Automated Capture of Rationale for the Detailed Design Process

Karen L. Myers Nina B. Zumel
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com zumel@ai.sri.com

Pablo Garcia
Innovative Product Engineering and Technologies
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
pgarcia@unix.sri.com

Abstract

The value of comprehensive rationale information for documenting a design has long been recognized. However, detailed rationale is rarely produced in practice because of the substantial time investment required. Efforts to support the acquisition of rationale have focused on languages and tools for structuring the acquisition process, but still require substantial involvement on the part of the designer. This document describes an experimental system, the Rationale Construction Framework (RCF), that acquires rationale information for the detailed design process without disrupting a designer's normal activities. The underlying approach involves monitoring designer interactions with a commercial CAD tool to produce a rich process history. This history is subsequently structured and interpreted relative to a background theory of *design metaphors* that enable explanation of certain aspects of the design process. Evaluation of RCF within a robotic arm design case has shown that the system can acquire meaningful rationale information in a time- and cost-effective manner, with minimal disruption to the designer.

Introduction

Representations of designs in current-generation computer-assisted design (CAD) frameworks consist primarily of diagrammatic specifications, possibly augmented with simple annotations and *ad hoc* documentation. Even the most sophisticated systems lack much in the way of structured *design rationale*, despite the well-accepted view within the design community that such information would be a tremendous asset. Design rationale would serve as a record of the basic structure of a design, codifying how the design satisfies specified requirements, as well as key decisions that were made during the design process. By making it easier to understand how a design works, this information would facilitate collaboration among multiple distributed designers, improve the maintainability of designs, and enable more effective reuse of designs.

Despite the many benefits that explicit design rationale would provide, it is rarely produced in practice. Tools that support the specification of structured rationale by a designer have met with limited success because they either demand substantial designer time to enter information (Carroll & Moran 1991) or change the manner in which designers work (Conklin & Yakemovic 1991). Furthermore, there is little motivation for a designer to participate in such activities since the benefits surface downstream of his or her contribution. Recently, nonintrusive approaches have been explored that involve video or audio recording of design sessions (Chen, Dietterich, & Ullman 1991; Shipman & McCall 1997); however, a lack of structure in the produced representations hinders effective use of the results.

Given the tremendous value of structured design rationale but the unacceptable burden of constructing it manually, we were motivated to explore the use of AI methods for automatically and nonintrusively generating rationale. Our focus is on the *detailed design* phase, in which tools (*e.g.*, CAD systems, analysis packages) are used to generate a schematic for an artifact. This contrasts with the *conceptual design* phase, in which the scope and capabilities are set for the artifact to be designed. During conceptual design, the emphasis is on identifying and resolving high-level issues of functionality and requirements, with design rationale recording the justifications for the decisions that have been made. During detailed design, functionality and requirements are considered at much lower levels of abstraction. Choices and decisions are grounded primarily in physical constraints on components and the designer's insights into the composition of good designs.

The premise for our work was the observation that many of the operations that a designer can perform with modern CAD tools (*i.e.*, the *design substrate* (Fischer & Lemke 1988)) have meaningful semantic content. For example, CAD tools allow users to select objects with assigned semantic types from predefined libraries. This contrasts with most tools for designing software, where interactions are at the level of keystrokes. Nonintrusive monitoring of the actions taken by a designer with a CAD tool would thus provide a rich, semantically grounded process history for detailed design. AI techniques could be used to structure this information into representations that support improved user comprehensibility of the design process and automated inference about *designer intent*. Valuable reasoning methods would include *clustering* techniques to aggregate CAD operations into abstract summaries of designer activity, *plan recognition* to identify key episodes of activity, and *qualita-*

tive reasoning about the emerging design.

This paper describes the Rationale Construction Framework (RCF), which embodies the above ideas in a system that automatically constructs rationale information for the detailed design process. RCF records events and data of relevance to the design process, and structures them in representations that facilitate generation of explanations for designer activities. RCF operates in an opportunistic manner, extracting rationale-related information to the extent possible from observed designer operations. Within RCF, rationale is interpreted broadly to encompass any information that will further the understanding of a design and its development. This philosophy fits the *generative paradigm* (Gruber & Russell 1992) for rationale construction, which is intended to support general queries about a design and its evolution rather than answers to fixed sets of questions.

