Requirements for a Design Rationale Capture Tool to Support NASA's Complex Systems

Becky L. Hooey
San Jose State University Research Foundation
at NASA Ames Research Center
Moffett Field, CA 94035
bhooey@mail.arc.nasa.gov; 650-604-2399

David C. Foyle
NASA Ames Research Center
Moffett Field, CA 94035

ABSTRACT
An understanding of the design rationale, or the justification for design decisions made throughout the design process, is necessary in order to understand, recreate, or modify a design. However, this information is rarely captured in a systematic and usable format because there are no tools that adequately facilitate and support the capture of these critical decisions. This paper summarizes the requirements for a design rationale capture tool that supports the capture and retrieval of relevant design knowledge throughout three phases of the design process: Conceptual Design, Design Implementation, and Design Evaluation and Transfer.

INTRODUCTION
A record of the design process and the decisions that were made is necessary to be able to understand, recreate, and modify a design; however without an understanding of the design rationale, it is not sufficient. Design rationale includes the reasons behind a design decision, the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision (Lee, 1997). Too often successful system development projects fail to leave a legacy of this design rationale information beyond the physical descriptions of the system. Important information about *why* the system was designed a certain way, or what design options were considered but rejected, is rarely adequately captured. The underlying intent (i.e., the rationale) for the included decisions is usually lost. Often this information is scattered throughout a collection of paper documents, project and personal notebook entries, and the memory of the designers (Klein, 1993). This makes the design rationale information very difficult to access and use, such that this design knowledge often "goes with the employee". In this era of budget constraints, and the long-term, dynamic nature of NASA's Vision for Space Exploration, NASA is at risk to lose large amounts of design knowledge vital for maintenance, training, operations, and design modifications. The need for a design rationale capture tool is prevalent in a wide array of NASA's design projects including Constellation, small satellites, air traffic control automation, and robotics. A particularly relevant example of the need to capture design rationale information is currently being experienced in NASA's Constellation Project. If design rationale could have been captured in an efficient and effective manner during the Apollo Era in the 1960's, NASA would be able to take advantage of this design knowledge and more easily apply these lessons learned to Constellation.

Existing Knowledge Capture Tools
Existing archive tools do not adequately support capture of critical design decisions and rationale

or the subsequent search and retrieval of such data. Most document archive tools or repositories merely store documents in a chronological fashion and are inadequate as design rationale capture tools because: 1) they make no attempt to ensure that the captured information is appropriate and sufficient for designers to understand, replicate, or modify the design; and, 2) they do not allow for iterations – that is, if a design element or system requirement is modified, there is no easy way to propagate that change throughout the body of design knowledge, and to understand the implications and consequences of those changes.

Recent efforts to develop design rationale capture tools attest to the value of comprehensive documentation of design rationale (see Carroll & Moran, 1991; Myers, Zumel, & Garcia, 1999); however, detailed design rationale is still rarely produced in practice because of the substantial time investment required. These tools that support design rationale capture demand substantial designer time to enter information (Carroll & Moran, 1991) and often change the manner in which designers work (Conklin & Yakemovic, 1991). A review of the literature suggests that efforts to support the acquisition of rationale have focused on languages and tools for structuring the acquisition process (Myers, Zumel, & Garcia, 1999), but less so on the actual needs and workflow processes of the designer. The result has been systems that are time consuming and burdensome, and offer little motivation for a designer to participate in documentation activities

Most current design knowledge capture systems function to either communicate or document design knowledge. Systems that are intended for communication capture all communications among team members during design meetings (e.g., the electronic cocktail napkin; Gross, 1996). These systems merely record thought that has occurred, rather than attempt to shape the process, and do not encourage the communication of high-level design decisions such as assumptions, constraints, and design philosophies. Also, these systems tend to lack the structure necessary to enable efficient retrieval of the information by future users of the knowledge. Systems that are intended to document design decisions (e.g., computerized lab notebooks; Gorry et al., 1991) are usually used for the purpose of enabling people outside the project group to understand, supervise, and regulate what is done by the team, or to secure intellectual property rights generated by the design team (Shipman & McCall, 1997). These systems are inadequate to capture the information necessary for designers to understand, replicate, or modify the design of a complex system. For example, reasons why a particular design was NOT chosen or implemented are rarely documented in such systems. As Leveson (2001) points out, specifications rarely state what a system should not do, and this incomplete specification has been a factor in aviation and space-mission accidents.

