6 Presentation of Rationale

This chapter examines issues of presentation for software engineering rationale (SER). The substance, the content of rationale, is always mediated by some presentation. The presentation could be free form, natural language text, or a formal, symbolic language; it could be printed sheets of paper, or three-dimensional displays in a virtual environment. The presentation of rationale has its own effects on the utility of rationale as an information resource in software development.

6.1 Introduction

6.1.1 General

The ingenuity and effort of creating a sound and comprehensive rationale is only worthwhile if people can use it. The use of rationale is always mediated by its *presentation*. The presentation of a rationale can be relatively formal and symbolic, for example, using types and logic with labeled links, or it can be relatively informal, such as free text or even a videotaped interview with a designer explaining his or her design. Various approaches to presenting rationale themselves have rationales. One significant advantage for Software Engineering Rationale (SER) presentation is that, unlike hardware devices, the designed artifacts themselves are stored electronically. This supports the potential to attach the rationale for the artifact directly to the artifact in ways that are impossible in other design domains.

6.1.2 Objectives of This Chapter

This chapter describes the main line of development of the IBIS (for Issue-Based Information System) notation for rationales, from the early innovations of Kunz and Rittel (1970), through work on hypertext and hypermedia rendering of IBIS graphs, through to studies of the use of IBIS and IBIS-derived approaches to presenting rationale. One of the key issues that emerges from this line of research is that there is a tradeoff between the discipline and clarity that one obtains from casting a design discussion into as IBIS presentation, and the inflexibility and cumbersome aspects of working with IBIS. In part, these tradeoffs led to a turn toward informal presentations of rationale in the mid-1990s and subsequently. Today, reconciling these approaches, and enhancing them through new techniques in information visualization, seems feasible, and perhaps even more necessary as the role of the software developer expands to include end users.

6.2 Codifying Rationale Semiformally

6.2.1 The rationale for rationale notations

Discussions of rationale quite appropriately tend to start with Kunz and Rittel's (1970) concept of Issue-Based Information Systems (IBIS). IBIS presents rationale as a structured discourse of *arguments* that support or oppose *positions* that themselves correspond to *issues*. This results in a straightforward and explicit relational decomposition of issues, positions, and arguments. However, IBIS quickly gets more complex: arguments can support or oppose *other arguments* as well as positions, and in particular, a given argument can support/oppose arguments that pertain to other positions on other issues. Issues have many interrelationships; one issue can *illustrate* another issue, *generalize* another issue, *resolve* another issue, etc. Thus, the hierarchy of issues, positions, and arguments is actually a network.

The key insight of IBIS can be regarded as essentially presentational: Kunz and Rittel emphasized that in planning and design problem solving the key ideas, the "solutions", were often "there" in plain view, but not always identified, weighed, and valued appropriately. IBIS makes explicit how the elements of a complex problem solving process interrelate. It presents the underlying argumentation as a graph of related propositions so that planners and designers can have more precise discussions.

An IBIS presentation of the status of a design rationale makes public what issues are currently identified and how they are related. This can focus disagreements and discussions and make them more productive. For example, a debate about what the positions are with respect to an issue is very different from that of how various arguments support or oppose a set of positions. It is efficient to distinguish between these two sorts of debates, among others.

An IBIS presentation of a design process can also be a generative tool. Laying out the network of currently identified issues, positions, and arguments, helps to suggest further issues that need to be raised, or further relations among issues already identified; it makes clear what positions have been identified for each issue, perhaps suggesting positions that still need to be articulated. Setting out the arguments for every position shows which positions are better supported than others, suggesting where attention can be focused to strengthen and/or eliminate some of the current positions.

Thus, an IBIS presentation both explicitly codifies the current state of a problem solving process, and poses a detailed agenda for further discussion and action. It seeks to improve the outcome of deliberative processes by highlighting divergence and even controversy. It gathers and integrates the knowledge distributed among members of a planning or design project, organizing the knowledge with respect to its relevance to the project. It makes the bases of eventual decisions more transparent and auditable.