Extracted information is organized along two lines. The first is a series of hierarchical abstractions of the design history: *what* the designer did, and *when*. In addition, RCF reasons about designer intent – *why* the designer performed particular actions. Extraction of rationale related to intent is driven by a set of *design metaphors*, which describe temporally extended sets of designer operations that constitute meaningful episodes of activity. Design metaphors provide the basis for inferring intent on the part of the designer by linking observed activities to explanations for them.

Automatic generation of complete rationale for all aspects of a design is clearly infeasible. Certainly, designers make many critical decisions that are not explicit in the designs or the design process. The work reported here seeks to automate documentation of important but low-level aspects of the design process in a time- and cost-effective manner, thus freeing designers to focus their documentation efforts on the more creative and unusual aspects of the design. Ideally, the methods presented here would be complemented by interactive rationale acquisition methods that enable designers to extend and correct automatically generated information.

RCF was evaluated in a case study involving the design of a three-degree-of-freedom surgical robotic arm. The main technical challenge for this design was to provide sufficient actuation torque, while maintaining low inertia and high precision. The arm has three main subassemblies: the *base assembly*, including the actuation motors; the *arm assembly* with transmission; and the *wrist assembly*, including the end effector and tool. RCF recorded designer activity over several versions of the arm, starting from a rough initial design, through various stages of refinements and optimizations. From these recordings, RCF was able to summarize designer activity at varying levels of abstraction, to identify phases where the designer concentrated on various parts or subassemblies or where design parameters were tuned, to track the results of design tradeoffs, and to explain key design changes. The results validate the idea that meaningful rationale can be generated nonintrusively through application of appropriate AI techniques.

RCF Architecture

As depicted in Figure 1, RCF contains three main components: an enhanced CAD tool, the Monitoring module, and the Rationale Generation module (RGM).

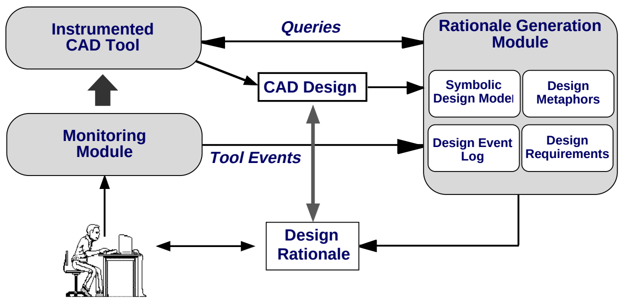

Figure 1: RCF Architecture

CAD Tool

The CAD system underlying RCF is MicroStation95, a commercial product that provides sophisticated modeling capabilities sufficient for a broad range of demanding electromechanical design tasks.

To support rationale acquisition within RCF, the set of operations provided by MicroStation95 was extended to include several capabilities that raised the overall semantic content of its design substrate. One class of added operations, *annotations*, enables users to specify the *semantic type* of an object along with corresponding type-specific *semantic attributes*. For example, the designer may declare that a given solid represents a gear, as well as specifying gear-specific attributes such as number of teeth, gear ratio, or quality. Such semantic information is a by-product of the parametric design methods and part selection capabilities found in numerous state-of-the-art systems. We also augmented MicroStation with a set of analysis programs that can be linked directly to components in the CAD drawing, thus extending the limited analysis capabilities within the off-the-shelf system. This modification reflects a growing trend toward building design environments that integrate a range of design tools. Finally, the ability to select components from predefined part libraries was added, which is standard in many CAD frameworks.

Monitoring Module

Within RCF, the designer interacts with the CAD tool as if it were a stand-alone application. The operations that he performs, however, are nonintrusively tracked by the Monitoring module, generating a stream of *tool events* that are sent in real time to the RGM. The monitoring facility captures only operations that are relevant to understanding the design process. Its coverage includes the creation, deletion, modification, and annotation of design objects, the creation and importing of parts, and the invocation of built-in analysis programs. Process-level commands such as the undoing and redoing of operations are also recorded. Aspects of the design process that are ignored by the monitoring module include certain geometric information associated with the manipulation and definition of objects (*e.g.*, spatial positioning), and viewing commands.

