A SURVEY ON DESIGN RATIONALE: REPRESENTATION, CAPTURE AND RETRIEVAL

Xiaochun Hu¹, Jun Pang, Yan Pang, Michael Atwood², Wei Sun¹, William C. Regli³

Geometric and Intelligent Computing Laboratory
Department of Mathematics and Computer Science
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104

ABSTRACT

This paper provides a brief 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 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 axes along which to describe and classify design rationale systems, including argumentation-based, descriptive, and 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 CAD tools.

This survey is structured around the fundamental different approaches, their representation schema, their capture methods, and retrieval techniques. A number of recent design rationale systems and representation schemes are presented, including JANUS, COMET, ADD, REMAP, HOS, PHIDIAS, DRIVE, IBIS. We conclude with an assessment of the current state-of-the-art and a discussion of critical open research issues.


INTRODUCTION

Design rationale is an explanation of why an artifact, or some part of it, is designed the way it is (30). 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 (6; 9; 24; 32; 42). 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.

Much progress has been made on the development of design rationale approaches and tools since the early 1980’s. The research has ranged from basic observations about the design process (46) 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. Recent research has a tendency to combine design rationale systems with
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 to think and discuss design within other forms of design support tools (2; 27).

One of fundamental goals of this survey is to answer several questions: if knowing design rationale is useful, why are design rationale systems not in widespread use in engineering; how can design rationale systems be better used to support engineering design; what are the obstacles to the creation of truly useful and usable design rationale systems? Some of our speculations on these issues are presented in our discussion and conclusions.

This survey is organized as follows: first, we give a framework of the survey, introduce basic concepts of design rationale, and review design rationale systems or prototypes in chronological order and summarized. Next, we provide a detailed description of a number of systems according to the survey framework. Lastly, we summarize our review and discuss the open research issues that must be addressed in order to advance the state-of-the-art in design rationale systems.
a certain knowledge representation framework. Figure 1 shows
the data flow of a general design rationale system, which was
generated from our survey by synthesizing the functions of dif-
ferent systems. The design rationale systems or prototypes that
we reviewed may not include all of the features in the diagram.

We categorized our survey on design rationale system into
four areas, which are described below:

**Approaches to Developing Design Rationale Systems**

The main approaches to design rationale are process-oriented and
feature-oriented. In fields with a relatively high degree of stan-
dardization, the feature-oriented approach is used to give logical
representation of artifacts, to follow the rigorous and logical
rules of the design process, while in dynamic design domains the
process-oriented approach is used to give historical representa-
tion of artifacts (6; 9). The integration of a design rationale sys-
tem with other design support systems such as knowledge-based
design decision support systems, commercial CAD systems, and
communication systems, also influences its approach.

**Representation Schema for Design Rationale**

Designing an artifact involves considering many alternatives—where one must
address, evaluate, and ultimately accept or reject each alterna-
tive. The choice of a representation schema is a critical issue (6)
because it determines how to organize this enormous amount of
diverse material and build it into a usable structure, and deter-
mines how to capture and retrieve the design rationale. Much ef-
fort has been made to achieve the following goals: to help lay out
the structure of arguments, to maintain consistency in decision-
-making, to keep track of decisions, to communicate about design
reasoning, to minimize the intrusion to designers, and to help
form answers in response to inquires. From our review, no rep-
resentation schema achieves all of the above goals.

**Capture of Design Rationale**

In a design process, capturing design rationale involves recording the reasoning, decisions, op-
tions, trade-offs, etc., and constructing a formal (39) or semi-
formal structure (6) so that the design rationale can be used in
the decision-making process during design. What should be
captured as design rationale and how to capture it are subjects
that many researchers have focused on. We observed that the
recording and construction processes have been implemented in
design rationale systems through both automatic capture or
user-intervention (i.e., requiring designers to input or record the
design discussions, decisions and reasonings themselves) ap-
proaches.

**Retrieval of Design Rationale**

At each different design stage,
there are various purposes for accessing design rationale: to an-
swer a user query, to show the logical aspects of an important
issue, to monitor design progress, or to get a document about the
designed artifact (8; 17; 23). Retrieval strategies are determined
by the representation schema and the requirements in design pro-
cess. The integration of design rationale systems with other de-
sign support systems can greatly improve the retrieval of design
rationale.

