REASONING WITH DESIGN RATIONALE

J. BURGE, D. C. BROWN
AI in Design Research Group
Department of Computer Science
WPI, 100 Institute Road
Worcester, MA 01609, USA

Abstract. Design Rationale (DR) consists of the decisions made during the design process and the reasons behind them. Because it offers more than just a "snapshot" of the final design decisions, DR is invaluable as an aid for revising, maintaining, documenting, evaluating, and learning the design. Much work has been performed on how DR can be captured and represented but not as much on how it can be *used*. In this paper, we investigate the *use* of DR by building InfoRat, a system that inferences over a design's rationale in order to detect inconsistencies and to assess the impact of changes.

1. Introduction

Standard design documentation consists of a description of the final design itself: effectively a "snapshot" of the final decisions. Design rationale (DR) offers more: not only the decisions, but also the reasons behind each decision, including its justification, other alternatives considered, and argumentation leading to the decision (Lee, 1997). This additional information offers a richer view of both the product and the decision making process. DR is invaluable as an aid for revising, maintaining, documenting, evaluating, and learning the design.

If design rationale were available, designers revising a design could use it to determine the original designer's intent, as well as determining what alternatives had already been considered and why they were rejected. This can help to avoid duplicating work that was done on a previous iteration through the design. In some cases, the reasons for making a decision may no longer be valid, and choosing a different alternative may be preferable. For example, one part may have been chosen over another because there was a surplus of them and the price was low. If this is no longer true, another part might be more cost effective. Rationale can also serve as a form of “corporate memory” providing valuable insight into a design that would otherwise be lost if designers leave the company (Peña-Mora and Vadhavkar, 1996, Brice and Johns, 1998).

In this paper, we describe InfoRat (Inferencing over Rationale), a system that inferences over a design’s rationale in order to detect inconsistencies and to assess the impact of changes. The analysis consists of two types of inferences: *syntactic* to inference over the “structure” of the rationale, and *semantic*, to look at the content.

The paper is structured as follows: in section 2, we discuss the problem of capturing, representing, and using design rationale. In section 3, we describe related work. Section 4 describes our approach, including a description of the sample problem used in the paper. Section 5 describes the implementation of the system and gives an example, while section 6 states our conclusions and plans for future work.

2. The Design Rationale Problem

Most work on design rationale has concentrated on *capture* and *representation*. Capturing, or recording, design rationale is a particularly difficult problem. Recording all decisions made, as well as those rejected, can be time consuming and expensive. The more intrusive the capture process, the more designer resistance will be encountered. Because it is time consuming and viewed as documentation, DR capture is viewed as expendable if deadlines are an issue (Conklin and Burgess-Yakemovic, 1995). Documenting the decisions can impede the design process if it is viewed as a separate process from constructing the artifact (Fischer, et al., 1995). Also, designers are reluctant to take the time to document the decisions they did not take, or took and then rejected (Conklin and Burgess-Yakemovic, 1995).

The representation of DR has also been studied extensively. Design rationale representations range from formal to informal. 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 provided to the system in a more rigid format. An informal approach provides data in formats that are easily generated and understand by a human but can not easily be used by the computer (e.g., natural language). Semi-formal approaches attempt to use the advantages of both approaches.

While capture and representation are important for design rationale, the real value of a system is how well the rationale can be put to *use*. Capturing large amounts of detailed rationale is not useful if it is never looked at again. If rationale is useful to the designers, there is a greater incentive for the designer to assist with the capture of the needed information, particularly if the designer who is recording it can immediately use the rationale. Also, knowing how the information will be used provides guidance about what information should be captured and how it should be represented. These are the key reasons why our research concentrates on DR use.

3. Related Work

How the DR can be used depends its representation format and content. Shipman and McCall (1996) describe three perspectives on design rationale: argumentation, documentation, and communication. Argumentation and documentation focus on the design decisions and the reasons behind them. The difference is that the goal of documentation is to convey understanding to people outside the project, while argumentation has the additional goal of structuring how the designer approached the problem. The communication perspective is an attempt to capture naturally occurring design communication, such as e-mail, meeting minutes, etc. Design Rationale can also be viewed as a design history – the sequence of events that occurred while performing the design (Garcia, 1993). In this case, the focus is more on what actions were taken over time and less on the reasons behind them.

Design Rationale representations vary from informal representations such as audio or video tapes, or transcripts, to formal representations such as rules embedded in an expert system (Conklin and Burgess-Yakemovic, 1995). A compromise is to store information in a semi-formal representation that provides some computation power but is still understandable by the human providing the information. Semi-formal representations are often used to represent argumentation.