Base-level Design Events	
Create	define a new object
Copy	create a new object from an existing one
Delete	delete a previously created object
Modify	change structural aspect of an object
Connect	create a joint between two objects
Import	read in a predefined part
Annotate	set the semantic attributes of an object
Analyze	invoke a tool for analyzing some design aspect
Manipulate	reorient an object in space

Process-level Design Events	
Undo	undo the previous 'undoable' operation
Redo	redo the last undone operation
File-Open	create or read-in a design file
File-Copy	copy a design file
File-Save	save a design file

Figure 2: Design Events

Rationale Generation Module

The Rationale Generation Module (RGM), the main inferential component of the framework, is responsible for the automated generation of rationale structures. The RGM maintains an abstracted, tool-independent representation of observed CAD events, called the *design event log*. Based on the design event log, the RGM incrementally constructs a *symbolic model* of the emerging design. The design event log and symbolic design model provide the evolving information base from which rationale is generated, in conjunction with a formal specification of the *design requirements* for the given task and a set of *design metaphors*. This section describes the design events and the symbolic design model; design metaphors and requirements are discussed in subsequent sections.

Design Event Models The operations within the design event model, while not exhaustive, were chosen for their adequacy with respect to interesting design tasks. Two high-level categories of operations are defined (Figure 2). *Base-level* operations support the direct creation, modification, and manipulation of design objects. *Process-level operations* either manipulate information and metalevel structures related to the design (such as files), or impact the interpretation for previously executed operations (*e.g.*, *Undo/Redo* operations). Tracking the impact of the latter type of process-level operations requires complex bookkeeping of current and previous states to maintain an accurate characterization of the current design within the symbolic design model.

Several key properties are maintained for each design event. Type-independent properties include the corresponding tool events, timestamps, and affected objects. Type-specific properties are also stored; for example, an analysis event includes information about objects and their attributes used in the analysis, the analysis program invoked, and the results of the analysis.

Symbolic Design Model The symbolic design model provides an abstracted representation of the emerging design that supports the reasoning required by the rationale construction process. It excludes certain information stored within the CAD model (*e.g.*, geometric information plays only a limited role), but augments the CAD representation to include relevant process information for objects within the design.

Several categories of information are stored for each design object. First, there is the standard definitional information: an object's geometric category (*e.g.*, sphere, slab, line) and key *structural attributes* (*e.g.*, the diameter for a sphere). In addition, the *semantic category* and category-specific *semantic attributes* (if assigned) are stored. Because the specification of attributes can provide much insight into the evolution of a design, RCF maintains records of evolving attribute values (both structural and semantic) that enable retrieval for any stage of the design process. Several inter-object relationships are stored for use in reasoning about rationale, including *parent/child* relationships that reflect hierarchical structuring of complex objects, *copies/source* relationships, and *attachment* relationships indicating connection of two design objects through a *joint* of a designated type. We define an *assembly* to be the closure of a set of objects under the attachment relationship (*i.e.*, under joint connectivity).

On the process side, records are kept of all operations performed on an object, related analyses, time spent, origins data (*i.e.*, selected from a part library, copied from a user, created by a designer), and status information (*:alive* or *:inactive*, based on *Undo*, *Redo*, and *Delete* commands).

Rationale Extraction: Technical Approach

RCF generates two categories of rationale information: *session content* and *designer intent*. Rationale extraction is organized around a set of domain-independent *design metaphors* augmented by limited amounts of task- and domain-specific design knowledge.

Session Content RCF provides comprehensive summaries of a design session from two perspectives. The *object-centered* perspective provides historical and explanatory information for individual design objects and groups of objects that can either be explicit in the design (*e.g.*, assemblies) or *inferred* to be meaningful by RCF. The *event-centered* perspective summarizes the design session at multiple abstraction levels, using a combination of design metaphors and clustering methods to perform the abstractions.

Designer Intent A finished CAD model shows the endproduct of a designer's efforts but omits the *changes* that were made in the development of the design. Changes provide insight into the evolution of the design, showing alternative paths explored by the designer and basic strategies used. For

```
Devent 263: DELETE JOINT90
Devent 264: DELETE JOINT91
Devent 265: CONNECT JOINT92
Devent 266: CONNECT JOINT93
Devent 267: DELETE JOINT65
Devent 268: DELETE JOINT66
Devent 269: CONNECT JOINT94
Devent 270: CONNECT JOINT95
Devent 271: DELETE JOINT86
Devent 272: DELETE JOINT87
Devent 273: CONNECT JOINT96
Devent 274: CONNECT JOINT97
Devent 275: DELETE JOINT80
Devent 276: CONNECT JOINT98
Devent 277: DELETE JOINT81
Devent 279: UNDO
Devent 280: DELETE JOINT82
```

```
From design event 263 to design event 280
   Assemblies involved:
     MAIN-ASSEMBLY
     WRIST
     ARM
   Parts added:
     DOBJ229 (GEAR_16_3) from assembly WRIST
     DOBJ228 (GEAR_16_3) from assembly WRIST
   Parts removed:
     DOBJ183 (GR_08) from assembly WRIST
   Part substitutions:
     DOBJ167 (BV_1875) was replaced by DOBJ227 (BV_1250)
     DOBJ168 (BV_1875) was replaced by DOBJ226 (BV_1250)
     DOBJ212 (ARM3) was replaced by DOBJ225 (ARM4)
     DOBJ213 (ARM3) was replaced by DOBJ224 (ARM4)
```

