Design Rationale Types and Tools

Janet E. Burge & David C. Brown

AI in Design Group, Computer Science Department, WPI

Fall, 1998

1. Introduction

A design can be documented in many different ways. Design documentation ranges from formal design specifications, often following a rigorous standard imposed by an outside agency, to informal notes contained in the notebooks of the individual designers. One type of documentation that is often only recorded informally, if recorded at all, is design rationale (DR) - the reasons behind the design decisions and, in some cases, a record of what decisions were *not* made and why.

This information could be valuable for many reasons, both during the design process, and later when the designed artifact requires modification. There have been many systems developed to capture and use rationale and several survey papers comparing them. The surveys resulted in several ways to classify design rationale tools. This paper combines these classifications and describes how they apply to a selection of DR tools. It also suggests some areas for future work.

1.1. Definitions

There are many definitions of Design Rationale:

"Design rationale expresses elements of the reasoning which has been invested behind the design of an artifact" [Shum & Hammond, 1993].

"Design rationale is the reasoning and argument that leads to the final decision of how the design intent is achieved." "Design intent is the `expected' effect or behavior that the designer intended the design object should achieve to fulfil the required function." [Sim & Duffy, 1994]

"Design rationale means statements of reasoning underlying the design process that explain, derive, and justify design decisions" [Fischer, et. a., 1995]

Design rationale means "information that explains why an artifact is structured the way that it is and has the behavior that it has" [Conklin, Burgess-Yakemovic, 1995].

"Design rationales include not only the reasons behind a design decision but also the justification for it, the other alternatives considered, the tradeoffs evaluated, and the argumentation that led to the decision" [Lee, 1997].

While all these definitions have their merits, Lee's [1997] definition most clearly states the content and purpose of design rationale.

1.2. Types of Rationale

Rationale can be classified into several types. These types are not mutually exclusive and some systems may support multiple types of rationales. The following types of rationale are discussed in this document:

- *Argumentation based* - the design rationale is primarily used to represent the arguments that define a design [Garcia, 1993]. These arguments consist of issues raised, alternative responses to these issues, and arguments for and against each alternative.

- *History-based* - the rationale consists of the design history – the sequence of events that occurred while performing the design [Garcia, 1993]. This information can be stored in many forms. It could be in the form of entries in a design notebook, an archive of e-mail messages, or other types of documents that capture actions taken over time.

- *Device-based* - a model of the device itself is used to both obtain and present rationale [Gruber, 1990]. The explanations of the design would be produced by using the model to simulate the behavior of the device. It would be possible for the user to view the model and ask questions about its design and behavior.

- *Process-based* -- the DR capture is integrated into the design process itself which guides the format of the rationale. In Ganeshan, et. al. [1994], the design description is modified only by changes to and refinements of the design objectives, thus capturing the rationale as part of the design process.

- *Active document-based* - the DR is pre-generated and stored in the system. In these systems, the designer creates the design and the DR system generates the rationale for it based on the system's stored knowledge. For each decision made, the system compares the decision made by the user with the decision that it would have made based in its knowledge. If the actions of the user conflict with the system recommendations, they are given the option of changing their decision or modifying some of the criteria.

1.3. Uses for Rationale

Design rationale consists of a lot of different information including the history of the design process, and the reasons for making each decision. This information can be useful in several aspects of designing. These include design verification, design evaluation, design maintenance, design reuse, design teaching, design communication, design assistance, and design documentation.

- *Design verification* -- the goal is to use the rationale in order to verify that the design meets the requirements and the designer's intent. This verification can occur at any point in the design process.

- *Design evaluation* -- similar to verification except that the rationale is used to evaluate designs (and partial designs) and design choices relative to one another.

- *Design maintenance* -- the design rationale is used to determine what choices were made when performing the design in order to locate sources of design problems or to indicate where changes need to be made in order to modify the design. By keeping track of alternatives that had been rejected, the designer can avoid making a choice that was rejected earlier.

- *Design reuse* -- the design rationale is used to determine which portions of the design can be reused and in some cases, suggest where and how it should be modified to meet a new set of requirements. It is especially important to let the designer know why the decisions were made. In some cases, what may seem like an inefficient solution may actually be critical to the design as a whole. Without the presence of rationale that indicates this, the design may be changed in a way that could be harmful.

- *Design teaching* -- design rationale can also be helpful in assisting teaching new personnel about the design. Besides providing insight into how it works, the rationale shows why each design choice was made. This conveys more information than a static description. Some design rationale systems allow the user to ask questions about the design, this is often a faster and easier way to learn about the design than wading through large amounts of design documentation. Learning support is especially crucial when the original designers are not available to teach a new designer.

- *Design communication* -- the presence of rationale improves design both after and during the design process. By capturing the design choices and the reasons behind them, this information can be made available to others affected to the design to both give them insight into the design and allow them the opportunity to provide their input into the process [Fischer, et. al., 1995]. It also can provide an efficient way to detect conflicts in the work of multiple designers and shorten the review cycle [Pena-Mora, et. al, 1995]. By capturing the reasons behind design decisions, design rationale can be used to answer questions from design reviewers who need to know why a particular choice was made or why an expected choice was not made.

