

A Generic Model for Reflective Design

PANAGIOTIS LOURIDAS and PERICLES LOUCOPOULOS

University of Manchester Institute of Science and Technology

Rapid technological change has had an impact on the nature of software. This has led to new exigencies and to demands for software engineering paradigms that pay particular attention to meeting them. We advocate that such demands can be met, at least in large parts, through the adoption of software engineering processes that are founded on a reflective stance. To this end, we turn our attention to the field of Design Rationale. We analyze and characterize Design Rationale approaches and show that despite surface differences between different approaches, they all tend to be variants of a relatively small set of static and dynamic affinities. We use the synthesis of static and dynamic affinities to develop a generic model for reflective design. The model is nonprescriptive and affects minimally the design process. It is context-independent and is intended to be used as a facilitator in participative design, supporting group communication and deliberation. The potential utility of the model is demonstrated through two examples, one from the world of business process design and the other from programming language design.

Categories and Subject Descriptors: D.2.1 [**Software Engineering**]: Requirements/Specifications—*Elicitation methods* (e.g., rapid prototyping, interviews, JAD); D.2.2 [**Software Engineering**]: Design Tools and Techniques

General Terms: Design, Human Factors, Management

Additional Key Words and Phrases: Design aids, design rationale, development, participative, reflective

1. INTRODUCTION

The nature of computer systems has been steadily changing. The primary role of information technology is no longer to automate (speeding up on existing tasks), but to informate (redefine work using technology as an enabler) [Zuboff 1988; Landauer 1995]. This has led to new exigencies and to demands for software engineering paradigms that pay particular atten-

This work was partially supported by the Alexander S. Onassis Public Benefit Foundation, Grant R-040/1995–1996.

Authors' address: Information Systems Engineering Group, Department of Computation, University of Manchester Institute of Science and Technology, UMIST Box 88, Manchester, M60 1QD, U.K; email: panos@co.umist.ac.uk; pl@co.umist.ac.uk.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

© 2000 ACM 1049-331X/00/0400–0199 \$5.00

tion to meeting them. Researchers and industry luminaries have highlighted the parallels between software, on the one hand, and architecture and design, on the other, and have argued that software development should become more cognizant of theories and practice from the field of Architecture and Design [Winograd and Tabor 1996]. Design patterns are an offshoot; design studios in computer science curricula are another.

Within Architecture and Design there is a substantial body of opinion that expresses disillusionment with the use of methodology, e.g., Alexander [1971] and Jones [1977]. A recurring element in this criticism is that what is most important is the designers' quality of practice and not the methods that they may adopt and follow.

This criticism suggests a search for approaches that pay particular attention to the way that the design process itself can be brought into focus. Such approaches attempt to enhance design by promoting a more conscientious and receptive attitude on the designers' part. The main advantage to be gained, whether practiced in isolation or within some methodological framework, is the increased awareness of the situation gained by exposing and bringing forth assumptions, reasoning, and points of view that would otherwise remain covert.

We agree that what is most important is the designers' attitude and the quality of their practice in itself, independently of the method followed, if any. Information systems development methods comprise guidelines, procedures, techniques, notations, models, and tools, which form the bulk of much Software Engineering research and practice. Without rejecting it, we propose a *complementary* research approach, to focus on the quality of development practice per se. Our work focuses not on ameliorating the design process by defining or altering it, *but by aiding it through a fostering of the appropriate reflective stance* [Schön 1983]. This is achieved through the provision of a conceptual framework for capturing, making models of, using, and reasoning on the systems development process, thus promoting a metadesign process, where the design process itself is brought into focus.

This objective is also shared by approaches in the field of Design Rationale [Moran and Carroll 1996]. Design Rationale aims at capturing the *why* behind the *how* in design, i.e., on externalizing and documenting the reasons behind design decisions and artifact features. In brief, Design Rationale aims for a careful and open consideration of the reasons behind the decisions taken and of the background behind the development process, by capitalizing precisely on this background. Design Rationale has produced a large body of research and a number of different approaches have been proposed with the aim to (among others): facilitate group discussions during cooperative design sessions; provide explanations about particular designs, the alternatives available, the arguments used for the choices made and the evaluation of different designs; discover generalized patterns of design for reusing them in similar settings.