Figure 3: Abstracting from Design Events to Part-level Operations

this reason, a key focus within RCF has been to identify and explain changes to a design. We have explored an approach that involves situating changes within contexts defined by *clustering* related events, and reasoning with *domain knowledge* about qualitative effects of design operations.

Design Metaphors

Design metaphors are multistep patterns of events (not necessarily contiguous) that describe episodes of coherent designer activity. They can be applied at varying scales of resolution: at the design event level, or over groupings or abstractions of events. RCF contains a suite of design metaphors whose recognition enables generation of defeasible explanations of a range of designer activity. Two example metaphors are presented here.

The *Refinement* metaphor is a cycle of *Analyze X - Revise* behavior, indicating that the designer is focusing on a particular design requirement *X*. It is reasonable to infer that intervening modifications to the design are performed with the goal of addressing *X*, although not all such revisions will have been performed with *X* in mind. Thus, while the metaphor does not definitively link action and intent, it does provide a plausible explanation for the designer's actions.

The *1:1 Part Substitution* metaphor captures the notion that the designer has swapped one functional component for another. In particular, Part *B* is considered to be substituted for Part *A* when it is observed that first, Part *A* is removed, and then some Part *B* from the same functional category is added to the assembly with the same connectivity as Part *A*. These part operations must occur within a certain window of activity but need not be consecutive.

The *Refinement* and *1:1 Part Substitution* metaphors capture general design principles and as such are applicable in most design applications. To date, all design metaphors within RCF are domain-independent. Task-specific design metaphors could readily be added to increase explanatory power, although at the cost of the knowledge engineering involved.

Task and Domain Knowledge

The use of background knowledge can greatly extend the rationale extraction capabilities. However, such knowledge can be difficult and expensive to acquire and represent. We explored a range of techniques that vary in the amount of background knowledge they require. Currently, RCF employs two kinds of optional background knowledge. First, overall *design requirements* are represented as a collection of properties, possibly with threshold constraints that must be satisfied (*e.g.*, $\textit{Arm-Inertia-}I_Z < 50\ \textit{lb-in}^2$). Nonmeasurable requirements, such as *Durability*, do not include explicit thresholds. Second, *qualitative models* of the effects of design operations can be used to generate deeper explanations of designer activity.

Event-centered Perspective

The event-centered perspective provides summarizations of a design session at varying levels of abstraction. Individual design events are grouped into *part-level operations*, which focus on design objects at the level of parts in an assembly. Next, operations are grouped into *activity phases*, which correspond to broader collections of activities with a common general design objective. Above that, phases are grouped into different *versions* for design components.

Part-level Operations The mapping from design events to part operations provides an abstracted view of the design process that is both more understandable to humans and more convenient for recognizing abstract design metaphors. Rather than examining activity on the level of features or components being modified or joined, a part-level chronology consists of parts being created, added or removed from the assembly, substituted for other parts, or modified.

Figure 3 shows a part-level abstraction produced by RCF. The excerpt from the design event log (on the left) constitutes a period of revision activity, in which the designer replaced certain components in the design. Within MicroStation95, a *joint* connects two design objects at a contact point; disconnecting a component from an assembly generally re-

```
>>>>>> Activity Phase 1: Type REVISION <<<<<<<<<<
In the BEGINNING, no detected focus
In the MIDDLE, MILD focus on (WRIST)
In the END, STRONG focus on (BASE)

>>>>>>> Activity Phase 2: Type REFINEMENT <<<<<<<
In the BEGINNING, focus on (ARM-INERTIA-IZ WEIGHT)
 Refinement of ARM-INERTIA-IZ:
     (60.0 50.0) --> REDUCE
   Modifications primarily on
     DOBJ92 (BASE_GEAR) from assembly BASE
     DOBJ207 (BV_1250) from assembly WRIST

 Refinement of WEIGHT:
     (2.2 2.0) --> REDUCE
   Modifications primarily on
     DOBJ92 (BASE_GEAR) from assembly BASE
     DOBJ207 (BV_1250) from assembly WRIST

In the END, focus on (STRESS)
 Refinement of STRESS:
     (0.6 0.4) --> REDUCE
   Modifications primarily on
     DOBJ31 (BV_0625) from assembly WRIST
     DOBJ207 (BV_1250) from assembly WRIST

>>>>>> Activity Phase 3: Type CONSTRUCTION <<<<<
In the BEGINNING, STRONG focus on (GROUP4 BEAR_BR)
In the MIDDLE, STRONG focus on (GROUP4 BEAR_BR)
In the END, MILD focus on (GROUP4)
```