There are several argumentation notations. Design Space Analysis (DSA) uses the Questions, Options, and Criteria (QOC) representation (MacLean, et al., 1991). This notation is used by Desperado (Ball, et al., 1999). QOC represents the argumentation as questions, options, and criteria for choosing the options. Another notation is Issue Based Information Systems (IBIS), used by gIBIS (graphical IBIS) and itIBIS (text based IBIS) (Conklin and Burgess-Yakemovic, 1995). IBIS represents the argumentation as issues, positions, and arguments. IBIS is the basis of another notation, PHI that is used in JANUS (Fischer, et al., 1995). PHI captures similar concepts to IBIS but links them together differently. There have also been many notations created for specific DR tools. Examples of this are DRCS (the Design Rationale Capture System) (Klein, 1992) and DRIM (Design Recommendation and Intent Model) (Peña-Mora, et al., 1995). DRCS represents argumentation using entities and claims about the entities. DRIM is used in SHARED-DRIM, which captures recommendation, justification, and intent for each participant in the design process. InfoRat bases its representation on DRL (Decision Representation Language), the representation used by SIBYL (Lee, 1990). DRL is described later in this paper.

There are also many different ways to capture DR. One approach is to build the rationale capture into a system used for the design task. Active Design Documents (ADD), a system that does routine, parametric design (Garcia, et al., 1993), uses rationale already built into a knowledge base and associates it with the user's decisions. Some systems capture DR by integrating the system into an existing design tool. This is done by M-LAP (Machine-Learning Apprentice System) (Brandish, et al., 1996). In M-LAP, user actions are recorded at a low level and formed into useful sequences using machine-learning techniques. This is also done in the RCF (Rationale Construction Framework) (Myers, et al., 1999). RCF uses its theory of design metaphors to interpret actions recorded in a CAD tool and convert them into a history of the design process. DHT (Design History Tool) (Chen, et al., 1990) is also integrated with a design tool and captures the history as a byproduct of the designing process. Some systems, such as itIBIS and gIBIS (Conklin and Burgess-Yakemovic, 1995) require that rationale be captured in a specific format, while others, such as HOS (Hyper-Object Substrate) (Shipman and McCall, 1996), use data captured informally during the design process and convert it into a useable form.

DR has a variety of uses. Systems such as JANUS (Fischer, et al., 1995), critique the design and provide the designers with rationale to support the criticism. Others, such as SYBIL (Lee, 1990), verify the design by checking that the rationale behind the decisions is complete. SYBIL also does some rationale evaluation. SHARED-DRIM (Peña-Mora, et. al, 1995) uses DR for conflict mitigation in collaborative design efforts. PTTT (Process Technology Transfer Tool) (Brown and Bansal, 1991) is used to transfer process design information between development and manufacturing. DME (Device Modeling Environment) (Gruber, 1990) is used to generate documentation "on demand" about electromechanical devices. C-Re-CS (Klein, 1997) performs consistency checking on requirements and recommends a resolution strategy for detected exceptions.

Less work has been done to study the usefulness of DR. Field trials were done using itIBIS and gIBIS for software development at NCR (Conklin and Burgess-Yakemovic, 1995). Capturing rationale was found to be useful during both requirements analysis and design. In particular, several errors were found during design that would not have been uncovered until much later when the code was written. IBIS also helped with team communication by making meetings more productive. A study was also performed using DR documents to evaluate a design (Karsenty, 1996). In this study, 50% of the designers' questions were about the rationale behind the design and 41% of these questions were answered by the recorded rationale. The rationale had been recorded manually using the QOC method.

4. Approach

In the InfoRat approach, design rationale is viewed as a bridge between design phases. The design begins with a set of *requirements* defining the system being designed. These requirements are then mapped to *goals* and, if required, sub-goals. Goals and sub-goals then can be satisfied by one or more *alternatives*. Each alternative then maps to an *artifact*, or a requirement for the next stage of design. The rationale for each choice is represented as arguments, expressed as *claims*, for or against each alternative. Figure 1 shows how design rationale links the requirements and the design.