To provide a general view of the development of the research
area, we list certain design rationale systems (in chronological
order) according to the catalog described above in Table 1. Each
system is described in five perspectives: knowledge representa-
tion, capture, retrieval, approach and their application domain.
Following sections will concentrate on the discussion of the sys-
tems cataloged in the table.

**APPROACHES TO BUILDING DESIGN RATIONALE
SYSTEMS**

At different stages of the design process of an artifact, de-
sign could be more process-oriented or more feature-oriented. In
design stages in which the problems are vague, the solution tech-
nology is poorly understood, or both, and in which there is little
or no standardization of designed artifacts (9), 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 (25). This design process
is more dynamic, with knowledge organized according to de-
sign progress, so the approach of design rationale at this stage
is more process-oriented. At the detailed design stage or in rou-
tine design, design process is more constrained by the rules in the
field or domain knowledge, so it is more a feature-oriented (18;
37) design process, in which features could be function, perfor-
mance, design, manufacture or implementation.

**Process-Oriented Approaches**

Process-oriented design rationale systems emphasize the de-
sign rationale as a history of the design process. Most of the de-
sign rationale approaches are process-oriented, in which the is-
ues, options and arguments are captured and organized accord-
ing to the design progress.

This approach originated from the Issue-Based Information
System (IBIS) framework for argumentation (28). A number of
other frameworks have been developed since then, including
DRL (30) and PHI (36).

The representation schema of this kind of rationale system
is generally graph-based, using nodes and links, with nodes indi-
cating 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 com-
munications, since multimedia nodes could be included as part
<table>
<thead>
<tr>
<th>System Name Acronym</th>
<th>Knowledge Representation</th>
<th>Knowledge Capture</th>
<th>Knowledge Retrieval</th>
<th>Approach</th>
<th>Design Domain</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRACK (15)</td>
<td>N/A</td>
<td>Auto</td>
<td>Trigger</td>
<td>FO</td>
<td>Kitchens</td>
<td>1989</td>
</tr>
<tr>
<td>VIEWPOINTS (15)</td>
<td>IBIS</td>
<td>N/A</td>
<td>Navigate</td>
<td>FO</td>
<td>Kitchen</td>
<td>1989</td>
</tr>
<tr>
<td>JANUS (14)</td>
<td>PHI</td>
<td>Auto</td>
<td>Hybrid</td>
<td>FO</td>
<td>Kitchen</td>
<td>1989</td>
</tr>
<tr>
<td>IBIS-style browser (31)</td>
<td>IBIS</td>
<td>Auto</td>
<td>Navigate</td>
<td>PO</td>
<td>Generic</td>
<td>1991</td>
</tr>
<tr>
<td>COMET (35)</td>
<td>LOOM</td>
<td>UI</td>
<td>Navigate</td>
<td>FO</td>
<td>Sensor-based tracker software</td>
<td>1992</td>
</tr>
<tr>
<td>ADD (18)</td>
<td>Argumentation &amp; Model-based</td>
<td>UI</td>
<td>Trigger</td>
<td>FO</td>
<td>HVAC</td>
<td>1992</td>
</tr>
<tr>
<td>REMAP (40)</td>
<td>IBIS</td>
<td>UI</td>
<td>Query</td>
<td>PO</td>
<td>Generic</td>
<td>1992</td>
</tr>
<tr>
<td>REMAP/MM (41)</td>
<td>IBIS</td>
<td>Auto</td>
<td>Query</td>
<td>PO</td>
<td>Generic</td>
<td>1995</td>
</tr>
<tr>
<td>ADD+ (17)</td>
<td>Rhetorical Structure</td>
<td>UI</td>
<td>Query</td>
<td>PO</td>
<td>HVAC</td>
<td>1997</td>
</tr>
<tr>
<td>HOS (44)</td>
<td>PHI</td>
<td>Auto</td>
<td>Trigger</td>
<td>PO</td>
<td>Generic</td>
<td>1997</td>
</tr>
<tr>
<td>PHIDIAS (44)</td>
<td>PHI</td>
<td>Auto</td>
<td>Trigger</td>
<td>FO</td>
<td>FO</td>
<td>2D, 3D</td>
</tr>
<tr>
<td>KBDS-IBIS (2), (27)</td>
<td>IBIS</td>
<td>UI</td>
<td>Query &amp; Navigate</td>
<td>FO</td>
<td>Chemical Plant</td>
<td>1997</td>
</tr>
<tr>
<td>DRIVE (10)</td>
<td>PDN</td>
<td>UI</td>
<td>Query</td>
<td>FO</td>
<td>Building</td>
<td>1997</td>
</tr>
<tr>
<td>DRARS (11)</td>
<td>QOC</td>
<td>N/A</td>
<td>N/A</td>
<td>FO</td>
<td>Building</td>
<td>1995</td>
</tr>
<tr>
<td>KRITIK (7), (39)</td>
<td>SBF</td>
<td>UI</td>
<td>Query</td>
<td>FO</td>
<td>Mechanical</td>
<td>1993</td>
</tr>
<tr>
<td>IDIS (8)</td>
<td>IBIS</td>
<td>UI</td>
<td>Navigate</td>
<td>FO</td>
<td>Chemical Plant</td>
<td>1998</td>
</tr>
<tr>
<td>RCF (37)</td>
<td>N/A</td>
<td>Auto</td>
<td>N/A</td>
<td>PO</td>
<td>N/A</td>
<td>1999</td>
</tr>
</tbody>
</table>