The network presentation of rationale, first developed as IBIS, has become a standard visualization for subsequent rationale projects—even those that construe the content of rationale in ways different from IBIS. For example, Questions, Options, and Criteria (QOC) is a variant of IBIS that seeks to document a design *solution*, as opposed to the discussion *process* that led to the solution (MacLean et al. 1989). Thus, where Kunz and Rittel (1970) wanted to capture and present the actual issues, positions, and alternatives as they were discussed in a design process, including parts that ultimately had no tangible impact on the final design solution, QOC seeks to present only the design argumentation that justifies the design solution. MacLean, Young and Moran (1989) saw QOC rationales themselves as a form of designed documentation for a design solution. Nevertheless, QOC rationales are typically presented in graphs that are isomorphic to IBIS graphs: design questions (essentially, IBIS issues), the options that address them (essentially, IBIS positions), and the criteria for assessing options (essentially, IBIS arguments).

6.2.2 Hypermedia Presentations of Rationale

IBIS was originally conceived as a paper-based information technology. However, as IBIS argument networks get larger and more complex, they become very difficult to read and edit in paper: they are too large for standard-sized sheets of paper, and as they change and grow, pages become cluttered with crossing lines, erasures, and annotations, and purely paper representations are not convenient to save, and very difficult to share with remote collaborators or to adapt and reuse in subsequent projects.

IBIS has been incorporated into design *war-room* practices in which a design problem is analyzed and managed through paper-and-string representations pinned to the walls of a workroom (Newman and Landay 2000; Whittaker and Schwarz 1995). Wall-sized pin-up representations are large enough to display nontrivial IBIS graphs, and, relative to paper, they are easily edited. However, rooms are expensive and cumbersome in their own ways as representational media: they cannot be saved for subsequent reference or reuse, and they cannot be shared with remote collaborators.

The advent of hypertext and hypermedia in the mid-1980s provided a breakthrough in the presentation of IBIS rationales. Conklin and Begeman (1988) described graphical IBIS (gIBIS), a browsing and editing tool for navigating and managing vast rationale networks. This tool provided many of the navigation and maintenance affordances of a wall-sized pin-up display, but rendered them accessible through a workstation user interface. This made possible saving, sharing, and reusing IBIS graphs.

Many hypermedia and hypertext tools for presenting rationale have been developed. For example, McKerlie and MacLean (1993) prototyped a hypermedia QOC rationale browser that incorporated documents, diagrams, images, and other media types directly into the nodes of a QOC graph.

6.2.3 Using Semiformal Rationales

Semiformal rationales lie in the gray area between notations with known properties and free-form expressions of rationale. Through the nearly 40 years of experience with IBIS and its descendants, there has always been a tension between beliefs that the discipline of categories and links could help to focus design thinking and beliefs that the notation could be an awkward distraction from the substance of design thinking. Indeed, Conklin and Begeman (1988) reported both patterns among their early users.

One of the benefits of semiformal notations is that they project a template structure onto design argumentation, highlighting gaps, and thereby helping to further articulate requirements. Because gIBIS was actually implemented and used (albeit mainly in research laboratory software development projects), it helped to identify some of the second-order challenges for rationale browsers—challenges that could only become apparent through the real use of rationale presentation tools.

Conklin and Begeman (1988) noted, for example, that the use of gIBIS helped to identify some specific problems having to do with the fact the IBIS does not represent design decisions per se. Decisions are critical events in design discussions; they resolve sets of positions on an issue, selecting one position and rejecting the others. The chosen positions are often embodied as a solution element (e.g., a specific piece of code). Conklin and Begeman (1988) observed that users had to keep track of design decisions and their associated solution elements *outside* the gIBIS system.

Conklin and Begeman considered indicating selected positions through display highlighting, to distinguish them visually from the rejected positions. However, one deficiency of this approach is that the rationale for the *decision itself*—as distinct from the rationale for the position as a response to a given issue—cannot be represented. A more comprehensive approach, also discussed by Conklin and Begeman (1988), is to create a separate layer of meta-argumentation for discussion about nodes and groups of nodes in an IBIS graph. This approach obviously adds a great deal more complexity.