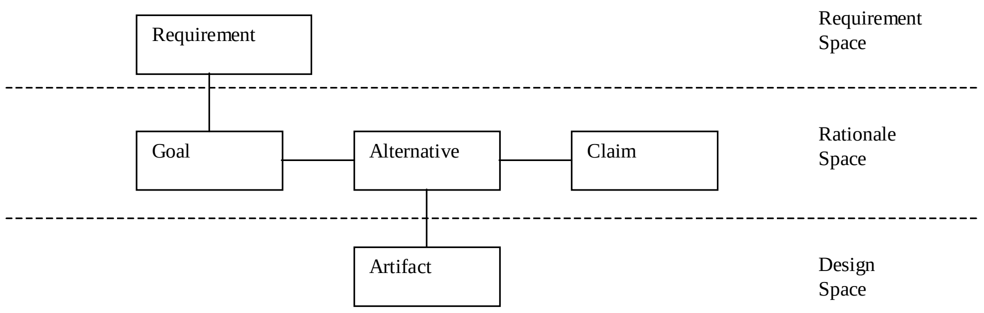


Figure 1. Design Rationale in the Design Process

The resulting rationale serves both to document the design and to provide a means for design verification. This verification involves ensuring that the design is consistent and complete, i.e., all requirements correspond to goals and all goals have selected alternatives. The following subsections describe the important aspects of this approach.

4.1 EXAMPLE PROBLEM

For illustrative purposes, a simple example of a traffic light design (Gogolla, 1998) was used. This was done to provide rationale that was simple to construct but rich enough to demonstrate the concepts. For more detailed information on traffic signal phase and cycle selection, see Zozayza-Gorostiza and Hendrickson (1987).

The traffic light example describes the high level design of the traffic lights for an intersection between two streets where one street had a heavier flow of traffic than the other, except during rush hour. This intersection also had frequent traffic turning from travelling South to travelling East. In addition to supporting those aspects of the intersection, the light system also had to be designed so that it would handle failure as safely as possible. Figure 2 shows the intersection.

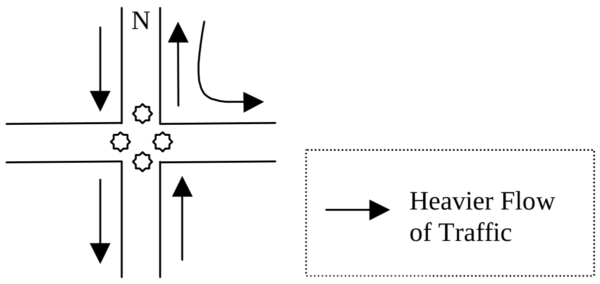


Figure 2. Intersection Diagram

This results in the following requirements for the traffic light system:

- Use four traffic lights
- Provide safe traffic flow
- Allow for heavier traffic on the North-South road
- Allow for traffic turning South to East
- Safely handle light failures

Each of these requirements can be satisfied in a number of ways. For example, choosing four traffic lights involves deciding what types of phases the lights should have, deciding if all four lights should be identical, and deciding if the lights should have arrows for turning or not. Providing safe traffic flow requires controllers for the lights to ensure that traffic can not be flowing on the E-W road at the same time that it is flowing on the N-S road. There are also a number of ways that the heavier traffic flow on the N-S road can be handled. Sensors can be used to monitor the flow of traffic or the lights can go to flashing yellow or red at times when traffic on the E-W road is lighter. Assistance for turning can be provided by delaying the lights or by using turn signals. There are also different ways that light failures that can be handled. One way is to shut down the intersection completely, although it might be better to turn it into a “four way stop” so that some traffic flow can still occur.

4.2 REPRESENTATION

As described above, there are a variety of methods for representing rationale. In order to support inferencing, a structured or semi-structured representation is required. DRL (Lee, 1990) has the richest rationale representation of the systems studied. A meaningful subset of DRL was chosen to allow exploration of possible inferences and to keep the representation relatively simple. For DRL, the elements represented are artifact, requirement, goal, alternative, claim, group, viewpoint, and question. DRL also supports several relations between these elements including: is-a-part-of, is-a-subclass-of, is-argument-for, and is-argument-against.

The InfoRat system implements a subset of these elements: requirement, goal, alternative, and claim. It also allows several relationships: supported-by, sub-goal, alternative-for, argument-for, and argument-against. Figure 3 shows the elements represented in InfoRat and the relationships between them.

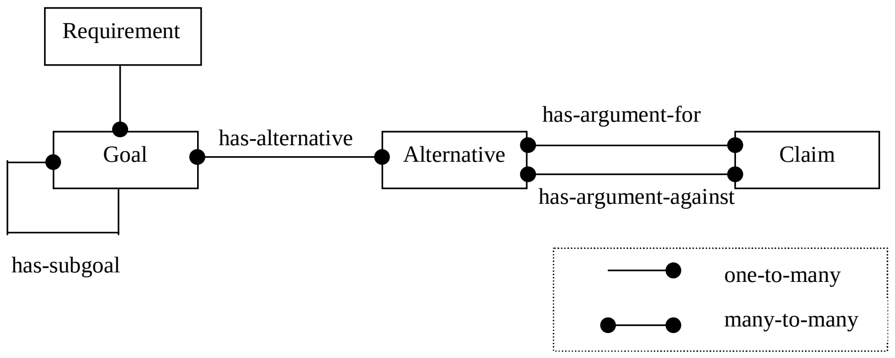


Figure 3. Design Rationale Elements

As the figure indicates, each goal can have multiple sub-goals, an alternative can be used to satisfy more than one goal, and a claim can be an argument for or against multiple alternatives. Figure 4 shows the goals as well as a partial set of alternatives and claims for the requirement to use four traffic lights.