**Table 1.** Table of prototype design rationale systems.

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

of the design schema (41; 44).

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. Effort has been made to create links among nodes; PHIDIAS (44) links the nodes by authoring and indexing; REMAP/MM (41) supports hyper-links among design deliberation records and multimedia objects. This conversion process can present a large overhead to the design rationale maintainer (a smart computer or a patient human being). Another method to handle the recorded information is incremental formalization, described in (43).

This approach to design rationale helps designers by providing descriptive history information answering questions such as why, what, and when. The completeness and meaningfulness of design rationale is dependent on the logical organization of captured information. Currently, it is not easy to translate this into representations that can be understood and processed by computers, so this approach provides support to design process only when designers access it and understand it (14).

**Feature-Oriented Approaches**

A feature-oriented approach starts from the design space of the artifact. It is usually developed in a task specific con-
text using an empirical study. Some models are extended from generic Design Decision Support systems (DDS), adding additional primitives to explicitly represent the design processes of a certain task. Generally, these kind of systems contain domain knowledge-bases, which can be used to support automated reasoning. CRACK is a typical feature-Oriented system (15); a detailed description is given in 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 a process-oriented design rationale system. In some systems, the design rationale is represented with links to the existing knowledge-base. The retrieval and reuse of design rationale seems very natural in the design process of later artifacts. An example model is GTMD (5), which is used to structure the acquisition process within DESIRE, a formal framework for compositional modeling.

Design rationale systems using this approach are usually included in design systems. This makes it possible to provide more active support to design or redesign processes, such as the evaluation of design decisions and conflict resolution (10).

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 (7).

A combination of the two approaches could overcome many of the limitations of either. Systems with 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 (2).

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 1 gives a general view of the integration of a design rationale system with other systems.

There is a trend towards tight integration of design rationale tools with other design representations (6), with the design rationale system being treated as an extension of the design system. Design rationale systems integrated with commercial CAD systems were reported in (2; 8; 37; 17; 18; 20; 41; 44). In some of these systems, the integration was achieved by the combination of design rationale system with knowledge-based design decision support systems (8; 17; 20; 41). 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 (39), and makes the capture and retrieval of design rationale driven by implementation concerns. The integration of design rationale with telecommunication was reported in PHIDIAS (41; 44).

REPRESENTATION SCHEMAS

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.

Argumentation-based design rationale is a mainly representational approach which uses a semi-formal graphical format (6) for laying out the structure of arguments. It uses node-and-link representation, with nodes representing components, and links presenting the 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 organizing information are IBIS, PHI, QOC and DRL.