Figure 4: Example Activity Phases with Attention Focus

quires a series of joint deletions. The part-level description on the right abstracts the explicit joint manipulations into a summary of the parts being removed, added, or replaced.

Note that each object within RCF has a unique *design object id, e.g.,* `DOBJ167`. Reference to design objects that are instances of parts generally include the name of the part from which it was created. Thus, `DOBJ212` and `DOBJ213` are both instances of the previously created part `ARM3`.

Activity Phases Activity phases are groups of events that describe designer activity at the level of abstract operations on components, parts, or the design artifact itself. Four types of activity phases are extracted from the sequence of part operations, with analysis and part-level statistics kept for each:

- *Construction:* a period of interleaved part creation and part addition events
- *Revision:* a period of interleaved part revision, addition, deletion, and/or substitution events
- *Analysis:* a period of analysis events not linked to any revisions
- *Refinement:* a period of related analysis and revision events

During a given activity phase, a designer will often focus on a specific part or set of parts for some time before switching attention to another aspect of the design. RCF identifies the evolving focus of designer attention, at the level of design requirements being addressed, individual parts, subassemblies of parts, and *implicit groups* of parts that are identified automatically during the extraction of activity phases (discussed below). The *focus* during an interval of activity is defined as a part, a grouping of parts, or a design requirement, for which the percentage of effort devoted to it exceeds some threshold. The effort metric used to determine attention focus can be either number of operations or accumulated time per part. Figure 4 displays summaries generated by RCF for a sequence of activity phases of different types, including the detected foci within each.

```
----------  Design object DOBJ207  ----------

 --- Source ---
 is standard part BV_1250 of type GEAR
 copied from file c:/models/version1/std_1250.dgn

 --- Implicit Constraints ---
 BV_1250 (GEAR), attributes (MATERIAL)
   constrains
 BV_0625 (GEAR), attributes (MATERIAL)

 --- Part Activity ---
 In Version 2, Activity Phase 1, type REVISION:
  >> added to assembly as replacement for BV_1875
    --> Effects: (ARM-INERTIA-IZ DOWN)(WEIGHT DOWN)

 Part modified
  Event 313: ANNOTATE MATERIAL ==> SS;
    --> Effects: (WEIGHT UP)(STRESS DOWN)
```

Figure 5: Example Part History

Versions Versions are episodes of activity that constitute a coherent set of changes on a design or one of its components. Within RCF currently, versioning is done at fairly coarse level, with version boundaries defined by the metaphor *Create/Copy File - Activity - Save File.*

Object-centered Perspective

For each design object, a detailed history is maintained that includes the object's origins, all related design events, and effort expended for that object. Additional information is kept for parts: related part-level operations, modification histories for the part and its various attributes, whether it was replaced by another part (*i.e.*, instances of the *1:1 Part Substitution* metaphor). The context of these operations is reported: which version and what type of activity phase. Also recorded are hypothesized explanations for those actions (see next section), detected relationships between a part and any design requirements, and inferred dependencies on other parts through membership in *implicit groups*.

The aggregation of objects into logically related groups provides a powerful mechanism for improving the understandability of complex structures. Assemblies and hierarchies provide examples of groupings that a designer defines explicitly. In addition, 'hidden' relationships can be present in a design that, if made apparent, could similarly improve understanding. For example, two parts may be implicitly dependent on each other, either structurally or functionally, in

OBJECT CLASS	ATTRIBUTE	TYPE	RANK	Precision Unloaded	Precision Loaded	Torque Avail	Inertia Arm	Friction	Max Stress	Cost	Durability	Resonance Mech.	Resonance Servo	Mass
Gears	Ratio	NUMERIC		+		+				+			+	+
	Radius	NUMERIC			+		+		+	+			-	+
	Diametral Pitch	NUMERIC		+					+	+				
	Quality	NUMERIC		+				-		+				
	Width	NUMERIC					+		+		+		-	+
	Ct-Ct Tolerance	NUMERIC		+				-		+				
	Adjustable Ct-Ct Dist	BINARY	TRUE	+						+				
	Material	ENUM	Lighter				-						+	-
			Yield						+	+	+			
			Stress						+		+			

Figure 6: Excerpt from the QIC Table for the Robotic Arm Design Case

a way that would not be apparent from examination of the finished CAD model.