- *Design assistance* - design rationale can also provide assistance during the design process. The ability to verify and evaluate design choices allows the designer to view the results of their design decisions. Documenting the argumentation can perform several functions: it clarifies the arguments by encouraging designers to document the information and it can be evaluated to ensure that all issues are resolved and that alternatives are selected that meet the requirements without violating any. Some features that provide immediate benefit are constraint/dependency checking, where rationale is used to verify that the design is correct, simulation, where the system allows the designer to check the impact of design modifications, and conflict mitigation, where the system looks for constraint violations between multiple designers and informs them when there is a problem.

- *Design documentation* - design rationale also assists in documenting the design by offering a picture of the history of the design and reasons for the design choices as well as a view of the final product. If the rationale is stored in a computer-readable form, it can be used as part of a custom documentation generating system by allowing documentation to be generated from different perspectives and, in some systems, allowing the user to ask questions about the design. Some systems use the rationale to generate documentation aimed at different groups of people. A customer, for example, would require a different level of detail than a designer.

2. Tool Attributes

The following subsections describe attributes of design rationale approaches and systems.

2.1. Services Provided

As described above, there are many different uses for design rationale. These fall into the

categories, listed above, of design verification, design evaluation, design maintenance, design reuse, design learning, design assistance, design communication, and design documentation. There is a great deal of overlap in these categories since one design rationale feature may support multiple categories. For example, design reuse and design maintenance are both concerned with knowing which parts of the design support which requirements and how the design will need to change if a requirement is modified.

Each of these categories provide services to the user. For example, a design assistance service would be simulation, where the user is given the capability to view the impact of a particular design choice on the rest of the design.

2.2. What Information is Represented

Almost any type of design related information is potentially useful and can be classified as some type of rationale. A generic structure for design rationale is described in Lee, [1997]. This consists of three layers: decision layer, design artifact layer, and design intent layer.

2.2.1. Design Intent Layer

This is the highest level of design rationale. It contains the information that drives the decisions made during the design process. This includes intents, strategies, goals, and requirements. It is important to capture this information so that the design, and later modifications to it, can be verified.

2.2.2. Decision Layer

The decision layer provides detailed information describing the decisions made during the design process. This is the information consulted to determine why a particular design choice was made. It consists five sub-layers:

- *issue* - individual issues and their relations (generates, depends-on, replaces).
- *argument* - arguments behind a decision and their relations (supports, refutes, qualifies).
- *alternative* - individual alternatives and their relations (component-of, incompatible, specializes)
- *evaluation* - evaluations - used to rank alternatives
- *criteria* - criteria used and their relations (mutually exclusive, tradeoffs, specializes). Criteria are used to group evaluations and arguments

2.2.3. Design Artifact Layer

This is information about the ‘thing’ being designed (such as components and how they relate to each other).

2.3. How the Information is Represented

Information can be represented in a range from formal (computer readable) to informal (not computer readable). A formal approach allows the computer to use the data but does not always output information in a form that a human can understand. In addition, it requires that data be given to it in a more rigid format. An informal approach provides data in forms that are easily generated and understand by a human but can not be used by the computer. Semi-formal approaches attempt to use the advantages of both approaches.

Another advantage of informal representation is that the rationale stored can be in the

original format found in the natural byproducts of the design process. This allows for non-intrusive capture. Unfortunately this results in large amounts of data that, while it contains much useful information, is difficult to navigate through or perform analysis over. One approach used is "incremental formalization" where data is captured informally but then selectively converted to a formal or semi-formal representation when needed.

2.4. How it is Produced/Captured

Producing and capturing design rationale is a major difficulty in creating a design rationale system. The ideal design rationale system would be non-intrusive. This is desirable because recording rationale is not only time consuming for the designer, it also can distract them from the design task they are performing. The following methods for rationale capture have been studied:

- *reconstruction* [Lee, 1997] – these are design rationales captured outside the design process, usually after it has been performed. The advantage of this approach is that it is non-intrusive, the disadvantage is that it may not accurately or completely capture the rationale

- *methodological byproduct* [Lee, 1997] – in this approach, the rationale emerges during the design process. This is done by having the designer use a methodology that aids in performing the design task that also captures the rationale.

- *apprentice* [Lee, 1997] – in this approach, the system watches the actions taken by the designer and asks questions when it does not understand an action. In these systems, the rationale is, to some extent, pre-generated – if the designers actions match the systems prediction than the system-generated rationale is saved.

- *automatic generation* [Lee, 1997] – in this approach, design rationales are generated from an execution history.

- *historian* [Chen, 1990] - In this approach, a person or computer program keeps track of all actions during the design process. This method is similar to *apprentice*, except the system does not make suggestions. It is also similar to *automatic generation* except that the rational is specifically recorded during the design process, not generated later.