The Complexities of NASA’s Design Problems

Existing design knowledge capture tools are inadequate to handle the complex systems typically developed in support of NASA’s missions. Many of NASA’s complex engineering problems can be characterized as ‘wicked’ design problems (Rittel, 1984). These are design problems that possess a number of distinctive properties that elude design methods that assume that the problem is already understood sufficiently for it to be analyzed using automatic tools, or top-down methods. Specifically, Rittel defined wicked problems as those that:

- Cannot be easily defined such that all stakeholders agree on the problem to solve
- Have no clear stopping rules
- Have better or worse solutions, (not right and wrong ones)
- Have no objective measure of success
- Require iteration

- Have no given alternative solution (they must be discovered)
- Require complex judgments about the level of abstraction at which to define the problem
- Often have strong moral, political, or professional dimensions that cannot be easily formalized

These complex and long-term design processes involve a series of iterative design and evaluation cycles in which the problem and questions are first defined and agreed upon. Subsequently, initial designs, based on known data, design principles, assumptions, constraints, and design philosophies, are evaluated using empirical techniques such as modeling and simulation. The empirical results are then used to verify or modify the initial design. This iterative design cycle produces a wealth of data and knowledge that forms the rationale for the final design.

REQUIREMENTS FOR A DESIGN RATIONALE CAPTURE TOOL FOR NASA'S COMPLEX SYSTEMS

What is needed is an efficient mechanism to document and manage the collection of this complex design knowledge in real time, as an integral part of the design process, so it can be synthesized in a useful manner. The first step in the development of an effective design rationale capture system to meet NASA's needs is the development of system requirements. To ensure that the final system will be useful (and used) by designers of NASA's complex systems, it is mandatory that the end-users (designers) be involved in the development process. In order to determine the requirements for the design rationale capture tool a number of methods and data sources were utilized including:

- Structured interviews with experienced designers from NASA and industry
- Review of designers' log books from previous design projects
- Review of designers' email communications from previous design projects
- Review of design processes from different domains (e.g., Team-X, Deutsch & Nichols, 2000; Human-Centered Design Process, Hooey, Foyle, & Andre, 2002; Usability Engineering, Rosson & Carroll, 2002)

This process revealed uses for a design rationale capture tool across three phases of the design process:

1. Conceptual Design
2. Design Implementation
3. Design Evaluation and Transfer

The phases of the design cycle are sequential in that Conceptual Design phase feeds into the Design Implementation phase, which then feeds into the Design Evaluation and Transfer phase. These design stages are also iterative, in that often the Design Implementation phase feeds back knowledge to modify the Conceptual Design, and similarly, results of the Design Evaluation and Transfer phase impact the Design Implementation process and outcome. Requirements necessary to ensure that the design rationale capture tool meets the needs of each design phase follow.

1. Conceptual Design

The Conceptual Design phase includes both system definition and preliminary design activities. In the Conceptual Design phase of any system, product, or interface, designers set out to understand and specify the context of use, the environment, the missions, and the tasks that the operator will conduct. This phase also typically includes developing design goals or a design philosophy. These high-level concepts often go unstated, or are forgotten – yet are critical for understanding a design. One reason that design goals and design philosophies are frequently not captured is because they are often difficult to formulate and communicate in the early stages of a design project. The design rationale capture tool can play a role in the development, documentation, and dissemination of this high level design knowledge not only by capturing it, but also by helping to shape it.

In the conceptual design phase, designers typically assert assumptions and constraints that will impact future design efforts. Assumptions are made about the end-user of the system, user goals, and how the system will be used. Constraints that may limit design options may include environmental, physical, technical, cost, time, or human information processing factors. It is important that these be captured as in many cases the design philosophy, assumptions, and constraints are the rationale for a given design element. For example, the minimum volume for the Crew Exploration Vehicle (CEV) cockpit is based on an assumption of a specific crew size.

Once the mission is understood, applicable design requirements and standards are identified (or created). For example, a requirement found in NASA's Man-System Integration Standards for hardware and equipment reads as follows: "Hand gripping surfaces that minimize abrasion to the EVA glove material shall be provided on handles of tools" (NASA, 1995). Documenting these requirements can prevent feature creep, or having unnecessary design elements that do not meet the need of the user. Feature creep can occur because it is often difficult to 'see' the underlying design philosophy. For example, a design philosophy for an avionics cockpit display might be to provide situation awareness information in a status-at-a-glance format to minimize pilot head-down time. If a designer, unaware of this underlying philosophy, opts to provide precision control information by which the pilot is expected to steer the aircraft, this would directly violate the intended usage and design philosophy. This problem can be reduced by a design rationale capture system that makes these high-level goals and philosophies more visible to the designers -- particularly if the design rationale capture tool requires that each design decision be linked to one of the defined goals, assumptions, constraints, or requirements.