The concern of Design Rationale approaches lies close to our concern of promoting reflectiveness; Design Rationale approaches, however, also comprise other extraneous characteristics stemming from the specific applica-

tions for which they were developed and their developers' preferences. We aim at a *generic* model for reflective design of potentially wide applicability. Through a systematic analysis we deconstruct existing Design Rationale approaches and derive their commonalities in terms of structure and process. In short, we devise a context-independent aid for reflective design by abstracting and synthesizing from a number of more specific purpose models in the same area.

In Section 2 we present various Design Rationale approaches, along with comments on general characteristics and trends. In Section 3 we use the approaches presented in order to develop a conceptual aid for participative design processes. Specifically, in Section 3.1 we abstract the semantic affinities of the approaches, and in Section 3.2 we abstract their operational affinities; in Section 4 the two sets of affinities are combined in a single model for reflective reasoning. Excerpts of real-world examples of its use are presented in Section 5. Space limitations dictate that only small parts of the applications can be included in this paper. However, the examples demonstrate the usability of the proposed model and its applicability to two diverse design activities. The choice of the examples from the two diverse domains is also intended to demonstrate the generality and context-independence of the approach. Discussion and conclusions follow in Section 6.

The proposed approach is intended to be used in complex problem solving settings, with various stakeholders, no clearly defined initial state, no clearly defined final state, and no clear way to get there. Information technology change management initiatives share these characteristics, as do many activities in Information Systems Development projects.

2. REFLECTIVE DESIGN THROUGH DESIGN RATIONALE

2.1 Design Rationale

Design Rationale (DR) can be defined as follows [Moran and Carroll 1996]:

- (1) An expression of the relationships between a designed artifact, its purpose, the designer's conceptualization, and the contextual constraints on realizing the purpose.
- (2) The logical reasons given to justify a designed artifact.
- (3) A notation for the logical reasons for a designed artifact.
- (4) A method of designing an artifact whereby the reasons for it are made explicit.
- (5) Documentation of (a) the reasons for the design of an artifact, (b) the stages or steps of the design process, (c) the history of the design and its context.
- (6) An explanation of why a designed artifact (or some feature of an artifact) is the way it is.

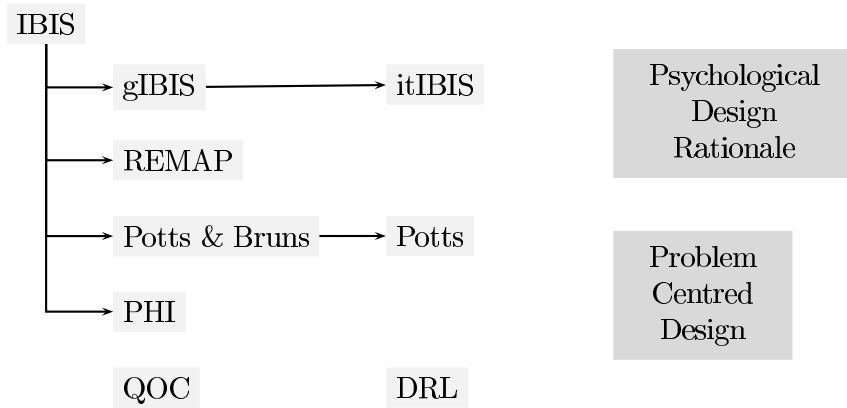


Fig. 1. DR approaches.

According to this definition, the DR research field comprises all research pertaining to the capture, recording, documentation, and effective use of the rationale in development processes. A complete record, e.g., by video, of the whole development process, combined with any materials used and produced, could, in theory, be used to find the rationale behind the decisions taken. Since this is unwieldy, the main idea in DR research is that the rationale can be structured by providing a proposed formalism using a small set of concepts appropriate for representing the deliberations taking place.