The following tables show several ways that these capture techniques can be grouped. One is to compare when the rationale is captured. Table 1 shows when each tool captures rationale.

Table 1. Rational Capture Time

Capture Method	After Design	During Design
Reconstruction	X	X
Methodological byproduct		X
Apprentice		X
Automatic Generation	X	
Historian		X

Another distinction that can be made is between design rationale and design history. Although the terms are often used interchangeably, a design history differs from rationale in that the rationale captures the choices made, the choices rejected and the reasons behind both. The design history only records the choices made and does not need to record the reasons. Table 2 shows which tools capture the history and which record the rationale. For

some tools, it is not clear if the capture method will capture the rationale, for these, a question mark is given in the rationale column.

Table 2. Design Rationale vs. Design History

Capture Method	Design History	Design Rationale
Reconstruction	X	?
Methodological Byproduct	X	X
Apprentice	X	X
Automatic Generation	X	
Historian	X	?

Another way of grouping the capture methods is by the amount of interaction with the designer. Capture methods that interact with the designer are often more intrusive than those that do not. The intrusiveness is lessened if the interaction is viewed as being useful or if it is consistent with the normal design process followed. Table 3 shows the amount of designer interaction for each capture method.

Table 3. Interaction with Designer

Capture Method	Low Interaction	High Interaction
Reconstruction	X	
Methodological Byproduct		X
Apprentice		X
Automatic Generation	X	
Historian	X	

2.5. How it is Accessed

Capturing design rationale is of limited usefulness unless it can be accessed. Design rationale retrieval can be classified as user–initiative or system–initiative [Lee, 1997], [Pena-Mora, et. al., 1995]. In user-initiative systems, the user takes responsibility for knowing what parts of the rationale they are interested in. The system supports this by allowing various types of queries and browsing. In system-initiative systems, the system decides what rationale the user needs to see. This can be more efficient than user–initiated but requires that the system have enough knowledge incorporated in it so it can determine what rationale is needed.

The systems also differ in how the information is organized. One common organization is to have the rationale organized by the component of the designed artifact that it relates to [Gruber, 1990]. This is a good representation if the system is to be used by new designers who need to understand how the design is organized. Other systems are organized around a single process that is modeled [Brown&Bansal, 1991]. This is useful for users who are approaching the problem at a higher level and are interested in the process, not just the product. It is also useful if the rationale system is designed to work for multiple domains and types of design where the process is similar. Some systems model different aspects of the design process and store the rationale in this way [Klein, 1992]. This is useful if the design needs to be viewed or documented from different perspectives.

2.6. Integration

Design rationale can be stored in many forms. These include designers notebooks, design sketches, comments in software code, e-mail messages, and many more.
Two types of integration are useful for design rationale. One is to tie all these different

design documents together so that the rationale can be extracted. Another is to integrate the rationale capture into a tool used in design. Integration with a tool is useful because then the rationale becomes a natural byproduct of the design process. Disadvantages of this approach are that it is highly dependent on the version of the tool (and may break if the tool vendor releases a new version) and that it only captures the rationale as coming from that tool. The information captured will form a history but will not record the reasons for any of the decisions unless the tool is modified to ask for them.

2.7. Domain Type

Some design rationale tools, specifically those that have knowledge behind them, are very domain specific. The methodology behind the tool, however, may be of general use. Other tools and methods are designed to be domain independent. The choice of tool depends on how it will be used - if the knowledge is built into the tool then the capture effort takes place "up front" during the knowledge elicitation required to obtain the knowledge. The advantage of this approach is that the system will not need to obtain as much information from the designer while they are designing. The disadvantage is that the knowledge elicitation is also time consuming and may end up gathering information that is never used. If the system will be used for many designs, then this effort may be worth while. If not, it may be more efficient to capture rationale during the design.

2.8. Types of Design Supported

Some DR tools are aimed at a specific type of design. The type of design supported influences what data needs to be captured and what can be done with it. For example, systems that anticipate designers' choices and allow simulation are usually built for routine, parametric design applications. In these applications, it is easier to anticipate a likely choice and the rationale behind it. If a system was designed to support creative design, its role would be weighted more toward recording decisions (and rationale) rather than recommending them. The type of design needs to be taken into account when choosing/building a design rationale tool so that the best match can be made between the design task and the tool chosen.

2.9. Stage/Phase of Design Supported

Design rationale is different, both in content and use, for different phases of design. For example, early stages involve more data that needs to be captured than later ones. Later stages (particularly maintenance) will have fewer new decisions that need to be made but a potentially greater need to access the reasons for decisions made earlier. Earlier design stages also are likely to involve more negotiation than later ones, especially if the requirements are not yet fixed. As the design progresses, the decisions become more detailed and the alternatives fewer and narrower. Decisions made earlier in the design will constrain those made later.