Requirements to Support Conceptual Design:

- Guide the user to develop and document design goals and philosophy, assumptions, constraints, and requirements
- Increase visibility and accessibility of high-level goals, philosophy, assumptions, constraints, and requirements
- Link each design decision to a goal, assumption, constraint, or requirement to guard against feature creep

2. Design Implementation

In the design implementation phase, prototypes are often developed, evaluated, and refined based on empirical data (which form the basis for the design rationale). This design phase often is collaborative with subject matter experts from many domains working together toward a final design. It is often characterized by group brain-storming sessions and often decisions are revisited iteratively as new data are available to support or refute these decisions. A tool must support the management of these iterative, collaborative design sessions.

To support this, it may be necessary to embed commonly used design tools, standards, and guidelines within the design rationale capture tool. For example, by providing access to relevant standards, guidelines, and reference materials via a shared tool, it becomes much easier to link each design decision to the standard or guideline that contributed to the decision. Related, the output of a typical design process includes a variety of products including emails, websites, text documents, spreadsheets, flowcharts, sketches, prototypes, blueprints, pictures, video, audio clips, CAD files, and other documents describing the final result of a long series of discussions and tradeoffs by the members of the design team. The design rationale capture system must be flexible enough to support the collection of these varied sources, yet structured enough to allow for efficient search capabilities. Multiple easy-to-use input methods such as drag and drop, digital audio, electronic whiteboards are important for supporting data capture without excessively burdening the designer.

Enable both individual and collaborative design. Design can be both an individual and a collaborative activity, thus a design rationale capture tool must support both methods of design by allowing users to create and track private design ideas as well as to share and communicate with other designers.

In support of individual design is the notion of private, password-protected, sandboxes where designers can archive and iterate on their own design ideas before sharing them or integrating with the design team. An important reason for this requirement revolves around intellectual property concerns with designers' desire to maintain ownership of design ideas that might not make it into the final design of the particular project, but could be re-used elsewhere.

At the same time, there is also a need for the tool to support the representation of multiple stakeholders' viewpoints, including end-users and members of the design team with different backgrounds and expertise (i.e., software, hardware, human factors). Shum, MacLean, Bellotti, & Hammond, (1997) refer to this as a 'dialectic collaborative' model, that is, the knowledge invested in a particular project is the product of more than one individual, and often beyond any individual's grasp. According to this view, the processes of articulating and reconciling different perspectives are central to design, and should be recognized and supported. This can be supported by the incorporation of features such as web-based collaboration tools (i.e., wikis) that allow designers to easily upload their design ideas and prototypes for feedback and comment, and to provide input for fellow designers. This also implies the use of web-based, platform-independent structures that can be easily accessed from remote locations by all participants.

Support iterative design. As discussed above, NASA's complex and long-term design processes typically involve a series of iterative design and evaluation cycles in which the problem and questions are first defined and agreed upon. Subsequently, initial designs, based on known data, design principles, assumptions, constraints, and design philosophies, are evaluated using empirical techniques such as modeling and simulation. The empirical results are then used to verify or modify the initial design. This iterative design cycle produces a wealth of data and knowledge that forms the rationale for the final design. For these design problems, framing and reframing the problem is an important process: Reformulating views places a strong requirement on a design capture tool – not only must they support restructuring problems, but must also help designers to identify the need to restructure in the first place. For example, a design capture tool, if not designed carefully, may convey a deceptive 'completeness', and as such may impose views of the design space that may carry excessive weight in shaping the final design.

Support data-driven design. Initial design decisions may be based on existing knowledge, literature, and experience with previous systems, however, in NASA's complex systems, many design decisions must be determined empirically and compared iteratively. This is normally done in the form of simulations, field tests, system modeling, and human performance modeling. These research efforts produce a wealth of data, sometime conflicting, and always context dependent. The results of these evaluations must then be synthesized and analyzed in order to derive appropriate design decisions. The data from these evaluations then become the evidence or rationale for each design decision that is made. It is these 'pieces of evidence' or rationales that must be documented. Providing data reporting templates (or enabling the development of custom templates developed by the design team) to document empirical test results, field observations, and subject matter expert interviews can support data-driven design. These templates, if used by the entire design team can help ensure comprehensive recording of methods and results, and consistency in reporting which will ease searchability.