The definition of Design Rationale given above is sufficiently broad to include a large number of approaches, not all of which are relevant to our purposes. Figure 1 shows a kind of genealogical tree of the approaches considered in this paper (the two shades of gray correspond to the distinction between metamodel-based and non-metamodel-based approaches that is explained in Section 3.1):

- Issue-Based Information System (IBIS)
- Representation and Maintenance of Process Knowledge (REMAP)
- Potts and Bruns, and Potts
- Procedural Hierarchy of Issues (PHI)
- Questions, Options, and Criteria (QOC)
- Decision Representation Language (DRL)
- Psychological Design Rationale (PDR)
- Problem-Centered Design (PCD)

2.2 The Approaches

2.2.1 Issue-Based Information System. Issue-Based Information System (IBIS) [Conklin and Begeman 1988] was the progenitor of most approaches.

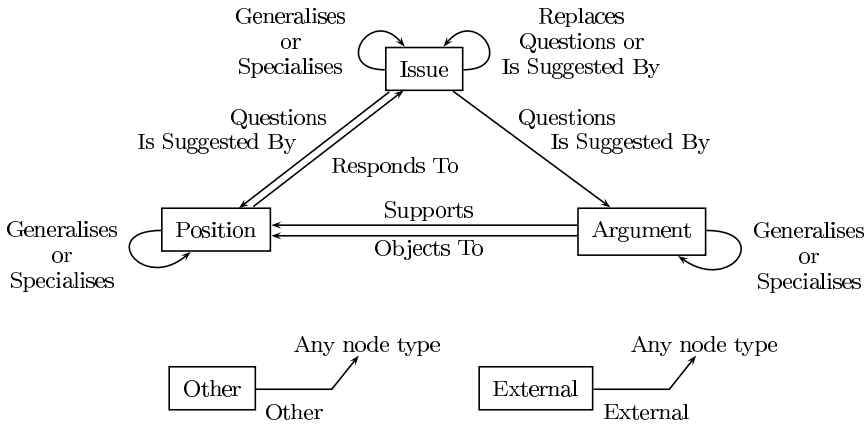


Fig. 2. The gIBIS metamodel.

It was the first DR approach to be proposed, and still the point of reference for most of the research in the field. IBIS aims primarily at improving deliberation sessions among the various stakeholders in the design process. In Figure 2 we see a variant of IBIS, gIBIS (for graphical IBIS), which has been used in an influential hypertext implementation. IBIS is structured along the following entities:¹

- Issues that represent anything that needs to be discussed, deliberated, and put into argumentation during a design project.
- Positions that are ways of resolving Issues.
- Arguments that are statements supporting or objecting to Positions.

These can be connected to each other by a number of different links representing specific relationships among them. The use of IBIS results in the construction of a highly interconnected network of Issues, Positions, and Arguments. The procedure starts when one of the stakeholders “posts” the *root* Issue of an IBIS *tree*. An IBIS tree is an Issue with all the corresponding Positions and Arguments. This is constructed incrementally, as stakeholders “post” Positions related to Issues, Arguments corresponding to Positions, in general any IBIS concept that can be linked to existing ones by the relationships provided. When new Issues, unconnected to an existing tree, are posted, we obtain a *forest*. This process goes on until a consensus regarding the resolution of the issues under consideration has been reached among the stakeholders.²

¹The main difference between gIBIS and the original IBIS metamodel is the existence of the *Other* and *External* entities. *External* provides the link to external artifacts, and *Other* covers the cases not subsumed by the other entities.

²The graphical aspect of the IBIS process is not essential. A variant of IBIS, itIBIS (for indented text IBIS), uses only indented text for the representation of IBIS trees and their aggregates, i.e., forests.