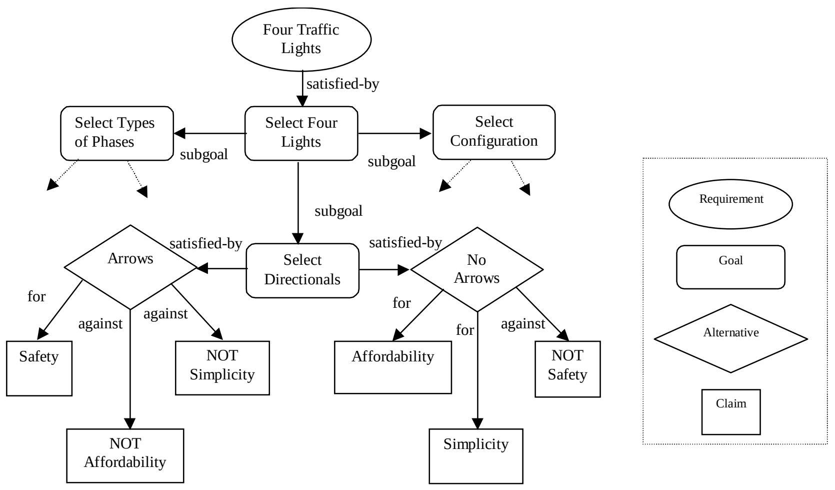


Figure 4. Subset of Alternatives for Requirement "Four Traffic Lights"

When a claim is used as an argument for or against an alternative, it is given a "rating" between one and ten to indicate its importance in the design decision. This weighting scheme was used because it is simple and easily understood by the designer. These ratings can be added together to indicate the overall rating for an alternative. For example, if the alternative "Arrows" (as shown in Figure 4) has a claim in its favor of "Safety", with a rating of seven, and claims against it of "NOT Affordability", with a rating of two and "NOT Simplicity", with a rating of one, its overall rating would be four.

4.3 INFERENCES

The InfoRat system inferences over the rationale to check for completeness and consistency. The inferencing can be broken into two categories: syntactic inferencing that uses the structure of the rationale, and semantic inferencing that looks at the contents/values of the different rationale elements.

Syntactic inferencing looks for the following inconsistencies in the rationale: requirements with no corresponding goals, and goals (or sub-goals) with no selected alternatives. The syntactic checks are primarily concerned with ensuring that the rationale is complete. Figure 5 shows the requirement "Four Traffic Lights" and its relationships. In this example, the goal "Select Type of Directionals" has two alternatives but neither has been selected. Figure 6 shows a syntactic check that looks to see if there are any requirements that do not trace to goals with selected alternatives. This check detects that the requirement "Four Traffic Lights" was not satisfied.

Both these figures, as well as those that follow, show actual output from InfoRat.

```
Requirement: Four Traffic Lights
      Goals:
         Goal: Select Four Lights
            Subgoals:
               Goal: Select Types of Phases
                  Alternatives:
                     German 4-Phase Lights
                     Italian 3-Phase Lights (Selected)
               Goal: Select Type of Directionals
                  Alternatives:
                     Light w/o Turn Signals
                     Light with Turn Signal
               Goal: Select Light Configuration
                  Alternatives:
                     Mixed Light Types (Selected)
                     All Lights the Same
```

Figure 5. Goals and Sub-goals for the Unsatisfied Requirement

```
***********************************************
*          Verify Design Rationale            *
***********************************************

Choose one of the following:

   1: Show Full Verification Report
   2: Check for Unsatisfied Requirements
   3: Check for Unsubstantiated Alternatives
   4: Check for Non-Optimal Alternatives
   5: Check for Contradictory Arguments

   E: Exit Menu

Enter Selection: 2

Unsatisfied Requirements:
   Four Traffic Lights
```