The Issue-Based Information System (IBIS)

The Issue-Based Information System (IBIS) is an issue-based approach that has been used in architectural design, city planning, and public-policy discussion. The key issues 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. An IBIS-style browser (31) is the implementation of a merged issue-based and truth-maintenance (12) dependency structure. It uses a graphical shorthand to represent nodes and links in a graph-based structure and can provide immediate feedback to designers by indicating the belief status of various issues (via color or other notations on the nodes).

The Representation and Maintenance of Process knowledge (REMAP) system (40) is also based on the Issue Based Information System (IBIS) method. 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. 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.

In an integrated and prescriptive form, Knowledge Based
Design System-IBIS (KBDS-IBIS) (2) 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. KBDS-IBIS introduced three new classes of object—artifacts, steps and tests—which integrate the representation of argumentation with the design process. Chung (8) 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 can be done easily.

Procedural Hierarchy of Issues (PHI)

The Procedural Hierarchy of Issues (PHI) (36) 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. Second, 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. Compared with IBIS, PHI provides dependency relationships between issue resolutions and considers the pros and cons of alternative answers (13), and more completely and accurately models the task structure of the design process, providing information that is more useful for tasks (14).

VIEWPOINTS (15) is a hypertext system for argumentative kitchen design based on the PHI design methodology. The elements of VIEWPOINTS are issues, answers, arguments and graphics. A graphical interface is used to facilitate its use, enabling designers to do extensive browsing and make decisions. CRACK (15), another computer-supported design rationale system, is a knowledge-based critic which has knowledge about how kitchen appliances can assembled into functional kitchen. JANUS (14) is the integration of these two systems. It allows architectural and interior designers to graphically construct artifacts by direct manipulation, and at the same time receive 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 the design.

PHIDIAS (44) is a hypermedia system which is based on PHI and 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. The Hyper-Object Substrate (HOS) (44) 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.

Design Space Analysis

Design Space Analysis places an artifact in a space of possibilities and seeks to explain why the particular artifact was chosen from these possibilities. QOC represents the design space using three components: questions identify key issues for structuring the space of alternative; options provide possible answers to the questions; criteria are the bases for evaluating and choosing among the options. The QOC representation emphasizes the systematic development of a space of design options structured by questions, and 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. Design Rationale Authoring and Retrieval System (DRARS) (11) is a design rationale system using a variation of QOC (11). 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 different way to record design rationale is using design rationale as an account of how the designed artifact serves or satisfies expected functionalities, as in Decision Rationale Language (DRL) (30) and Function Representation (FR) (7). DRL is an expressive language, which represents the space around decisions. It can be maintained independently or integrated with traditional design representation. DRL was implemented in a system called SIBYL (29), and is being used to explore various kinds of computational service over DRL structures.

Functional Representations

Functional Representation (FR) (7), a representational scheme, describes how a device works (or is intended to work). In 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 behavior 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. It can support many of the above tasks; the limitation of
FR as design rationale is that it only captures the causal knowledge about device operation.

The Structure, Behavior and Function (SBF) model [20] is an approach for designing devices which explicitly represents the functions of the device (the problem), the structure of the device (the solution), and the internal causal behaviors of the device. Function can be defined as what (an object) does, behavior as how (it) does what (it) does, and structure as what (an object) is. SBF models provide a powerful solution for adaptation problems and for performing case-based and variational design [16], in which old design cases are adapted to address new design challenges. KRITIK [20; 19] is a system which uses the functional representation scheme, called the Environmentally-bound Structure-Behavior-Function (EBSF), to represent and organize knowledge of the functioning of a device, including the role of its environmental interactions.

Other Representation Schema

Mark [35] 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 behavior specifications of modules interrelated by commitment constraints. Developers and software engineers can examine the commitments 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 [35].

Augmenting Design Documentation (ADD) [18] is an integrated computational model for assisting designers in documenting projects at design time. The ADD model represents design rationale as a combination of argumentation-based rationale 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.

More recently, Garcia proposed a system, called ADD+ [17], which includes Rhetorical Structures in active documents. ADD+ uses the same basic model as ADD; however, it improves the system’s interactions with the user. In ADD+, the wealth of knowledge kept in ADD’s knowledge base is organized into high-level Rhetorical Structure Theory (RST) [34] schema and mapped onto input and output screen configurations that gear the interaction between systems and users.