In the early 1990s, influential empirical studies of the use of semiformal rationales presented through hypermedia browsers identified substantial cognitive and social obstacles (Buckingham Shum and Hammond 1994). Indeed, these specific studies were assimilated to a more general critique of efforts to support intellectual work directly with formal and semiformal knowledge representations (Grudin 1994; Shipman and Marshall 1999a). Recent work on semiformal rationales presented through hypermedia browsers has focused on providing a richer vocabulary of categories and data types, and more flexible user interactions (Buckingham Shum et al. 2006.).

6.3 Codifying Rationale Informally

The tradition of rationale presentations inaugurated by IBIS focused on constrained symbolic descriptions. This was intended to benefit analysts and designers by providing a relatively precise description language as well as a discipline for using the language. However, for the most part this is more of an intention, a vision of what rationale could be, rather than an achievement *tout court*.

The semiformal notations, such as the standard IBIS graphs, do not actually provide very much descriptive constraint, and to the extent they do provide constraint—as in the example of including no category for decisions, the constraint were sometimes found to be inappropriate.

Nevertheless, pursuing even a programmatic interest in constrained descriptions is different, eschewing such concerns. Starting in the mid-1990s, less formal approaches to rationale became more common.

Many of these less formal approaches to rationale were part of a concurrent rethinking of software design, and a turn toward less formal approaches to specifications and other software design representations (Carroll 1995; Fowler 2003). A central characteristic of these approaches was (1) a focus on *narrative*: stories of workflows and other organizational processes, scenarios of user interaction, and use cases of system interactions, and (2) a deliberate compromise of semantic precision for conceptual richness. Thus, where IBIS tried to impose (albeit programmatically) conceptual austerity on planning and design—the most "wicked" of problem types, in Rittel's famous term—these latter approaches took the more naturalistic stance of confronting the wickedness first.

Scenario-Claims Analysis (SCA) conceptualizes the rationale for interactive software systems as a collection of natural language propositions (claims) that are implicit in the usage scenarios afforded by the system (Carroll and Rosson 1992; Carroll 2000). The propositions are used to identify tradeoffs in the rationale for the system. Consider a simple scenario in which a person is trying to copy text using an information system that grays out currently inappropriate/disabled menu items. Going to the Edit menu *before* selecting the text to be copied, the person finds Copy grayed out, but after selecting the text, the Copy command is no longer grayed out, and the operation can be completed. This scenario illustrates a claim that graying out is an effective visual signal for currently inappropriate/disabled commands. This claim also helps identify potential tradeoffs, downsides of the graying-out technique; for example, the user might not make the right interpretation; the grayed-out command might just seem to be broken in the software, instead of suggesting that its argument needs to be specified.

SCA rationales are usually presented in tables, not as IBIS graphs, but in fact there is an obvious, though perhaps rough, mapping between the two: each scenario in SCA presents an *issue*, or possibly a nexus of related issues. The design artifacts described in the scenario (such as the graying-out technique) are *positions* that respond to the issue or issues, and the claim tradeoffs are *arguments* for and against these positions. Of course there are also differences: a user interaction scenario is both more complex and more narrow than an IBIS issue. For example, scenarios often present more than one issue, and generally illustrate only a single position for a given issue, not a range of possible positions responding in various ways to the issue. A similar comparison can be carried out for other scenario-based approaches such as Lewis, Reimann, and Bell's (1996) problem-based evaluation approach in which a set of problem scenarios are identified, each presenting one or more issues, and then used to analytically evaluate a set of design proposals (positions) via an informal walkthrough (producing a set of arguments for the positions).

Other contemporaneous efforts at naturalistic capture and presentation of rationales explored narrative frameworks that were even less schematic than scenarios. Some of this work captured ethnographic design history material. For example, the Raison d'Etre project captured and presented the individual rationales and understanding of project members at specific points in time during a software development project. A dozen core members of a software product design team were recurrently interviewed during a 12-month period. The developers were individually asked about the goals and approaches of the project. A video database of about a thousand short clips was created (Carroll 2000; Carroll et al. 1994). The video clips could be browsed and retrieved using a set of tags (e.g., <project vision>).