Figure 6. Unsatisfied Requirement Check

Semantic inferencing looks at the reasons for and against the alternatives. There are three types of discrepancies looked for: selected alternatives where the arguments against the alternative outweigh the arguments for the alternative, as shown in Figure 7, selected alternatives where the alternative selected is not the best choice, as shown in Figure 8, and selected alternatives where the same argument is used both for and against the alternative, as shown in Figure 9.

```
Arguments AGAINST outweigh FOR:

   For Goal: Priority to NS Traffic
      Selected Alternative: Configuration Changes w/Time
         (Rating = -3)
```

Figure 7. Arguments Against Outweigh For

```
Best Alternative not chosen for Select Light Configuration
   Selected Alternative: Mixed Light Types
         (Rating = 3)
   Best Rated Alternative: All Lights the Same
         (Rating = 5)

Best Alternative not chosen for Priority to NS Traffic
   Selected Alternative: Configuration Changes w/Time
         (Rating = -3)
   Best Rated Alternative: Sensor Controlled E/W
         (Rating = -2)
```

Figure 8. Best Alternative Not Chosen

```
Enter Selection: 5
Same argument for and against:
   For Goal:If EW traffic, no NS traffic
      Alternative: Individual Light Control
         Claim FOR:Safety and Claim AGAINST: Safety

   For Goal:If NS traffic, no EW traffic
      Alternative: Individual Light Control
         Claim FOR:Safety and Claim AGAINST: Safety
```

Figure 9. Contradictory Arguments

4.4 VOCABULARY

In order to support semantic inferencing, it is necessary to have a known vocabulary for claims (arguments for or against an alternative). The vocabulary consists of two categories: a pre-defined, standard vocabulary, and a user-defined, domain-oriented vocabulary. We refer to these as the Standard Claim Vocabulary and the User-Defined Claim Vocabulary respectively.

The Standard Claim Vocabulary is pre-defined to match the design task. For software design, a vocabulary has been built based on the “ilities” (Filman, 1998). Figure 10 shows the Standard Claim Vocabulary used by InfoRat.

```
Standard Arguments:

   Affordability
   Safety
   Availability
   Simplicity
   Reliability
   Adaptability
   Configurability
   Trustability
```

Figure 10. Standard Claim Vocabulary

Claims can be added to the User-Defined Claim Vocabulary at any time during the design process. These are arguments that are specific to the design project. Figure 11 shows the User-Defined Claim Vocabulary for the traffic light design problem.

```
User Defined Arguments:

   Starves one direction
   Optimizes Traffic Flow
```

Figure 11. User Defined Claim Vocabulary

4.5 RATIONALE PATTERN DETECTION

Because InfoRat supports semantic inferencing, there are several additional inference types that have been investigated and could be supported in future versions of InfoRat. Most, and perhaps all, of these types could be supported by looking for *rationale patterns*–sets of claims that frequently appear together when evaluating an alternative.

One such pattern is a *tradeoff*–a pattern of claims where one type of claim is being traded off against another. An example of a typical, well-known tradeoff would be cost versus strength. These tradeoffs can be detected by looking for frequent groups of claims where one subset of the group is for one alternative and another subset is for a competing alternative. This information, combined with background information indicating which claims have causal relationships with other claims, can be used to detect if the tradeoff appears to be balanced or if there is a preference for one claim, or set of claims, over another. If both claims, or sets of claims, are considered to be of equal importance, then a balanced approach would be desirable. If the requirements indicate that one is more important, then it should be preferred and any deviations from that preference can be shown as potential errors. For example, for a cost versus safety tradeoff, if safety is considered more important than cost, as it would be for the traffic light example, a selected alternative that preferred cost over safety might signal a possible error in reasoning.

Another possibility would be *co-occurrence patterns*–sets of claims that frequently occur together in an argument. For example, safety and trustability might occur together in evaluating alternatives. If trustability then occurs alone, InfoRat could indicate that the designer might want to look at safety for that alternative as well.

There may also be *modification patterns*–patterns in the original rationale version history. If there are commonalties in the rationale for portions of the design that have been changed, these patterns could be used to predict the location and likelihood of changes in new, similar designs.

By detecting these and other rationale patterns, InfoRat would be able to look deeper into the design rationale to detect more subtle inconsistencies than are supported by other evaluation systems.

5. Implementation and Examples

InfoRat has been implemented in CLIPS **(**CLIPS, 1998) and performs three main functions: Rationale Browsing, Rationale Modification, and Rationale Verification.

5.1 BROWSE RATIONALE

The browse function is used to examine the rationale stored in the system. The designer can examine the status of each element and its relationship. with other elements.

