AI Magazine Volume 14 Number 2 (1993) (© AAAI)

The 1992 Workshop on Design Rationale Capture and Use

Jintae Lee

■ The 1992 Workshop on Design Rationale Capture and Use took place on 15 July in San Jose, California. The goal of the workshop was to bring together people interested in design rationale management and promote interaction among them. Participants were selected from different parts of academia (computer science, human-computer interaction, management, civil engineering, mechanical engineering) as well as from industry. This article summarizes the issues that were raised and discussed during the workshop, categorized under these headings: the nature of design rationale, services: what good are design rationales, representation: what information is worth capturing and reusing, production of rationales, semiformal approaches, and future collaboration.

The 1992 Workshop on Design Rationale Capture and Use was held on 15 July in San Jose, California, in conjunction with the National Conference on AI. The workshop was sponsored by the American Association for Artificial Intelligence. Over 40 people from academia, as well as industry, attended the workshop.

In the past few years, we have seen growing interest in *design rationale management*, that is, in representing the deliberations underlying a design process and using these captured rationales to support better design. The foremost goal of the workshop was to bring together the people from various disciplines who are working in this area and to promote interaction among them. The fields represented included AI, software engineering, mechanical engineering, civil engineering, computer-supported work, and human-computer interaction. Workshop organizers hoped that through the workshop, people would become aware of the issues to heed in design rationale management capture, learn what other people have done to resolve these issues, and be able to transfer techniques across disciplines (for example, between software and very large-scale integrated [VLSI] designs).

The workshop began with an introduction and an overview of the general issues. The rest of the workshop was divided into six sessions: (1) Rationale Management in Hardware Design, (2) Rationale Management in Software Design, (3) Constructing Rationales, (4) Rationales in Supporting Coordination/ Integration, (5) Semiformal Approaches to Rationale Management, and (6) Linking Semiformal Approaches to Formal Approaches. The workshop concluded with a wrap-up session, where participants discussed possible future collaborations.

Although some of the discussions in the sessions were specific to the chosen topics, many issues raised were of a general nature and relevant to most of the sessions. Hence, this report is not a chronological account of the workshop but is instead organized by the issues that were raised and answered during the workshop. These issues are grouped into the following categories: the nature of design rationale, services, representation, production of rationales, evaluations, and future collaboration.

Nature of Design Rationale

An answer to the question, What is design rationale? was given by T. Gruber: "an explanation that answers a question about why an artifact is designed as it is." This definition is appealing because it can be used to generate further questions: Who is the audience? When and who generates this explanation? What kind of questions does it answer?

The question of whether the nature of rationale is different in other design domains was raised explicitly by T. Moran but also implicitly in the sessions on software design rationale and hardware design rationale. The goal was to see whether the research in one area is transferable to the other and how much of the research is domain specific. It turned out that most of the issues raised and the discussions held in these two sessions were relevant to both hardware and software designs. For example, it was pointed out by G. Arango that the transformational approaches, most visible in software design, would be suitable equally for other domains, where you can formally specify the requirements, their decomposition, and a set of legal transformations (for example, VLSI design). Therefore, you might say that the form of the rationales can be different in different domains if a domain is defined in terms of the nature of the reasoning involved but not if it is defined in terms of the objects involved.

Services: What Good Are Design Rationales?

It was asked what services become possible with the use of rationales. The proposed answers can be categorized as follows: documentation and retrieval support (A. Garcia), maintenance support (I. Baxter, B. Durney), dependency management (W. Mark), generation of explanations (Gruber, E. Kant, W. Swartout), organizational design (I. Hulthage, L. Gasser), simulation and diagnostics (B. Chandrasekaran), requirement engineering (J. Lee, B. Ramesh, V. Dahr), and design methodology support (D. Bahler, K. Singley, A. Wong).

Some people also pointed out the use of rationales for coordination among designers or integration of different aspects of design. For example, explicitly represented rationales can serve as a basis for communication among designers or for project management, such as tracking unresolved issues and their dependencies or as a way to update newcomers. Rationales can also be used for producing consensus, for example, on type definitions (P. Johnson) or ontology (J. Bradshaw). A model of rationales was also proposed that supports distributed subgoaling as a result of decision making and propagation of the effects of decision revision through a distributed dependency network (C. Petrie). Other issues addressed in this context include how to develop a shared vocabulary for representing rationales for coordination (Bradshaw), use multimedia electronic mail to support effective rationale sharing (J. Glicksman), and integrate the decision making and the artifact aspects of the design process (M. Klein).