Capture ideas that were rejected, not just those that were accepted. To understand why a system design is the way it is, one must also understand how it could be different, and why the choices that were made are appropriate (MacLean, Young, and Moran, 1989). It is often more informative to know why something was not included than to know why something was included (Leveson, 2001). Information about designs or design elements that were considered, evaluated and rejected is very important, and yet rarely documented. If documented, it could reduce wasted time and resources spent re-learning these lessons during either a redesign of the system, or design of similar systems.

Requirements to Support Design Implementation:

- Embed commonly used design tools, standards and guidelines
- Accept multiple file types as created during the design process
- Provide multiple easy-to-use input methods
- Support private designer sand-boxes
- Represent multiple stakeholders viewpoints
- Include web-based collaboration tools (wiki-like)
- Implement web-based, platform-independent, structures
- Allow ideas to be moved, changed, and linked easily
- Do not force ideas into pre-existing labels
- Provide data reporting templates (or allow custom development)
- Link design decisions to empirical evidence or rationale
- Tie decisions to requirements identified during the Conceptual Design phase
- Enable capture of ideas that were considered but rejected
- Provide structure but not rigidity

3. Design Evaluation and Transfer

The Design Evaluation and Transfer phase includes design verification and validation as well as the transfer of design information for a variety of purposes including the development of training or instructional materials, procedures for operational use, system manufacturing, or design modification.

Design verification refers to the process of confirming that the system has been designed as specified and that the design output meets the design input guidelines, whereas validation refers to the process of checking that the design output addresses the users' needs and intended uses (Andre, Hooey, & Foyle, 2007). In essence, design verification and validation serve as the checkpoints for a good design. In order to accomplish design verification and validation, one needs access to the rationale behind the design process. Often a design can only be evaluated via the design trace. For example, two designs may seem comparable, however, they may differ with respect to the design process used. One may have undergone a more thorough design and evaluation process and the decisions may be based on a more solid empirical foundation, which would otherwise be invisible to those removed from the design process. A design rationale capture tool can aid in design verification and validation if it allows users to generate customizable reports to interrogate the database for answers to questions such as: How has this requirement been filled?, Have any requirements been violated? and, Is this feature applied consistently? Similarly, the design rationale capture system should enable the evaluator of a design to identify design elements that lack relational components -- that is, design elements that are not linked to supporting evidence or rationale.

Perhaps one of the most compelling benefits of a design rationale capture system is re-use of design knowledge to enable design modifications. Design modifications may create new problems if the original design rationale is not considered. A system could be adopted or modified for use under circumstances for which it was never intended, creating safety hazards, or a failure to realize potential benefits. Unless the original designer is involved, often those carrying out the design modifications have no way to access important design details, recommended procedures, and other usage constraints that are not contained in the code or visible from the prototype or finished product. As a result these are often ignored or misrepresented as the system travels through the development and modification process (Andre, Hooey, & Foyle, 2005). A design rationale capture tool can aid in the design modification process by identifying which design elements are affected if a critical assumption, constraint, or related design element is modified.

It is difficult to anticipate all of the future uses of design rationale information and therefore the data retrieval needs. The information needs of engineers and designers are very different than those of management, contract or acquisition specialists, and operational personnel. Data retrieval needs must be flexible enough to allow retrieval for diverse purposes such as understanding the final design for production, maintenance, or operational use; verifying the design, or proposed modifications to the design; and transferring knowledge to other design teams for similar or related systems. Multiple views such as a high-level snapshot from which one can drill down for a more detailed trace may be required to meet the varied needs of users. Gruber and Russell (1996) also encountered this problem and argued for a generative approach in which design rationales are constructed in response to information requests, from background knowledge and information captured during design. They concluded that it is more important to capture the data that might be used to infer answers later than to attempt to anticipate the questions and capture pre-formatted answers.