The first option, List DR Element Types, allows the user to quickly view the different DR elements currently in the system. Figures 12 through 14 show the element listings for requirements, goals, and alternatives.

```
Requirements:

   Four Traffic Lights (Satisfied)
   Safe traffic flow (Satisfied)
   Traffic heavier N-S (Satisfied)
   Frequent South to East Turning Traffic (Satisfied)
   Safely Handle Light Failures (Satisfied)
```

Figure 12. Requirement Listing

```
Goals:

   Select Types of Phases (Satisfied)
   Select Type of Directionals (Satisfied)
   Select Light Configuration (Satisfied)
   If EW traffic, no NS traffic (Satisfied)
   If NS traffic, no EW traffic (Satisfied)
   Safe Flow of Traffic (Satisfied)
   Priority to NS Traffic (Satisfied)
   Turn Assistance to SE Traffic (Satisfied)
   Select Four Lights (Satisfied)
   Stop all if Light Fails (Satisfied)
```

Figure 13. Goal Listing

```
Alternatives:

   German 4-Phase Lights
   Italian 3-Phase Lights (Selected)
   Light with Turn Signal (Selected)
   Light w/o Turn Signals
   All Lights the Same
   Mixed Light Types (Selected)
   Central Light Controller (Selected)
   Individual Light Control
   Blinking Red/Yellow
   Sensor Controlled E/W
   Configuration Changes w/Time (Selected)
   Turn Arrow for S->E (Selected)
   Delayed Green
   All Lights go to Blinking Red (Selected)
   All Lights go to Solid Red
```

Figure 14. Alternative Listing

The remaining options give the user a more detailed view of each element. Figure 5 (in Section 4) showed the information displayed about a requirement and its goals. Figure 15 shows the contents of an alternative, Blinking Red/Yellow.

Each rationale element contains a version number and a description of the element. The version number is used to keep track of changes in the rationale so that it can be determined if the state of any rationale element was changed during the design process. The description is used to describe the element to the user. InfoRat also allows the user to view the version history to see the changes made to the rationale and the reasons for the changes in the rationale. Figure 16 shows an example of a version history.

```
Alternative: Blinking Red/Yellow

         Alternative for:
            Priority to NS Traffic (Not Selected)

         Claims For:

            Claim: Simplicity
            Applicability: IS
            Weight: 3

            Claim: Affordability
            Applicability: IS
            Weight: 4

         Claims Against:

            Claim: Safety
            Applicability: NOT
            Weight: 7

            Claim: Starves one direction
            Applicability: IS
            Weight: 7
```

Figure 15. Alternative Blinking Red/Yellow

```
Version History:
   Version: 1

      Change: Removed claim [Safety] from
               [Configuration Changes w/Time]
        Reason: Duplicate Argument

   Version: 2
      Change: Removed claim [Affordability] from
               [All Lights the Same]
               Reason: Contradiction with another argument
 Version: 3
      Change: Added new Argument: [Optimizes Traffic Flow] for
               Alternative: [Mixed Light Types]
        Reason: Mixed lights can optimize flow

Version: 4
      Change: Removed claim [Safety] from
               [Individual Light Control]
        Reason: Individual lights are less safe (synch problems)

 Version: 5
      Change: Changed weight of argument [Optimizes Traffic Flow]
               to 5
        Reason: Traffic flow is very important
```

Figure 16. Version History

The first two changes were made in response to errors detected by InfoRat. The remaining three could either be triggered by the system or in response to changes in the requirements. Notice that the reasons given for the first two changes are reasons for changes to the rationale, not reasons for changes to the design.