RCF searches for *implicit groups* of design objects that satisfy such hidden relationships. While the system may or may not be able to identify precise constraints among the parts in an implicit group, it can bring them to the attention of a designer, who may be able to identify a reason for such a dependency. RCF contains design metaphors for recognizing several types of implicit groups, including (a) parts that are consistently added, removed, or revised together during the same period of construction activity, and (b) parts that are always modified together within a single refinement phase, in connection with the same design requirement.

Figure 5 displays an object summary for the part `bv_1250` (a gear), including information about its introduction into the design, subsequent modifications and their effects, and a detected implicit constraint between the `material` attributes of `bv_1250` and part `bv_0625`.

Explaining Changes

A key aspect of rationale is the linking of activity with intent. RCF reasons about designer intent during refinement phases, using a set of increasingly more knowledge-intensive methods. Specifically, the system gathers *evidence* to support a range of *hypotheses* as to why particular objects were modified and the effects that those changes had on the overall design. These hypotheses postulate that the designer intended (or not) to impact some design property (*e.g.*, *Torque*), possibly to move in some specified direction (*e.g.*, *increase* or *decrease*). A calculus for combining evidence is used to infer defeasible conclusions about designer intent.

To start, *change clusters* are generated that group related events. Analysis events provide the basis for cluster generation, in accord with the metaphor *AnalysisA...AnalysisK - Activity - AnalysisL....AnalysisN*. RCF hypothesizes that events within the scope of the cluster were performed to address one or more of the design requirements linked to the cluster analyses. These basic hypotheses can be bolstered (or countered) by additional evidence, such as detection of *refinement trends* whereby an analyzed design property is observed to change monotonically within a cluster. When available, quantitative knowledge about task-specific *thresholded design requirements* can be used to weaken or strengthen cluster-based hypotheses. For example, if a change occurs that increases the degree of satisfaction of a design requirement that is already known to be satisfied, it is less likely (but not impossible) that the designer was intentionally focusing on that design requirement.

Richer explanations of designer intent can be generated by using domain-specific background knowledge. We have developed an approach that involves reasoning qualitatively about the effects that actions have on design requirements. The possible effects are encoded in a *qualitative impact of change (QIC)* table. To date, only changes to semantic attributes are addressed, but the approach can be readily extended to handle structural changes (*e.g.*, the addition or deletion of objects). Figure 6 presents the gear portion of the QIC table for our robotic arm design case. *Numeric* attributes show positive or negative correlation with design properties. *Binary* attributes show correlation when the attribute assumes values of `true` or `false`. *Enumerated* attributes specify correlation for various ranking functions.

QIC information expands the space of both hypotheses and evidence for explaining designer activity. Figure 7 presents two example clusters. Each cluster includes (a) its key design events (here, analyses and changes to semantic attributes of objects), and (b) classification of effects of change operations within the cluster (extracted from the QIC table) as *intended* or *side* effects. Evidence to support the classification is provided: MATCHED-ANALYSES indicates multiple analyses of a particular design requirement, KNOWN-UNSAT indicates that a design requirement was known to be unsatisfied when a change was made that improved the design along that requirement, MODIFY indicates that the effect was brought about by a designated action, and MODIFY-WRONG-DIR indicates that the modification is contrary to the direction in which it should move (to satisfy some design requirement).