Garza [10] 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 the various design objects in the KRM. The dependency and has-relationship semantic net links of (45) of RSM generates the Parameter Dependency Network, which can determine how the designers arrived at a particular design decision, also can determine how one object-parameter affects other object-parameters and further affect additional object-parameters.

DESIGN RATIONALE CAPTURE

The primary requirement of the design knowledge capture process is that it capture 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 [24]. We can divide the design rationale capture methods into two categories: those that require user-intervention and those that are automatic.

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: what decision designers made, when they are made, who made them, and why [44].

Most of the design rationale systems are using mechanisms that help designers to record design decisions and reasoning during the design process. The REMAP model [40] supports the capture of design rationale knowledge by providing a mechanism for design teams to conduct their deliberations using the primitives (issues, positions and arguments) in the model. As introduced in Section, these primitives are defined manually by the designers during the design process.

The Comet system [35] uses explicit representation and reasoning with commitments to aid the software design. In terms of the rationale capture, it requires the user to introduce new module descriptions as specializations of existing module descriptions, which limits the design flexibility.

In Garcia’s [18] ADD system for HVAC, rationale is captured by using the computer as a “designer’s apprentice”. ADD contains a predefined set of relationships between various design parameters. These relationships allow the system to expect certain values for the design parameters. If a designer proposes a value different from the expected value, the computer asks the designer for justification regarding these differences. The just-
DRIVE (10), which does not have a fixed and predefined set of relationships between its various model parameters, builds relationships between various object-parameters, i.e., as the design develops. It helps designers express the rationale behind their design decisions in a computer-processable format. It also assists engineers in formulating suitable design alternatives by presenting rationale about an existing design. It starts out with a totally empty rationale database and relies on the designers to create relationships among various object-parameters, so it needs more user-intervention than ADD does.

Even though some design rationale systems need user-intervention to record design information, it must naturally support the design process; otherwise, engineers will consider the use of the system as a distraction. Chung’s (8) 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.

### Automatic Rationale Capture

The automatic capture of design rationale assumes there is a method to capture the communication among the designers and design teams or between the designer and the design support system. The communication records can then be used to extract design rationale as they evolve during the design process (44). Usually, communication employs Computer-Supported Collaborative Work Tools (CSCW) (38) or meeting technologies. This includes, for example, telephone, tape recorders, video camera, shared applications and email, to capture oral discussions as well as writings and drawings exchanged between the designers. When communications 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 of form, full of disorder and digressions (44). Raw communication lacks structure, and by using this approach to capture design information the knowledge retrieval process may be ineffective as a result.

HOS (44) provides an environment supporting computer network design with the combination of natural communication and design in an argumentation structure. HOS includes facilities for importing e-mail 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.

PHIDIAS’s (44) 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: 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, then using hyper-links to interconnect items of knowledge, regardless of their level of formality.

Ramesh (41) 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 deliberation and also supports hyper links among design deliberation records and multimedia objects.

JANUS’s (14) architectural design system integrates a CAD-like editor with a rule-based design critic and an argumentation-structured hypertext architecture environment. 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, so the design rationale used during design is captured automatically by the system. The JANUS system has demonstrated that hypertext can be used in conjunction with knowledge-based design environments to ease design knowledge capture.

An experimental system, the Rationale Construction Framework (RCF) (37), was designed to acquire rationale information for 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.

The future trend of design rationale capture is to find non-intrusive methods to capture design information without disrupting the design process, while extracting useful rationale.

### DESIGN RATIONALE RETRIEVAL

The reuse of design rationale is realized by the successful retrieval of it. 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 (1; 3; 4; 33). 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 on 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 (2; 17).

Navigating Archived Design Rationale

Design rationale navigation involves permitting designers to investigate design rationale by traversing from one node to another by existing links. Systems may provide facilities to help with such navigation (13). 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 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.

The VIEWPOINTS (13) system, which uses the PHI representation, is used as a look-up manual where designers can find answers to specific problems and consider various arguments for and against them.

An IBIS-style browser (31) 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 (18) proposed a read-only interface to allow users to navigate graphically through the decisions and reasonings connected to the designed artifact.

COMET (35), the software design support system for Computer-Aided Software Engineering, allows software developers to review and check any existing module descriptions. There is a design memory window with a nodes-links diagram to help with navigation.