2.10. Number of Designers Supported

Some design rationale systems only support one designer. Others support multiple designers. A single designer system is somewhat limited. Most design projects that are large enough to warrant rationale capture will have more than one designer involved. Allowing multiple designers is difficult. There needs to be mechanisms for ensuring that the terminology is consistent. For example, an argumentation system needs to ensure that the same argument is not conducted multiple times under different names. Multiple designer systems can also be used to detect conflicts between the work of the different designers.

2.11. Notation

Some DR systems are built on a notation, or language, for representing the DR. Many different systems may share the same notation. The IBIS notation is used by two IBIS systems: gIBIS (graphical IBIS) and itIBIS (text based IBIS) [Conklin&Burgess-Yakemovic, 1995]. It is also the basis for another notation, PHI that is used in JANUS [Fischer, et. al, 1995]. For many systems, they created a notation that they then used for a DR tool. Examples of this are DRCS [Klein, 1992], which uses the DRCS language and SHARED-DRIM [Pena-Mora, et. al., 1995], which uses the DRIM model. The notation used supports both the capture, by specifying the information that needs to be captured, and the representation, by showing the relationships between different data items captured.

3. Existing Tools

The following subsections describe some design rationale tools used for capture and/or access and how they can classified by the attributes described above.

3.1. Design History Tool

The Design History Tool [Chen, et. al., 1990] records the design history, i.e. the constraints and decisions that occur from the initial design specification to the detailed design. This system is intended to both document and playback the design and design process. The data representation is based on the results of videotape analysis. The resulting model contains design objects, decisions, operators, and three types of constraints (given, derived, and introduced). The design constraints are considered to be the most important aspect of this system/methodology. The type of constraint identifies its source: given constraints are those that come from external sources, derived constraints are generated during the design process, and introduced constraints are those introduced during the design process but not derived from other constraints. Introduced constraints include those introduced by the designer based on their knowledge of the domain. The playback facility exists so the user can retrieve information about the design.

System Type	Process Model Based w/Argumentation
Services	Documentation
Represented Information	Artifact layer (objects, features), Decision layer (alternative proposals, rejected proposals), Intent Layer (Constraints)
Representation Method	Semi-Formal
Capture Method	Can not tell (not implemented)
Access Method	User-initiated
Domain	Mechanical Design
Design Type	Configuration
Design Phase	Specification to Detailed Design
Number of Designers	Single?
Notation	Their own design model

3.2. An Intelligent Design Evolution Management System (AIDEMS)

AIDEMS [Thompson & Lu, 1990] is a design environment that also provides the ability to display the design rationale. It performs configuration design starting with the initial product specification. It uses a model of design where the first step is creating a product specification. The specification is then refined into more and more detail. At each level, there exist functional requirements and constraints. When attributes values are assigned,

they are tested against these constraints.

System Type	Process Model Based
Services	Documentation, Simulation
Represented Information	Artifact layer, Intent layer (constraints)
Representation Method	Formal
Capture Method	Methodological byproduct
Access Method	User Initiated
Domain	General
Design Type	Configuration
Design Phase	All
Number of Designers	Single
Notation	Its own

3.3. Process Technology Transfer Tool (PTT)

PTT [Brown&Bansal, 1991] captures design information throughout the manufacturing process. It records information in the original form in which it is created/used by the designer. The information is connected to a model of the manufacturing process. This model is a hierarchical depiction of the manufacturing process where each step can be decomposed lower level steps. The design documents are attached to the appropriate phase and level of the hierarchy. The system also controls document modifications by requiring that all modifications be explained.

System Type	History-based
Services	Documentation
Represented Information	Varies
Representation Method	Informal
Capture Method	Methodological byproduct/Reconstruction
Access Method	User Initiated
Domain	Manufacturing (could be generalized)
Design Type	Any
Design Phase	All
Number of Designers	Multiple
Notation	The manufacturing process model

3.4. Design Rationale Capture System (DRCS)

DRCS [Klein, 1992] uses a rationale language built on a generic model of design reasoning. It is intended to support the entire design process by supporting the capture of design description knowledge and design rationale. DRCS uses assertions consisting of entities (modules, tasks, specifications, and versions) and claims about these entities to capture design reasoning. These assertions and claims can be used to describe artifact synthesis, plan synthesis, evaluation of the design, the design intent, relationships between different versions of the design, and the argumentation behind the design.

System Type	Process-based, Argumentation-based
Services	documentation
Represented Information	Artifact, Intent, Decision
Representation Method	Formal/Semi-formal
Capture Method	Not clear
Access Method	User-initiated

Domain	Any
Design Type	Hierarchically decomposed
Design Phase	Any/all
Number of Designers	Multiple
Notation	Its own

3.5. Active Design Documents (ADD)

Active Design Documents [Garcia, et. al., 1993] is a design rationale system for routine, parametric design. By having a domain and task specific knowledge based, the designer can assign parameters. If the designer's recommendation matches the system, the system records rationale already built into the knowledge base. If there is a conflict between the designer's action and the systems, the designer is informed and allowed to either modify the criteria, change their action, or override the system's recommendation. ADD also allows the designer to simulate parameter changes.