5.2 VERIFY RATIONALE

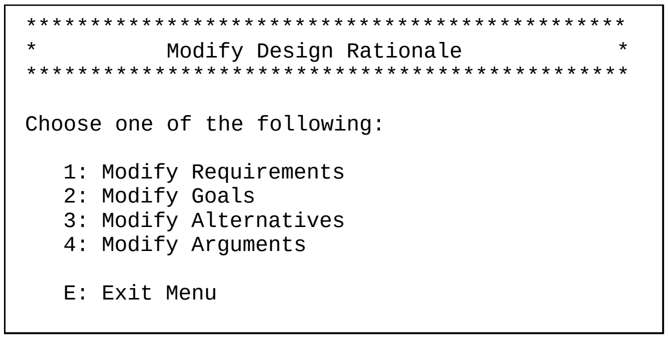
InfoRat verifies rationale by generating several different verification reports. The system can check for unsatisfied requirements (requirements that do not have goals associated or that have goals associated where the goals and their sub-goals do not map to selected alternatives), unsubstantiated alternatives (alternatives with a negative overall rating), non-optimal alternatives (when the alternative selected for a goal has a lower overall rating than one or more of the other alternatives for that goal), and contradictory arguments (arguments where the same argument is used for and against an alternative). InfoRat can also perform a summary check for all of these problems and produce a report. Figure 17 shows an example of a complete verification report.

```
Unsatisfied Requirements:
   None!

Arguments AGAINST outweigh FOR:
   For Goal: Priority to NS Traffic
      Selected Alternative: Configuration Changes w/Time
         (Rating = -3)

Best Alternative not chosen for Select Light Configuration
   Selected Alternative: Mixed Light Types
         (Rating = 3)
   Best Rated Alternative: All Lights the Same
         (Rating = 5)
Best Alternative not chosen for Priority to NS Traffic
   Selected Alternative: Configuration Changes w/Time
         (Rating = -3)
   Best Rated Alternative: Sensor Controlled E/W
         (Rating = -2)

Same argument for and against:
   For Goal: Select Light Configuration
      Alternative: All Lights the Same
         Claim FOR: Affordability and Claim AGAINST: Affordability
   For Goal: If EW traffic, no NS traffic
      Alternative: Individual Light Control
         Claim FOR: Safety and Claim AGAINST: Safety
   For Goal: If NS traffic, no EW traffic
      Alternative: Individual Light Control
         Claim FOR: Safety and Claim AGAINST: Safety
```

Figure 17. Full Verification Report

5.3 MODIFY RATIONALE

InfoRat allows the user to modify the different DR elements. Figure 18 shows the modification choices.



```
**********************************************
*          Modify Design Rationale           *
**********************************************

Choose one of the following:

   1: Modify Requirements
   2: Modify Goals
   3: Modify Alternatives
   4: Modify Arguments

   E: Exit Menu
```

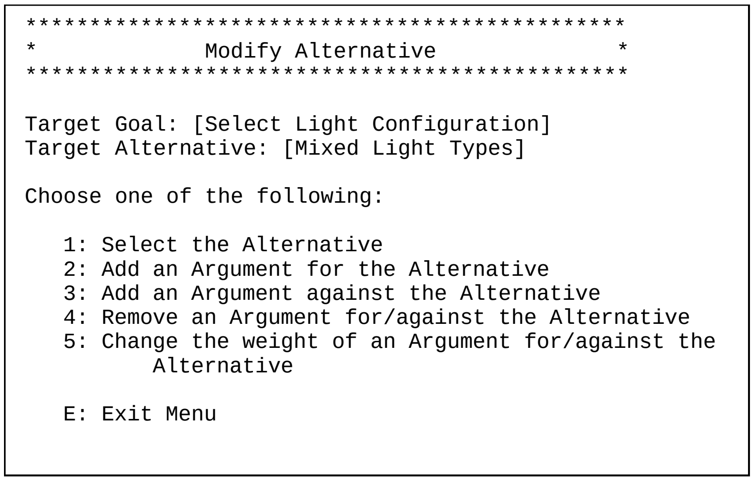
Figure 18. Modify Rationale Options

For requirements, the user is allowed to add a requirement, delete a requirement, or change which goals are associated with the requirement. Goals can either be associated or disassociated with the requirement. If a requirement is deleted, the delete cascades, i.e. any goals, sub-goals, and alternatives that only relate to this requirement are also removed.

For goals, the user can add a new goal or modify a goal already in the system. Allowable modifications for existing goals are adding a sub-goal, deleting a sub-goal, adding an alternative, removing an alternative, or selecting an alternative. When an alternative is selected, any alternative for that goal that may have been selected earlier is deselected to ensure that only one alternative can be selected for a goal.

For alternatives, the user again has the option of adding a new alternative or modifying an existing one. For an existing one, the user must first specify which goal the alternative is for. This is required because an alternative can apply to more than one goal. The user is then presented with several options for changing the arguments for and against the alternative. Figure 19 shows the options for modifying alternatives.

For arguments, the only option is adding additional arguments. When each modification is made, the user is prompted for a reason for the change. This provides additional information that can be retrieved by the user as part of the version history.



```
**********************************************
*             Modify Alternative             *
**********************************************

Target Goal: [Select Light Configuration]
Target Alternative: [Mixed Light Types]

Choose one of the following:

   1: Select the Alternative
   2: Add an Argument for the Alternative
   3: Add an Argument against the Alternative
   4: Remove an Argument for/against the Alternative
   5: Change the weight of an Argument for/against the
          Alternative

   E: Exit Menu
```

Figure 19. Modify Alternative Options

6. Conclusions

InfoRat supports a designer by inferencing over DR to check for completeness and consistency, as well as other problem indicators. This augments existing approaches, such as constraint satisfaction, that only reason about the design. Our work complements the work by Klein and Lee on reasoning over design rationale.