This project showed that there is an abundance of rationale generated every day in software development. However, it also showed that there is only a partial convergence and consensus as to why decisions were taken, or even about what decision were taken. Developers were very interested to review and discuss the database of interview clips, but the most practical application of the Raison d'Etre materials was to help new project members get better oriented to the issues that the project had faced, the diversity of positions that had been taken, and arguments that had been advanced for those positions.

Mackay, Ratzer, and Janecek (2000) also employed video to capture and present design requirements, concepts, and rationales. Their approach focused on documenting a system in use by videotaping both expert users and novices in actual work contexts. They also videotaped design meetings in which new design proposals were described and critiqued. Finally, they used these real materials to plan and construct animated storyboard scenarios showing how particular design proposals might be implemented and how they might change the system in use.

6.4 Directions

The original challenge in presenting rationales was the complexity and vastness of the considerations that can bear on wicked problems of planning and design. The IBIS notation brought an order to research on this challenge, but the challenge remains. Today, software technology advances in databases, and more generally in information repositories, and in data visualization present new opportunities for developments in managing rationale.

6.4.1 Reusable Rationale Databases

Since the early 1990s, papers on design rationale have suggested the possibility of repositories or libraries of rationale. Indeed, one argument that could be made for semiformal design rationale notations is that they provide a rubric for structuring and retrieving rationale elements in such repositories. Such repositories could improve the cost–benefit balance for developing rationales in three distinct ways: they amortize the costs of developing comprehensive design rationales by permitting many authors to contribute rationale, they could improve the validity and applicability of rationales by moving the level of design discourse beyond single projects and into the entire software design community, and they could increase the benefits of developing rationales by allowing many developers to access and use rationales once they are created.

Sutcliffe and Carroll (1999) defined a structural schema for claims to facilitate claim retrieval and reuse. Their schema includes a series of labeled slots for each claim, including parent claims, projected usage scenarios, design effects, upsides, downsides, issues, dependencies, evaluation data, and basis in theory. Developers could search or browse a claims repository using the values of these slots. Chewar et al. (2005) adopted this proposal and developed a rationale repository to support the design of notification systems (interactive interface displays like Really Simple Syndication (RSS) clients that run in background of a primary task and notify users of updates). Their Leveraging Integrated Notification Knowledge with Usability Parameters (LINK-UP) system presents claims for typical notification system scenarios. On-going evaluation of the use of LINK-UP by novice designers has been encouraging (see also Fabian et al. 2006; Payne et al. 2003).

The Software Engineering Using RATionale (SEURAT) system (Burge and Brown 2004) uses the RATSpeak representation (Burge and Brown 2003) implemented as a reusable rationale database schema. When rationale is required for a new project, the initial rationale-base is populated with the required schema tables and a fully populated Argument Ontology that contains a hierarchy of reasons for making software decisions. SEURAT has only been used as a single-user system. The relational database would make it straightforward for multiple users to contribute rationale but there are other SEURAT capabilities, such as the ability to associate that rationale with the code, that cannot be distributed using the current implementation.

6.4.2 Multi-Scale Presentations of Rationale

All of the standard presentations of rationale articulate a great amount of structure at basically a single level. This is obvious in the vast networks that gIBIS tried to manage through hypermedia browsing. However, in some ways this does not reflect the structure of a rationale space as designers and users experience it. Some issues, positions, and arguments are first-order elements of the design argument; others are subordinate. However, these relations are not necessarily clear or even codified at all in standard IBIS graphs.

This could be seen as an example of multiscale data structures. For example, in a map of the world the continents and oceans are always visible, but the Hudson River may or may not be visible at that scale. However, in a map of the state of New York, the Hudson River is always visible, but the individual streets in the town of Ossining (located on the river) would most likely not be visible, though they would be on a map of Ossining or of Westchester County. The point is that map data is understood to be multi-scale data, and is typically presented in multiscale presentations.

Analogously, rationale data might be organized so that the coarsest scale would present only the leading issues, positions, and arguments. However, one could drill down to finer scales to see the subordinate issues, positions, and arguments. The multiscale concept is most typically discussed with respect to visualization techniques, as illustrated by maps. Perhaps its application to presentations of rationale should be pursued especially with respect to visualizations of design argumentation (e.g., Kirschner et al. 2003).