System Type	Active document-based
Services	Simulation, constraint-checking, documentation
Represented Information	Artifact, Intent, Decision
Representation Method	Formal
Capture Method	Apprentice
Access Method	System-initiated
Domain	HVAC
Design Type	Routine, parametric
Design Phase	Preliminary Design
Number of Designers	Single
Notation	It's own

3.6. Reconstructive Derivational Analogy (RDA)

RDA [Britt&Glagowski, 1995] is part of a larger system, the Circuit Designer's Apprentice (CDA). CDA is a system that, when given the requirements for a new electrical circuit, searches a database of already designed circuits to find the closest match. If there are no circuits that match, or match after minor adjustments, RDA is used to create a design plan from the existing circuit. This design plan is then 'replayed' with the new requirements to create the new circuit design.

System Type	History-based
Services	Maintenance, Documentation
Represented Information	Artifact, Decision, Intent
Representation Method	Formal
Capture Method	Reconstruction
Access Method	User-initiated?
Domain	Electrical Circuit Design
Design Type	Routine, hierarchical, must have database available.
Design Phase	Detailed Design/Implementation
Number of Designers	Single
Notation	Its own

3.7. gIBIS, itIBIS

gIBIS and itIBIS [Conklin&Burgess-Yakemovic, 1995] are two systems that implement the IBIS (Issue Based Information Systems) notation for design and planning dialogs. These systems create "issue-maps" starting with a root issue specifying the main problem with resolutions and arguments supporting or objecting to these positions. It expands into secondary issues, which have resolutions and arguments, and so on. gIBIS uses a graphical notation, while itIBIS uses a simple indented text format. The text format is easy to learn and use but does not scale well. The itIBIS system was used in case studies as a way to structure note taking during meetings. It was found to be useful because it organized the meeting notes and helped guide the progress of the meeting by identifying when the discussion got off track.

System Type	Argumentation-based
Services	Documentation, Maintenance
Represented Information	Decision Layer
Representation Method	Semi-Formal
Capture Method	Methodological Byproduct/Historian
Access Method	User-initiated
Domain	Generic
Design Type	Any
Design Phase	Any
Number of Designers	Multiple
Notation	IBIS

3.8. JANUS

JANUS [Fischer, et. al., 1995] is a design environment for kitchen design. It consists of two parts: JANUS_CONSTRUCTION where kitchens are designed from a palette and JANUS_ARGUMENTATION, an argumentative hypertext system. JANUS_ARGUMENTATION uses an issue-base that was pre-created from various KA activities (protocol studies, design books). The user can navigate through the issue base to explore issues, answers, and arguments. The reason behind JANUS is that studies have shown that designers stop recording rationale when they start the actual construction (detailed design) process. In JANUS, the designer uses JANUS_CONSTRUCTION to do the design. JANUS_ARGUMENTATION comes in if the designer makes a design decision that violates a design rule. JANUS_ARGUMENTATION presents the designer with the reasons for the violation and examples of designs that meet the requirement.

System Type	Argumentation-based
Services	Constraint checking, Simulation
Represented Information	Artifact, Decision, Intent
Representation Method	Semi-Formal
Capture Method	Not specified - assumes KB of rules and example designs exists. Rationale use, not capture.
Access Method	System-initiated
Domain	Kitchen Design

Design Type	Configuration
Design Phase	Construction
Number of Designers	Single
Notation	PHI/IBIS

3.9. Design Recommendation and Intent Model (DRIM/SHARED-DRIM)

SHARED-DRIM is a system built using the Design Recommendation and Intent Model (DRIM) [Pena-Mora, et. al, 1995]. Its main goal is to capture design rationale for use in conflict mitigation. SHARED-DRIM records design decisions and the rationale (argumentation) behind them and shares the information among the participating designers. By capturing rationale for each decision, and rationale behind when decisions are not accepted, the design modification and approval cycle is shortened.

System Type	Argumentation
Services	Constraint checking
Represented Information	Artifact, Decision, Intent
Representation Method	Semi-Formal
Capture Method	Apprentice
Access Method	System-initiated
Domain	Civil Engineering (large scale systems)
Design Type	Configuration - hierarchical representation
Design Phase	All
Number of Designers	Multiple
Notation	DRIM

3.10. Hyper-Object Substrate (HOS)

HOS [Shipman&McCall, 1996] is a hypermedia representation of DR combined with KB system features. Design artifacts and design communication represented together. It is built using incremental formalization where all design communication is captured but only selective portions are converted into a formal representation for use. A useful capture feature is that it allows import of e-mail and news so designers can keep using normal communication methods.

System Type	Argumentation
Services	Documentation, Maintenance
Represented Information	Artifact, Decision, Intent
Representation Method	Semi-Formal (incremental)
Capture Method	Historian, Reconstruction
Access Method	User-Initiated and System-Initiated
Domain	Generic
Design Type	All
Design Phase	All
Number of Designers	Multiple
Notation	Argumentation notation.

