comput. Integrated Manuf., 12(6), 525–540
84. Liang, J., Shah, J. J., Souza, R. D., Urban, S. D., Ayyaswamy, K., Harter, E., Bluhm, T. (1999) Synthesis of consolidated data schema for engineering analysis from multiple step application protocols. Int. J. Comput. Aided Design, 21, 429–447
85. Xue, D., Yadav, S., Norrie, D. H. (1999) Knowledge base and database representation for intelligent concurrent design. Comput. Aided. Design., 31(2), 131–145
86. Steels, L. (1990) Components of expertise. Al Mag., 11(2), 28–49
87. Gruber, T. R. (1990) The role of standard knowledge representation for sharing knowledge-based technology. Technical Report KSL 90–53. Knowledge Systems Laboratory, Stanford University
88. Cutkosky, M. R., Tenenbaum, J. M., Glicksman, J. (1996) Madefast: Collaborative engineering over the internet. Comm. ACM, 39(9), 78–87
89. Olsen, G. R., Cutkosky, M., Tenenbaum, J. M., Gruber, T. R. (1994) Collaborative engineering based on knowledge sharing agreements. ASME Database Symposium. Minneapolis, MN
90. Tove, G., Cutkosky, M. R., Leifer, L. J., Tenenbaum, J. M., Glicksman, J. (1993) Share: A methodology and environment for collaborative product development. IEEE Workshop on Infrastructure for Collaborative Technologies (Also available as Stanford CDR-TR #19930507)
91. Cutkosky, M. R., Tenenbaum, J. M. (1992) Toward a framework for concurrent design. Int. J. Syst. Automation: Res. and Applic., 1(3), 239–261
92. Cutkosky, M. R., Engelmore, R. S., Fikes, R. E., Genesereth, M. R., Gruber, T. R., Mark, W. S., Tenenbaum, J. M., Weber, J. C. (1993) Pact: An experiment in integrating concurrent engineering systems. IEEE Computer, 26(1), 28–38
93. Sriram, D., Stephanopoulos, G., Logcher, R., Gossard, D., Groleau, N., Serrano, D., Navinchandra, D. (1989)

Knowledge-based systems applications in engineering design research at mit. Al Mag., 10(3), 79–96
94. Sriram, R. D. (1997) Intelligent Systems for Engineering: A Knowledge-Based Approach. Springer-Verlag
95. Duffy, A. H. B. (1997) The 'what' and 'how' of learning in design. IEEE Expert/Intelligent Systems and their Applic., 12(3), 71–76
96. Sim, S. K., Duffy, A. H. B. (1998) A foundation for machine learning in design. Artif. Intell. for Eng. Design, Analysis and Manuf., 12(2), 193–209
97. Umeda, Y., Tomiyama, T. Functional reasoning in design. IEEE Expert/Intelligent Systems and their Applic., 12(2), 42–48
98. Duffy, S. M., Duffy, A. H. B. (1996) Sharing the learning activity using intelligent cad. Artif. Intell. for Eng. Design, Analysis, and Manuf., 10(2), 83–100
99. Gero, J. S., Sudweeks, F. (eds.), Second International Conference on Artificial Intelligence in Design. Kluwer Academic
100. Liu, X., Gan, M. (1991) A preliminary structural design expert system (spred-1) based on neural networks. In: Gero, J. S. (ed). First International Conference on Artificial Intelligence in Design (AID'91), Edinburgh, Scotland, pp 785–799.
101. Maher, M. L., Brown, D. C., Duffy, A. H. B. (1994) Special issue on machine learning in design. Artif. Intell. for Eng. Design, Analysis and Manuf., 8(2), 81–82
102. Schreiber, G., Wielingas, B. (1997) Configuration-design problem solving. IEEE Expert/Intelligent Systems and their Applic., 12(2), 49–56
103. Sycara, K., Navinchandra, D. (1989) Representing and indexing design cases. Proceedings of the Second International Conference on Industrial Engineering Applications of Artificial Intelliegence and Expert Systems, Tullahoma, T.N. ACM Press, pp 735–741
104. Mostow, J. (1989) Design by derivational analogy: Issues in the automated replay of design plans. Artif. Intell., 40(1–3), 119–184
105. Pena-Mora, F., Vadhavkar, S. (1997) Augmenting design patterns with design rationale. Artif. Intell. For Eng. Design, Analysis and Manuf., 11(2), 93–108
106. Tazi, S., Novick, D. (1998) Design rationale for complex system documentation. Conference on Complex Systems, Intelligent Systems and Interface, Nimes, France
107. Carey, T., McKerlie, D., Wilson, J. (1996) HCI design rationale as a learning resource. In: Moran, T. P., Carroll, J. M. (eds.), Design Rationale Concepts, Techniques and Use, Ist ed. Lawrence Erlbaum. pp 373–392
108. Casaday, G. (1996) Rationale in practice: Templates for capturing and applying design experience. In: Moran, T.P., Carroll, J. M. (eds.), Design Rationale Concepts, Techniques and Use, Ist ed. Lawrence Erlbaum, pp 351–372
109. Johnson, C.W. (1996) Literate specification: Using design rationale to support formal methods in the development of human-machine interfaces. Human-Comput. Interaction, 11(4), 291–320
110. Putnam, R. D. (1993) Making Democracy Work: Civic Traditions in Modern Italy. Princeton University Press
111. Orlikowski, W. J. (1992) Learning from notes: Organizational issues in groupware implementation. Conference on Computer Supported Cooperative Work (CSCW), Toronto, Canada. ACM/SIGCHI, pp 362–369
112. Davenport, T.H., Pruzak, L. (1998) Working Knowledge: How Organizations Manage What They Know, Vol. 1. Harvard Business School Publishing
113. Grudin, J. (1994) Groupware and social dynamics. Comm. ACM, 37(1), 93–105
114. Davenport, T. H. (1994) Saving IT's soul: Human-centered information management. Harvard Bus. Rev., 94(2), 39–53
115. Mackayn, W. (1990) Patterns of sharing customizable software. Conference on Computer Supported Cooperative Work (CSCW), Los Angles, CA. ACM/SIGCHI, pp 209–221
116. Greenbaum, J., Kyng, M. (eds.) (1991) Design at Work – Cooperative Design of Computer Systems. Laurence Erlbaum
117. Fischer, G. (1994) Domain-oriented design environments. Applications and Impacts, Information Processing, pages Hamburg, Germany. International Federation for Information Processing, North-Holland, pp 115–122
118. Orlikowski, W. J., Hofman, J. D. (1997) An improvisational model for change management: The case of groupware technologies. Sloan Manage. Rev., 38(2), 11–21
119. Zimmermann, B., Selvin, A. M. (1997) A framework for assessing group memory approaches for software design projects. Conference on Designing Interactive Systems, New York, N.Y. ACM Press, pp 417–426