Requirements to Support Design Evaluation and Transfer

- Generate customizable reports
- Identify design elements that lack relational components
- Identify affected design elements if a critical assumption or constraint is changed
- Provide multiple search levels including a high-level snap shot and detailed drill-down views
- Follow a generative-approach to collect data that can be used to infer answers later (rather that attempting to collect answers)

CONCLUSION

There are several benefits to be gained from documenting the design rationale of any design project. The benefits of capturing and transferring critical design assumptions, constraints, design philosophies and design rationale extends not only to the system developer, who is likely to produce a more veridical and effective version of the product in the presence of a detailed representation of the design specification, but also to fellow researchers and system designers, who can learn from viewing the trace of design philosophy, rationale, decisions and data that led to the ultimate design. A design rationale capture tool can also be used to ensure that the final product design meets the design requirements and may also be useful for those who require detailed knowledge of the design for purposes of certification, manufacturing and production, development of documentation materials (i.e., user manuals), and training and procedures development. The diversity of this secondary user group emphasizes the need for a very flexible, customizable, information retrieval system. Finally, a design rationale capture tool can enable effective maintenance throughout the lifecycle of the product or project – allowing for the design to change as conditions change, yet still building on the underlying foundation of the original design.

Integration of design rationale capture tools into NASA's design processes will result in:

- Improved efficiency of the design process
- Better design of complex systems due to integration of evidence-based rationale
- Reduced risk that designs or redesigns will violate assumptions and design philosophy
- Improved acquisition, management, and retention of complex design knowledge
- Better communication among team members and transfer to NASA industry partners

REFERENCES

Andre, A. D., Hooey, B. L, & Foyle, D. C. (2005). Capturing the research and development process of aviation systems: Creating a multi-media living legacy. Proceedings of the 13th International Symposium on Aviation Psychology. Oklahoma City, OK.

Andre, A. D., Hooey, B. L, & Foyle, D. C. (2007, in press). Human-systems integration engineering support. In M. Kaiser & J. McCandless (Eds). Space Human Factors

Engineering: Technical Gap Analysis White Papers. NASA Ames Research Center: Moffett Field.

Carroll, J. M., & Moran, T. P. (1991). Introduction to this special issue on design rationale. Human-Computer Interaction, 6(3-4). 197-200.

Conklin, E. J., & Yakemovic, K. B. (1991). A process-oriented approach to design rationale. Human-Computer Interaction, 6: 357-391.

Deutsch, M., & Nichols, J. S. (2000). Advanced approach to concept and design studies for space missions. Astrophysics and Space Science, 273 (1), 201-206.

Gorry, G. A., Long, K. B., Burger, A. M., Jung, C. P., & Meyer, B. D. (1991). The virtual notebook system: An architecture for collaborative work. Journal of Organizational Computing, 1(3), 233 – 250.

Gross, M. D. (1996). The electronic cocktail napkin – Computer support for working with diagrams. Design Studies, 17 (1).

Hooey, B. L., Foyle, D. C., & Andre, A. D. (2002). A human-centered methodology for the design, evaluation, and integration of cockpit displays. Proceedings of the NATO RTO SCI and SET Symposium on Enhanced and Synthetic Vision Systems. Ottawa, Canada.

Klein, M. (1993). Capturing design rationale in concurrent engineering teams. IEEE Computer Journal. Special Issue on Computer Support for Concurrent Engineering, 26(1), 39-47.

Lee, J. (1997). Design rationale systems: understanding the issues. IEEE Expert, 12(3), 78-85.

Leveson, N. (2001). Systemic factors in software-related spacecraft accidents. AIAA Space 2001 Conference and Exposition. Paper AIAA 2001-4763. AIAA: Reston, VA.

MacLean, A., Young, R. M. & Moran, T. (1989). Design rationale: The argument behind the artifact. Proceedings of Computer Human Interaction. 247 – 252.

NASA (1995). Man-Systems Integration Standards – Revison B. Available at http://msis.jsc.nasa.gov/

Myers, K. L., Zumel, N. B., and Garcia, P. (1999). Automated capture of rationale for the detailed design process. Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence.

Rittel, H. (1984). Second generation design methods. In N. Cross (Ed.) Development in Design Methodology. New York: John Wiley & Sons,

Rosson, M. B., and Carroll, J. M. (2002). Usability Engineering: Scenario-based Development of Human Computer Interaction. San Francisco: Morgan Kaufmann.

Shipman, F. M., & McCall, R. J. (1997). Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. Artificial Intelligence in Engineering Design, Analysis, and Manufacturing (AIEDAM), 11 (2), 141-154.

Shum, S. J. B., MacLean, A., Bellotti, V. M. W., & Hammond, N. V. (1997). Graphical argumentation and design cognition. Human-Computer Interaction, 12 (3), 267-300

ACKNOWLEDGEMENT

The authors gratefully acknowledge Anthony Andre and Mark Chignell who provided insights into the design process and the system requirements for a design rationale capture tool.