3.11. PHIDAS

PHIDIAS [Shipman&McCall, 1996] is a system used for 2D and 3D graphical design. It supports use of design argumentation and multimedia information by providing indexing and browsing capabilities. It also provides knowledge-based critiquing of designs. It is not clear how information is captured and imported into the system.

System Type	Argumentation
Services	Documentation, Maintenance, Critiquing
Represented Information	Artifact, Decision, Intent
Representation Method	Semi-formal
Capture Method	Historian, Reconstruction
Access Method	System-Initiated and User-Initiated
Domain	Graphic design
Design Type	Any
Design Phase	Any
Number of Designers	Multiple
Notation	PHI

3.12. M-LAP

M-LAP [Brandish, et. al., 1996] is a machine-learning apprentice system that is integrated with design tools. It records user actions at a low level and then forms them into useful sequences. These sequences are parts of tasks, which are then parts of higher level tasks, etc. These sequences are formed using machine-learning as the system is used. It is not clear how much use needs to occur before enough learning occurs to be useful. The information stored is primarily the history, not the rationale, since reasons for choices are not recorded.

System Type	History-based, Active document-based
Services	Assistance
Represented Information	Decision
Representation Method	Formal
Capture Method	Apprentice
Access Method	System-Initiated
Domain	Generic
Design Type	Any supported by a design tool.
Design Phase	Any
Number of Designers	Possibly multiple
Notation	Requires integration with a design tool.

3.13. Device Modeling Environment (DME)

DME is an environment that models structure, behavior and function of electromechanical devices [Gruber, 1990]. In this system, documentation is generated by DME "on demand." This allows the user to ask questions about the parts of the system that they are interested in. The system supports answers to a pre-enumerated set of questions. These include "what is it," "what happened," and "what if." Answers are turned into text using a natural language production system. DME can also be used to simulate behavior of a device. The simulation is used to create the rationale by performing "rationale by demonstration." In this technique, the system is given a scenario and the system provides the results and the reasons. The scenario is manipulated until the desired results are obtained and the reasons for that result then comprise the rationale.

System Type	Active document-based
Services	Documentation, Maintenance, Simulation
Represented Information	Artifact, Decision, Intent
Representation Method	Formal
Capture Method	Documentation, Methodological Byproduct
Access Method	User-Initiated
Domain	Electro-Mechanical
Design Type	Any
Design Phase	Any
Number of Designers	Multiple
Notation	System Model

3.14. SIBYL

SIBYL is a Qualitative Decision Management, a system that assists in the management and representation of qualitative aspects of the decision making process [Lee, 1990]. This system uses a Decision Representation Language (DRL) that represents the argumentation for the decisions being made. SIBYL uses DRL to create "decision graphs" showing the different alternatives and their evaluations.

System Type	Argumentation-based
Services	Constraint checking, Simulation
Represented Information	Artifact, Decision
Representation Method	Semi-Formal
Capture Method	Not specified - assumes KB of rules and example designs exists. Rationale use, not capture.
Access Method	System-initiated
Domain	Any
Design Type	Not design specific
Design Phase	N/A
Number of Designers	Multiple
Notation	DRL

4. Related Topics

Design rationale capture and use has some similarities with other topics. It is useful to studies these similarities so that research in design rationale can take advantage of research done on other problems. Some fields that address similar problems are:

- *Automated design* -- both automated design and design rationale research is concerned with the representation and use of design knowledge [Thompson&Lu, 1990].

- *Writing support tool research* -- Both have to address similar problems: the possibility of forgetting needed information, keeping track of goals, choosing incorrect paths and needing to backtrack, and others [Shum&Hammond, 1993].

- *Document management* -- in some DR systems, the capture process consists of obtaining the design documentation and storing it in way where useful information can

be accessed easily. This is a documentation management problem as well as a representation problem. The issues include how to store multiple forms of data (drawings, text, etc.) and how to find information within the data stored (indexing schemes, etc.).

- *Configuration management* - maintaining different versions of the design, and the rationale, requires the version control capabilities found in many configuration management systems. Version control involves not only managing a single chain of modifications, it also involves handling versions that diverge. This is similar to the problems faced when maintaining rationale for new designs based on existing ones.

- *Change Management* - changes made to an artifact are tracked and recorded, much like design rationale would be. Much of the information used in change management is rationale. For example, rationale (if present) is used to determine what changes need to be made. The description of the changes, and the reasons for them, are also rationale.

5. Areas for Investigation

Although there has been a large amount of research done in design rationale, there are still some areas that could use further exploration. The following subsections describe these areas.

5.1. Relationship Between Rationale Capture and Use and the Design Phase

The nature of design rationale, both capture and use, changes throughout the design life-cycle. Different design phases may produce different types of rationale. For example, more argumentation is likely to occur in the earlier stages than in the later. The level of detail in the rationale increases, as the design becomes more detailed. The rationale used also changes. For example, while there is not much argumentation created in the maintenance phase of design, argumentation from earlier phases is very useful so that the impact of proposed changes can be determined. This capability would be provided by a DR system that supports verification and evaluation.