In the first cluster, QIC knowledge exists for only the change in material for `DOBJ92` from `Delrin` to lighter `Aluminum`. RCF uses it to conclude that the possible effects are to reduce `weight`, decrease `inertia`, or increase `stress`. From the collected evidence, RCF's evidential reasoning calculus infers that event 312 was performed to reduce `weight`, but also caused an undesirable increase in `stress` (as indicated by the high positive and negative likelihood values, respectively). These inferences bolster the hypothesis in the second cluster that the designer

```
>>>  Cluster #1  <<<
Key Design Events:
 #306: ANALYSIS of ARM-INERTIA-IZ with result 60.0
 #308: ANALYSIS of WEIGHT with result 2.2
 #310: ANNOTATE DOBJ209 (BV_1250)
         CAT_NUM --> BERG_M48N2A;
 #312: ANNOTATE DOBJ92 (BASE_GEAR)
         CAT_NUM --> F32-A6-96;
         MATERIAL --> DELRIN (was ALUMINUM);
 #313: ANALYSIS of ARM-INERTIA-IZ with result 50.0
 #314: ANALYSIS of WEIGHT with result 2.0

 Intended:
  ==> (WEIGHT DOWN) Likelihood = 3
     Evidence:
      (MATCHED-ANALYSES #308 #314)
      (MODIFY #312)
      (KNOWN-UNSAT #308)

  ==> (ARM-INERTIA-IZ DOWN)  Likelihood = 2
     Evidence:
      (MATCHED-ANALYSES #306 #313)
      (KNOWN-UNSAT #306)

 Side Effects:
  ==> (STRESS UP)  Likelihood = -5
     Evidence:
      (MODIFY-WRONG-DIR #312)
```

```
>>> Cluster #2  <<<
Key Design Events:
 #316: ANALYSIS of STRESS with result 0.6
 #318: ANNOTATE DOBJ209 (BV_1250)
         CAT_NUM --> BERG_M48N2A;
         MATERIAL --> SS (was ALUMINUM);
 #321: ANNOTATE DOBJ31 (BV_0625)
         CAT_NUM --> BERG_M48N3;
         MATERIAL --> SS (was ALUMINUM);
 #323: ANALYSIS of STRESS with result 0.4

 Intended:
  ==> (STRESS DOWN)  Likelihood = 3
     Evidence:
      (MATCHED-ANALYSES #318 #321)
      (PREVIOUSLY-COUNTERED Cluster-1)
      (KNOWN-UNSAT #316)

 Side Effects:
  ==> (WEIGHT UP)  Likelihood = -5
     Evidence:
      (MODIFY-WRONG-DIR #318 #321)
```

Figure 7: Sample Analysis Clusters with Evidence

is attempting to reduce `stress` through material changes in design events 318 and 321 (reflected in the evidence PREVIOUSLY-COUNTERED), even though doing so counteracts the `weight` decrease of the first cluster.

Evaluation and Discussion

The motivation for building RCF was to determine whether nonintrusive methods could extract useful rationale without disrupting the design process. Although no formal evaluation has yet been undertaken, results from the application of RCF to the robotic arm design case represent a qualified success: useful rationale was extracted and represented in structures that provide ready user accessibility. However, additional mechanisms will be required to produce a deployable tool for designers.

One area in which to extend RCF is to incorporate more design metaphors, with particular focus on identifying intent. For example, an *Exploration* metaphor would track branching of a design or design component into distinct alternatives, in contrast to the linear model of versioning employed currently within RCF. Such a metaphor would enable improved understanding of the space of options that a designer had considered before settling on his or her final choice. Versioning itself could be improved by grounding it in the clustering of related changes to an object, rather than relying on file operations to mark version boundaries.

The inspiration for RCF was the observation that many of the operations that a designer can perform with a CAD tool have meaningful semantic content. As CAD tools increase in sophistication, the set of semantically meaningful operations will also increase, thus enabling additional automated rationale extraction. For example, Active Catalogs (Ling *et al.* 1997) provides a rich query interface for selecting parts from online libraries, as well as simulation capabilities to support a "try before you buy" model of interaction. Observation of the queries formulated by a designer interacting with such a tool would yield a rich data stream from which additional rationale could be inferred.

There is always a trade-off in the design of knowledge-based systems between the cost of adding more knowledge (in terms of knowledge acquisition and maintenance) and the value that the added knowledge brings to the problem-solving process. We intentionally designed RCF with an *incremental* knowledge model that enables the system to run with varying levels of domain-specific knowledge. The main categories of such knowledge within RCF are design requirements and the QIC tables. The system can operate without this information, but generates increasingly better results as more of it is provided. For domains involving many one-off designs, development of extensive background theories will not be justified. However, for domains in which designs will be repeatedly produced, the application of domain-specific knowledge could greatly increase the extent of the rationale that can be generated.

Related Work

Early work on acquisition of design rationale focused on *direct solicitation* methods, whereby users are explicitly queried or provided with structured interfaces to elicit rationale (Russell *et al.* 1990; Gruber 1991). Because these approaches impose a heavy documentation burden on designers, they have had limited success.