Wahid et al. (2004) describe a simple but concrete example from their claims repository work: they visualize a central claim as surrounded by concentric orbits of supporting or otherwise related claims. The user can filter the visualization to see only the core claim, or to see only the core claim with its most related claims, or to see the maximum map of related claims.

6.4.3 Integrated Presentation

As stated earlier, the capture and use of rationale for software development has a significant advantage over rationale for other domains. Since software is stored entirely electronically, the rationale can be attached directly to the artifacts that it describes. This is aided significantly by progress in software development environments that have emphasized the ability to integrate and extend the various tools used in developing software. These tools include word processors used to write and access documentation, UML editing tools used in design, and the Interactive Development Environments used to write, edit, compile, and debug the code. The extensibility of software development environments has also benefited from the increasing availability and use of open-source applications in these environments, which provide even more flexibility and openness in customizing the environment to support and accommodate rationale.

One of the issues in the capture and use of rationale is the need for developers to record and use their rationale as part of their normal development process. The need to have to use a separate tool for rationale has been a deterrent toward doing this. When examining past scenarios where rationale could have been beneficial in saving time or money, one question arises: would the person who could have benefited from the rationale have actually looked at it? Would they have even known that it existed? While rationale does have some benefit as a generative tool, it should not be treated as "write-only" documentation.

Software design is often documented using the Unified Modeling Language (UML). Zhu and Gorton (2007) developed a UML profile that models design decisions in UML and captures the relationships (support, break, help, hurt) between the decisions and nonfunctional requirements (NFRs). UML stereotypes were used to model each of these elements. The design decision stereotype describes the decision, design rules applying to the system components, design constraints, the set of architectural elements (such as UML classes) the decision refers to, and the rationale (in an unspecified format). The NFR stereotype gives attributes specific to that NFR, and the relationship stereotype describes any constraints that apply to that relationship. The profile supports consistency checking between design decisions and related architectural elements.

When building a Rationale Management System, one issue that must be addressed is how and when the developer should be informed that there is rationale available. Systems working in domains that are more constrained than software, such as the JANUS system (Fischer et al. 1989) which supported kitchen design, served as critics that presented rationale when the designers' actions appeared to contradict rules embedded in the system. The user is informed of the presence of rationale when they make a decision that appears to be incorrect. Rationale is also used interactively within a design environment in the Representation and Maintenance of Process Knowledge (REMAP) system (Ramesh and Dhar 1992) where the rationale behind the functional specification is used to help make design choices.

While rationale can be used prescriptively to assist with designing, it is also valuable when used descriptively by providing insight into why the system is implemented the way it is. The user is more likely to be aware of, and read, the rationale behind the code if the rationale is integrated either directly into the code that they are modifying or the environment that they are modifying it with. The SEURAT system (Burge and Brown 2004) integrated rationale capture and presentation into the Eclipse (www.eclipse.com) development framework. The rationale argumentation structure was displayed in a tree format within an Eclipse "view." In addition, three standard Eclipse views were extended/used to show the presence of rationale: the Java Package Explorer was augmented by an icon overlay on every file that had associated rationale, rationale associations were stored as Eclipse "bookmarks", and each bookmark giving an association was shown in the editor used to modify code. The bookmarks could be used to jump directly from the rationale alternative to the code that implemented it. The goal behind the integration was to reduce the likelihood of a developer or maintainer working with code while oblivious to the presence of the rationale that could assist them.

6.5 Summary and Conclusions

Probably the two greatest innovations in presenting rationale are still the original information schema of IBIS and the gIBIS hypermedia browser for IBIS graphs. Some of the dichotomies that have structured research and development on rationale presentations during the past several decades have dissolved. For example, the distinction between semiformal notations and informal notations seemed paradigmatic in the early 1990s, but will probably matter less as information systems increasingly create structure out of content, and thus do not need to force structural constraints on the humans that use them. Thus, the presentation of rationale—how it appears to its human users—will tend to matter more in the future.