It was also asked what questions we want to answer with design rationales, for example, Why was one structure chosen over another? What is the intended function of the artifact? What design decisions are affected by a change in functional requirements, design objectives, or available alternatives? How is the function achieved by behavior? Why does this piece of code exist? Why was it designed the way it was? How did it evolve into its current form?

Thinking about whom we want to support with design rationales makes clear what kind of services we should aim toward. Thus, someone asked a question about who the intended audience is for this research. The captured rationales, for example, might be used to support designers at design time, designers at redesign time, maintainers, troubleshooters, or trainees. If we want to support designers at design time, the ability to manage dependencies becomes important. If it is the designers at redesign time that we want to support, it becomes important to define a similarity metric (for example, requirement overlap) that can help retrieve and reuse relevant parts of the design rationales. Of course, these different requirements create different cost economics, for example, how much and how formally information needs to be captured.

Representation: What Information Is Worth Capturing and Reusing?

The representations proposed at the workshop range from unstructured (for example, electronic notebooks that record rationales in natural language) through semistructured (for example, templates of different types such as requirements, specifications, goals, alternatives, arguments) to completely formal (for example, states, causal connections, annotations).

The differences among these representations reflect the different services that they are designed to support. For example, if the primary goal of managing rationales is to help people archive, retrieve, and examine the reasons for their decisions, the representation can be semistructured; that is, only parts of the representations need to be understood by the computer, and the rest can only be interpretable by humans. However, the more intelligent services we want the computer to provide, such as managing dependencies and suggesting new alternatives, the more formally the domain knowledge has to be represented.

An interesting observation is that despite this diversity, there was some convergence, suggesting that there might be some generic structure of design rationales. For example, in many representations, decision constructs such as alternatives, criteria, and evaluations play important roles. They are often complemented by constructs for representing requirements (for example, requirements, goals, specifications, excludes [requirement, requirement]), representing arguments for evaluations (for example, claim, supports [claim, claim], denies [claim, claim], qualifies [claim, claim]), or representing different parts of what is being designed (for example, artifact, attribute, module, interface, has-attribute).

Production of Rationales

An important question for a design rationale management system is how to produce the rationales needed for the services previously discussed.

At least three major ways were suggested for producing rationales: record and replay, post hoc reconstruction, and generation from domain knowledge. In *record and replay*, the rationales are captured as they unfold, that is, as the designers deliberate over possible alternatives and criteria, answer questions, and so on. For example, people can use a shared database to raise issues, propose new alternatives and criteria, or enter evaluations. The rationales can be captured in unstructured form, in structured representations, or as annotations (B. Reeves, F. Shipman). They can also be captured synchronously (for example, in a capture room, where people use a shared public screen to respond to each other) or asynchronously (for example, electronic mail). This capture methodology usually implies that the representation cannot be too rich or too formal because such a representation would create excessive overhead and disrupt the flow for the designers.

The second way of producing rationales is to reconstruct rationales after the fact. The deliberations are captured in the most unobtrusive form, for example, a video camera. Then, the rationales are recast into the constructs of a particular representation. This methodology has the advantage that reconstruction forces reflections, and the representation can be more formal and more systematic. However, reconstruction can be a luxury that many people cannot afford.

The third way of producing rationales is to generate rationales from formally represented domain knowledge. Some of the suggested methods for constructing rationales were simulation (Gruber), context monitoring (R. Cohen), reverse engineering (G. Kim), product modeling (Arango), interactive verification (Garcia), rule-based construction of composition hierarchy (B. Britt), and a domain-independent decision-revision theory encapsulating domain-specific informal objects (Petrie). This strategy has the high initial cost of compiling the knowledge needed to construct the rationales but has the appeal of creating rationales at no cost to the user later and being able to maintain consistent and up-to-date rationales.