Rationale also needs to be maintained throughout the life-cycle. Some rationale will be elaborated as more detailed decisions are made. Other rationale will be modified completely, such as if a decision made earlier in the design is shown to be incorrect and needs to be changed. As the rationale changes, these changes need to be propagated throughout the design and any inconsistencies detected.

Rationale reuse brings up many interesting issues. If a design, or part of a design, is reused to create a new design, the two designs will share some of the same rationale as well as additional rationale indicating why the reuse was performed. If changes to the original design require changes to the "child" design or vice versa, changes will need to propagate through the rationale as well.

5.2. Relationship Between Rationale Capture and Use and the Domain Type

Another area that might be interesting to investigate is the relationship between rationale capture and use and the type of domain. This has two aspects: the affect of the type of domain on the rationale capture method used and the ability to come up with rationale capture and use methods that can be used for multiple domains. The latter is particularly valuable if rationale is being captured for a system with components from multiple design disciplines.

The ease of rationale capture differs depending on the domain. For example, in some ways capturing rationale for software design is easier than for domains such as mechanical or electrical design. Much of the software knowledge (documentation and code) is already stored in a computer readable form and could be extracted and indexed. On the other hand, the rules and constraints governing a software design are not always as well known as for some other disciplines. Rationale capture for software is also difficult because software can be changed very easily – this is likely to result in more rationale to capture and maintain. It also greatly increases the chance that the rationale will become out of date.

5.3. Rationale Capture Issues

All research on DR agrees that rationale capture is a difficult problem. The "ideal" design rationale system would not intrude on the designers as they accomplish the design task. Existing systems try a number of approaches to solve this problem. One solution would be to observe the designers' every move and translate this information into rationale. This avoids intruding on the designer but has two problems: it may not capture all the information needed and the amount of effort to translate this information into a usable format is very great. Another method is to embed the design knowledge into the system so that it can supply the rationale itself. This also avoids intruding on the designer but has its own problems: the rationale stored by the system may not actually match that of the designer and there is a great deal of work involved in obtaining the knowledge to store in the system.

There are several ways that the capture problems could be addressed. One is to provide more incentive for the capture of information. This can be done by finding ways in which the rationale captured can be immediately useful to the designers. If they know that recording the rationale will benefit them, not just a colleague using the system years from now, they will be more likely to record this information. Another way is to find more ways that the rationale capture can be built into the current practice of the engineers. The HOS system [Shipman&McCall, 1996] imports e-mail into the rationale system. Similar use could be made of other design artifacts.

Another way to approach the capture problem is to determine which types of rationale will be the most useful for the proposed use or uses. This will not solve all the capture problems but it will reduce the overall amount of effort by targeting the capture effort toward areas that will provide the most payback.

5.4. Rationale Use Issues

Given that capture of rationale is difficult and often expensive, it is especially important to examine how rationale can be used and how to make this use easier. By offering useful features such as design verification and evaluation, the incentive to provide the rationale in the first place is increased. There were several ways that rationale can be used described earlier in this document: design verification, design evaluation, design assistance, design documentation, design maintenance, design teaching, and design collaboration.

One area of design where design knowledge has been captured and used successfully is for design documentation [Shipman&McCall, 1996]. Often there are externally imposed requirements that enforce the documentation effort. The disadvantages of the static documents is that they only provide one view of the design [Gruber, 1990]. This means that they can only supply information that was predicted to be useful, rather than responding to actual information requests. Another area where documentation is important is in tracing the design requirements. This is an area where rationale capture would be

useful – the reasons behind alternative choices are often the requirements for the design.

5.5. Relationship to other Areas

As described in an earlier section, there are many research areas that have similar issues to those for design rationale. More investigation could be done to see what could be learned from these areas.

5.6. Multiple Designer Support

Many of the current design rationale systems are not set up for multiple designers. The systems for which rationale is the most useful are often larger systems. A design rationale system needs to support multiple designers. This will allow for the full benefit of rationale by creating rationale systems that support conflict mitigation, evaluation, and information sharing.

6. Future Work

Most discussions of design rationale center around the very difficult area of design rationale capture. While capture poses many challenging problems, an area that should be investigated first is how the rationale is going to be used. The capture effort is only useful if the information gathered is useful. Systems that make immediate use of design rationale, rather than storing it for use by future designers, provide incentive for recording the information.

As described above, in section 5.4, there are many potential uses for rationale. Of these uses, one of the more interesting is the use of design rationale for design verification and design evaluation. When a design task begins, there are usually a number of requirements that the design needs to fulfill. As designing progresses, this list may change as requirements are refined, new requirements are added, and some requirements are corrected or deleted. It would be useful to have a methodology that would track these changes throughout the design process and provide a means of verifying that all requirements have been met. Design rationale is one way to address this problem.