Several attempts have been made to ease the problem of automated rationale generation by imposing structure on the design process (Ganeshan, Garrett, & Finger 1994; Brazier, van Langen, & Treur 1997). The general idea is to model design as selection from predefined transformation rules. When a rule is selected, the choice is recorded along with rationale associated with that rule. By structuring the design process, these approaches can provide deeper explanations of designer intent than can RCF. However, they greatly constrain designer activities and are unsuitable for *ad hoc* design cases requiring novel design methods.

The Active Design Document (ADD) system (Garcia *et al.* 1997; Garcia & Howard 1992) generates rationale for *parametric design tasks*, in which the design process involves constraining or selecting values for fixed sets of parameters. This class of designs, while important, is much narrower in scope than that addressed by RCF. The early ADD system observed the designer's actions, and then attempted to generate explanations for them. If the system was unsuccessful, or predicted an action other than that taken, it would ask for explanation. In this way, a knowledge base could be built from observed cases. More recent versions of ADD can generate ranked alternatives.

The Design History Tool (Chen, Dietterich, & Ullman 1991) stores structured, hierarchical representations of a design that are extracted from manually transcribed videotapes of design sessions that include what the designer *says* as well as the operations he performs. The resultant design history is browsable with respect to structure, evolution, alternatives considered, and dependencies in the design. This system provides better coverage of designer intent than does RCF, but at the costs of intrusiveness into the design process and labor-intensive data input.

Conclusions

RCF's methods for acquiring design rationale present an innovative approach to a difficult and important problem. Previous work on rationale acquisition has focused on highly intrusive techniques that either require extensive participation by the human designer or change the underlying design process. In contrast, our idea of extracting design rationale from observations of designer activity is rooted in a philosophy of nonintrusiveness: rationale is produced as a natural by-product of the design process. The human designer will need to intervene to supply certain information of relevance to the design but should be relieved of responsibility for recording information about the noncreative aspects of a design. We have shown within RCF that this type of automation is possible by applying key AI methods: knowledge representation, knowledge-based plan recognition, clustering techniques, and basic qualitative reasoning.

Acknowledgments This research has been supported by DARPA Contract N00014-96-C-0162.

References

Brazier, F. M.; van Langen, P. H.; and Treur, J. 1997. A compositional approach to modelling design rationale. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* 11:125–139.

Carroll, J. M., and Moran, T. P. 1991. Special issue on design rationale. *Human-Computer Interaction* 6(3-4).

Chen, A.; Dietterich, T. G.; and Ullman, D. G. 1991. A computer-based design history tool. In *NSF Design and Manufacturing Conference*, 985–994.

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

Fischer, G., and Lemke, A. C. 1988. Construction kits and design environments: Steps toward human problem-domain communication. *Human-Computer Interaction* 3:179–192.

Ganeshan, R.; Garrett, J.; and Finger, S. 1994. A framework for representing design intent. *Design Studies* 15(1):59–84.

Garcia, A. C. B., and Howard, H. C. 1992. Acquiring design knowledge through design decision justification. *Artificial Intelligence for Engineering, Design, and Analysis in Manufacturing* 6(1):59–71.

Garcia, A. C. B.; de Andrade, J. C.; Rodrigues, R. F.; and Moura, R. 1997. ADDVAC: Applying active design documents for the capture, retrieval, and use of rationale during offshore platform VAC design. In *Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence*, 986–991.

Gruber, T. R., and Russell, D. M. 1992. Generative design rationale: Beyond the record and replay paradigm. Technical Report KSL 92-59, Knowledge Systems Laboratory, Computer Science Department, Stanford University.

Gruber, T. R. 1991. Interactive acquisition for justifications: Learning 'why' by being told 'what'. *IEEE Expert* 6(4).

Ling, S. R.; Kim, J.; Will, P.; and Luo, P. 1997. Active catalog: Searching and using catalog information in internet-based design. In *Proceedings of the ASME Design Engineering Technical Conferences*.

Russell, D. M.; Burton, R. R.; Jordan, D. S.; Jensen, A. M.; Rogers, R. A.; and Choen, J. 1990. Creating instruction with IDE: Tools for instructional designers. *Intelligent Tutoring* 1(1):3–16.

Shipman, F. M., and McCall, R. J. 1997. Integrating different perspectives on design rationale: Supporting the emergence of design rationale from design communication. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* 11:141–154.