The following technologies were suggested as appropriate for capturing and accessing rationales (Moran): scribbling tools, audio-video, groupware, argument representation, hypertext, expert systems, document management, and design tools. The following elements were also suggested as possible candidates for constructing rationales (Gruber): designers at design time, designers at documentation time, designers' support tools, and programs at reengineering time.

Semiformal Approaches

The first thing that would probably strike anybody looking at the literature is the division between formal and informal approaches to design rationale. This division cuts across all the aspects that have been discussed so far: service, representation, and production of rationales. It has been pointed out that formal representations have the advantage of being interpretable by computers and having well-established inference procedures, but they are hard to create and comprehend. Also, the domain knowledge needed to create formal representations is often missing. However, informal representations are easy to create and natural, but they are not interpretable by computers and rely on unarticulated background knowledge (G. Fischer).

Semiformal approaches take the middle ground by having some parts of the representation interpretable by the computer but others only by human users. For example, task-specific objects (for example, decision, goal, claim) can be formal objects with their own attributes and can formally be related, but the system might allow these attribute values to be filled in by designers in the form of informal descriptions or other formal objects that the designers might choose to create. The system then processes the descriptions to the extent that they are formalized but leaves others for human processing. The appeal of this approach is that there is relatively less overhead in capture (in fact, semiformal representation can be easier to deal with than informal representations by suggesting expected information and defaults), yet computational operations exploiting the formalized part of the representation can be defined. The degree of formalization can also be controlled by different representations.

The following techniques were also proposed as ways of linking more formal representations with semiformal representations so that we can define more automated design support as we understand more of the domain: the creation of rules from the keywords assigned to different parts of the rationales (K. Nakakoji), a database of requirements and issues that motivate and help people to incrementally formalize domain knowledge (Lee), a programmable design environment combining knowledge-rich cooperative problem-solving systems and programmable applications (M. Eisenberg), a representation combining hypertext and knowledge representation (D. Barman, F. Lakin), the evolving artifact approach (J. Ostwald), or the extension of semiformal representation with domain-specific formal representation (L. Ruecker). Some of the questions asked for all these approaches were (Fischer), What are the roles of the human and the computer in these systems? How are the two linked? How do they communicate? How does the distribution between human and system change over time? Is there a shared knowledge between human and system? If so, what kind? How does it get created? How does it change over time?

Future Collaboration

In the wrap-up session, the participants discussed ways to continue to work together on design rationale. One idea suggested was the use of a set of canonical examples. Detailed records of a few design processes (in video or in transcripts) would provide the researchers with the data. Also, they would provide common examples and vocabulary that should facilitate comparison and communication among the different models underlying the design rationale research. We agreed to evaluate the examples that have been used for similar purposes in other areas (for example, design of a lift system) as well as other examples that data have already been collected for (for example, automated teller machine or heating, ventilating, and air conditioning system design). Another related idea for facilitating future collaboration was to set up a clearinghouse, not only for the canonical examples but for other resources as well. For example, we can also share task-specific ontologies, domain knowledge, and even inferencing modules. We decided that this idea would be pursued in the context of the ongoing attempts to share knowledge.

Many issues were raised and discussed at the workshop other than those listed here. For example, it was pointed out and agreed that the success of a rationale management system critically depends on the successful resolution of nontechnical issues, such as integrating such a tool with standard operating systems and tying proper incentives and resource allocation to the use and refinement of parts, for example, software modules (Dahr). The limited space, however, prevents us from expanding on these discussions here.[1]

Acknowledgments

Thank you to all the people who participated in the workshop and made it exciting. The program committee members, each of whom put in a lot of effort reviewing the papers and moderating the sessions, deserve special thanks. Thank you to Tom Gruber, Bob Halperin, Charles Petrie, and Tom Moran for their comments on this report. I also appreciate the support from AAAI as well as the Center for Coordination Science and the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology.

Note

1. The papers presented at the workshop were compiled in the form of notes. Requests for further information or comments can be addressed to the chair of the workshop.

Jintae Lee is an assistant professor in the Information and Computer Sciences Department at the University of Hawaii at Manoa. His interests include rationale managment and its use and reuse for organizational design and simulation.