A predefined vocabulary is provided so that the contents of the arguments can be used for inferencing. The user can extend this vocabulary by adding additional arguments that are more design problem specific. When the user modifies the design rationale, the system prompts them for modification rationale. This combination of a standard, machine-interpretable vocabulary and user-supplied rationale allows the design history to be kept, and enables the system to reason over the rationale.

One drawback to InfoRat is that it does not eliminate the need to manually enter the DR. This, however, was not the focus of our research. Ideally, DR capture should be a byproduct of the design process, not a separate task that creates more work for the designer. One way to make this process easier is to integrate InfoRat with a design tool.

The target domain for InfoRat is software design. There are several points in the software design process where InfoRat could obtain information from software tools. These include the CASE tools used in software design to capture the initial DR elements, configuration management tools used to aid in capturing the design history, and problem reporting tools used to capture the reasons why the design required modification as well as what changes were made.

Besides tool integration, future work for InfoRat includes implementation of the rationale patterns previously described. Also, the representation needs to be extended to add the ability to form "groups'' of rationale elements, allowing InfoRat to scale to larger design problems as well as additional extensions needed to capture possible interactions between the alternatives. InfoRat also needs to be extended from supporting the initial high level design stage, as shown in this paper, to supporting multiple stages in the design process. In addition, more investigation is needed to see how InfoRat could be used to support teams of designers who may not agree on the claims for and against the alternatives. The integration of modification rationale with design rationale also needs attention. Finally, the user interface needs to be replaced with a Graphical User Interface (GUI).

The concepts developed in this work, as demonstrated by the InfoRat system, provide a new and different way of looking at DR use. Intelligent reasoning over DR will provide more beneficial use for the collected DR than just its retrieval and presentation. Such reasoning can provide strategic guidance for the design process. In addition it can provide a novel way of checking for design quality, as designs with poor rationale are less likely to be of high quality. We believe that this research provides a new view of how to use Design Rationale whose development has great potential.

Acknowledgements

We would like to thank George Heineman for his discussions about how design rationale could be used to support the software design process. The first author was supported by WPI's Robert S. Park Fellowship while performing this work.

References

Ball, L., Lambell, N., Ormerod, T., Slavin and S., Mariani, J.: 1999, Representing Design Rationale to Support Innovative Design Reuse: A Minimalist Approach, *from Proceedings of the 4th Annual Design Research Thinking Symposium*, MIT, May 1999.

Brandish M., Hague, M. and Taleb-Bendiab, A.: 1996, M-LAP: A Machine Learning Apprentice Agent for Computer Supported Design, *AID'96 Machine Learning in Design Workshop.*

Brice, A. and Johns, B.: 1998, Improving process design by improving the design process, QSL-9002A-WP-001, *QuantiSci*, October 1998.

Brown, D. C. and 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. and 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-Yakemovic, 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.

Filman, R. E. : 1998, Achieving Ilities, Workshop on Compositional Software Architectures, Monterey, California, Jan. 1998. http://www.objs.com/workshops/ws9801/papers/paper046.doc.

Fischer, G., Lemke, A., McCall, R. and 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.

Garcia, A., Howard, H. and 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, CA.

Gogolla, M.: 1998, *UML for the Impatient*, Research Report 3/98, Universität Bremen

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

Karsenty, L.: 1996, An Empirical Evaluation of Design Rationale Documents, in *Proceedings of the Conference on Human Factors in Computing Systems*, Vancouver, BC, April 13-18.

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.

Klein, M.: 1997, An Exception Handling Approach to Enhancing Consistency, Completeness and Correctness in Collaborative Requirements Capture, *Concurrent Engineering Research and Applications*, March. 1997.

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.

Myers, K., Zumel, N. and Garcia, P.: 1999, Automated Capture of Rationale for the Detailed Design Process, In *Proceedings of the Eleventh National Conference on*

Innovative Applications of Artificial Intelligence, AAAI Press, Menlo Park, CA, pp. 876-883.

Peña-Mora, F. and Vadhavkar, S.: 1996, Augmenting design patterns with design rationale, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 11, Cambridge University Press, pp. 93-108.

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

Shipman, F. and 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, Cambridge University Press, pp. 141-154.

Zozayza-Gorostiza, C. and Hendrickson, C.: 1987, An Expert System for Traffic Signal Setting Assistance, *ASCE Journal of Transportation Engineering*, 113(2), pg. 108-126.

CLIPS Reference Manual: 1998, *Volume I: Basic Programming Guide*, Version 6.10, http://www.ghgcorp.com/clips/download/documentation.