In Ganeshan [1994], the design process is driven by the need to satisfy design objectives. These objectives can also be considered requirements. The design rationale capture consists of recording how these objectives are refined and how values are assigned in order to create the design. A similar way to address this would be to provide a way of ensuring that for every requirement, there is an alternative that satisfies it.

This is a useful process to aid in requirements engineering but does not provide a complete or understandable picture of the design. In Gruber [1991], the device-modeling environment (DME) was used as a means of documenting the design. The model could be queried in order to discover what the various components of the design did. An interesting approach would be to create a system where all design decisions are based on requirements, either original, refined, or derived, and where the information is represented based on a model of the device. The argumentation would be attached to the device model where appropriate (at a detailed or abstract level). This would provide the means to validate the design by ensuring that all requirements are met but would also provide an effective way of documenting the design.

7. References

Brandish M., Hague, M., Taleb-Bendiab, A. (1996), M-LAP: A Machine Learning

Apprentice Agent for Computer Supported Design, *AIinD Machine Learning in Design Workshop.*

Britt, B., Glagowski, T. (1996), Reconstructive derivational analogy: A machine learning approach to automating redesign, *in Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, No.10, Cambridge University Press, pp. 115-126

Brown, D., Bansal, R. (1991), Using Design History Systems for Technology Transfer, in *Computer Aided Cooperative Product Development*, D. Sriram, R. Logcher, and S. Fukuda, eds., Lecture Notes Series, No. 492, Springer-Verlag, New York, pp. 544-559

Chen, A., McGinnis, B., Ullman, D., Dietterich, T. (1990), Design History Knowledge Representation and Its Basic Computer Implementation, *The 2nd International Conference on Design Theory and Methodology*, ASME, Chicago, IL, pp. 175-185.

Conklin, J. and Burgess-Yakamovic, K. (1995), A Process-Oriented Approach to Design Rationale, in *Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carroll, eds., Lawrence Erlbaum Associates, Mahwah, NJ, pp. 293-428.

Fischer, G., Lemke, A., McCall, R., Morch, A. (1995), Making Argumentation Serve Design, *in Design Rationale Concepts, Techniques, and Use*, T. Moran and J. Carroll, eds., Lawrence Erlbaum Associates, pp. 267-294.

Ganeshan R., Garrett J., Finger, S. (1994), A framework for representing design intent, *Design Studies Journal*, V15 No. 1, January, pp. 59-84.

Garcia, A., Howard, H., Stefik, M. (1993), *Active Design Documents: A New Approach for Supporting Documentation in Preliminary Routine Design*, Tech. Report 82, Stanford Univ. Center for Integrated Facility Engineering, Stanford, Calif.

Gruber, T. (1990), Model-based Explanation of Design Rationale, in *Proceedings of the AAAI-90 Explanation Workshop,* Boston, July 30, 1990.

Gruber, T., Russell, D. (1991), *Design Knowledge and Design Rationale: A Framework for Representation, Capture, and Use*, Knowledge Systems Laboratory, KSL 90-45, Stanford University, Stanford: CA.

Klein, M. (1993), Capturing Design Rationale in Concurrent Engineering Teams, *Computer*, January, pp. 39-47.

Klein, M. (1993), DRCS: An Integrated System for Capture of Designs and Their Rationale, in *Artificial Intelligence in Design '92,* Gero, J. (Ed.), Kluwer Academic Publishers, pp. 393-412.

Lee, J. (1990), SIBYL: A qualitative design management system. In P.H. Winston and S. Shellard, eds., *Artificial Intelligence at MIT: Expanding Frontiers*, Cambridge MA: MIT Press, pp. 104-133.

Lee, J. (1997), Design Rationale Systems: Understanding the Issues, *IEEE Expert*, Vol. 12, No. 3, pp. 78-85.

Pena-Mora, F., Sriram, D., Logcher, R. (1995), Design Rationale for Computer-Supported Conflict Mitigation, *ASCE Journal of Computing in Civil Engineering*, pp. 57-72.

Pena-Mora, F., Vadhavkar, S. (1996), Augmenting design patterns with design rationale, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 11, pp. 93-108.

Potts, C., Bruns, G. (1988), Recording the Reasons for Design Decisions, *Proceedings International Conference on Software Engineering*, IEEE CS Press, pp. 418-427.

Shipman, F., McCall, R. (1996), Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication, *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 11, pp. 141-154.

Sim, S., Duffy, A. (1994), A New Perspective to Design Intent and design Rationale, in *Artificial Intelligence in Design Workshop Notes for Representing and Using Design Rationale*, 15-18 August, pp. 4-12.

Shum, S., Hammond, N. (1993)*, Argumentation-Based Design Rationale: From Conceptual Roots to Current Use*, Tech. Report EPC-1993-106, Rank Xerox Research Centre, Cambridge

Thompson, J., Lu, S. (1990), Design Evolution Management: A Methodology for Representing and Utilizing Design Rationale, in *2nd International Conference on Design Theory and Methodology*, ASME, Chicago: IL, pp. 185-191.