In IDIS (8), 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; a user can find out all the issues and rules related to it by visiting the viewpoint and its parents and children. It also provides supplemental facilities to support keeping track of design progress and for reviewing a project when it is completed by answering questions such as “what is new” in the design process, “what are the outstanding issues”, “what has been said about a certain topic”, “how did we get here” and “what are the differences between two design alternatives”.

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 (22) 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 (21).

REMAP/MM (41) uses a deductive query language to define various types of ad-hoc queries, and provides an easy to use graphical interface for displaying queries and retrieving desired information. The queries can be recursive, which supports selective retrieval of process knowledge. It also allows the relevant components of the design process to be replayed.

In KRITIK (7; 39), knowledge access is involved in the process of diagnostics generation, design verification and redesign. The system searches the corresponding Causal Process Description (CPD), and follows states to retrieve the desired knowledge.

Better design rationale representation structure can improve the retrieval greatly. DRIVE (10) can process the design rationale automatically, since 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.

Automatic Triggering

Several design rationale systems enable the capturing of design rationale through automatic triggers or “critics” that detect or monitor certain conditions according to the design context. This type of approach shares many similarities to 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 come out automatically. This type of retrieval of design rationale is especially suitable for use in the design process.

The PHIDIAS (44) 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 by in-
The design rationale is connected to the drawing of the artifact, the critique to the design task or some specific operations on designed artifact. Critics have been built according to the indexing, some of are left active, and thus to be triggered out design rationale according to design status or conditions automatically, some of which must be requested by users to execute.

CRACK (14) has critics as 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 (15).

ADD+ (17) 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.

**Hybrid Retrieval Strategies**

Hybrid Retrieval Strategies provide convenience and efficiency for design rationale retrieval. JANUS (14) 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 (2) 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, it requires prescriptive information to determine the validity of the argumentation stored within KBDS-IBIS, so it is more than just retrieval of what have been recorded.

ADD also allows users to explore 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.

**DISCUSSION**

We have 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.

A design rationale system is not effective as a stand-alone 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 and design reasoning and communication during the design process. Ideally, such a system can transform raw design rationale and design history into knowledge for later re-use—providing facilities to retrieve design rationale when needed for review, maintenance or redesign. The open issues in design rationale are summarized below.

**Representation Issues** 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 (9). 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 jeopardizes the usefulness of the method (8; 18).

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 (13; 32). 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 (22), to explore the possibility of representing more generic clauses in the formal components of the rationales (26), 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 (21).

**Capture Issues** The basic concern is how to capture process knowledge with minimal overhead, with the least interference to the natural progression of design activities (17; 41).

Other problems worth noting include how to resolve the conflicts that arise when new knowledge captured may violate the knowledge previously captured (39). 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. Further work could contribute to better tools for real-time capture and processing of rationale (2). 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 (44) (14).

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 (46) is a way to keep consistency in the capture and construction of design rationale (25).

**Retrieval Issues** Since process-oriented design rationale is organized chronologically, research on retrieval strategies is needed to manage the enormous amount of chronologically organized design rationale (9) and increase human usability and computational tractability (30). 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 (2), (17).

**Approaches Issues** 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) (6).

From our survey, while there have been design rationale systems and prototypes deployed in lab settings, much effort is needed before systems are ready for widespread deployment in working engineering environments. (10).

**Conclusions** This paper has provided a brief survey on recent research in the area of design rationale. In doing so, this work has covered a number of diverse research disciplines, including Mechanical Design, Software Engineering, Artificial Intelligence, Civil Engineering and Human-Factors and Human-Computer Interaction. While this survey has focused primarily on prototype design rationale systems, we have attempted to outline some of the major trends and research issues. In particular, we believe that this work can provide insights as to why design rationale systems have not seen widespread adoption in industry and where the current research issues lay. We hope this study can contribute to our understanding of these important problems and help generate new advances in this field.

**REFERENCES**


A. Goel. A model-based approach to case adaptation. In


Steven E. Poltrock and Johnathan Grudin. CSCW, groupware and workflow: Experiences, state of the art, and future trends. ACM SIGCHI Tutorial, Pittsburgh, PA, April 1999.