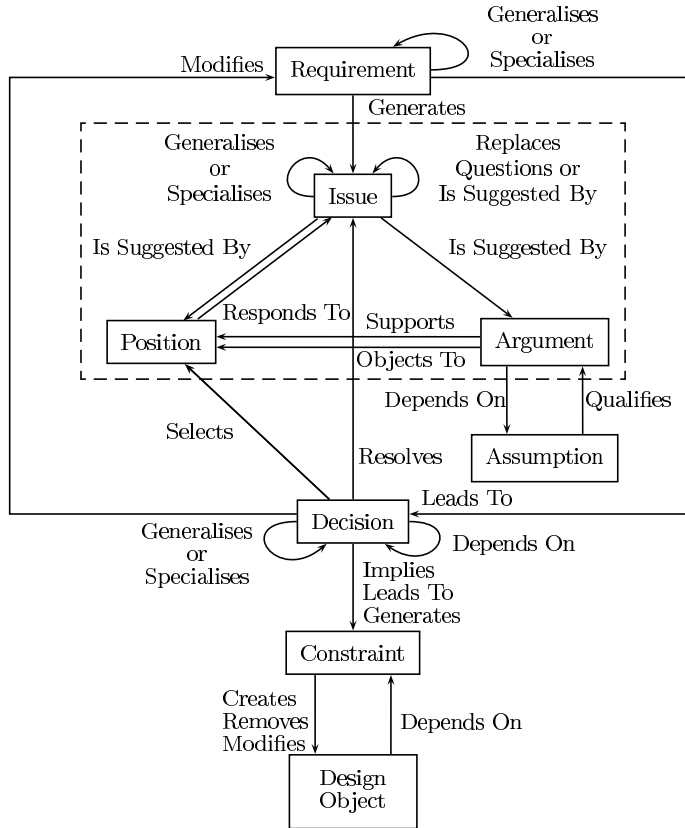


Fig. 3. The REMAP metamodel.

2.2.2 Representation and Maintenance of Process Knowledge. The Representation and Maintenance of Process Knowledge (REMAP) approach is similar to that of IBIS, but with additional entities and relationships which render it semantically richer [Ramesh and Dhar 1992]. This can be seen in Figure 3 (the part of the figure enclosed in dashed lines is the original IBIS metamodel). Specifically, the following additional entities are introduced:

- Requirements that are the goals that guide the design process by generating specific issues.
- Decisions that are the actions by which a resolution is reached and a Position is selected.
- Constraints that represent commitments, created by Decisions, that have to be satisfied.
- Design Objects that represent the artifacts under consideration.
- Assumptions that provide the context in which an Argument is considered to be applicable.

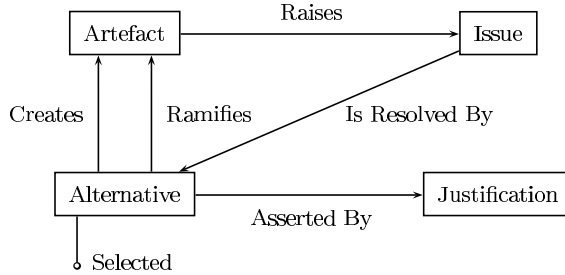


Fig. 4. The Potts and Bruns metamodel.

The number of allowed relationships among the entities proposed by REMAP rises proportionally.

The REMAP way-of-working³ is similar to the one proposed by IBIS. However, whereas IBIS can be used with minimal or no computer support, REMAP is too complex to be employed in this way. It was designed to be embedded in computer implementations offering temporal reasoning capabilities, deductive rules, and integrity constraints and rules.

2.2.3 Potts and Bruns' Approach. The Potts and Bruns approach to DR [Potts and Bruns 1988] was developed to answer questions regarding the tracing of artifact evolution in the design process. It aims at constructing a projection of the artifact's path in relation to the issues raised in its development and the deliberations on them. The Potts and Bruns metamodel (depicted in Figure 4) is structured along the following entities:⁴

- Issues that are similar to Issues in IBIS but are supposed to be raised only by artifact features.
- Alternatives that resolve features.
- Justifications that provide the grounding for Alternatives.
- Artifacts that represent the objects of the design process.

A restricted set of relationships among the entities is also provided.

A session with the Potts and Bruns metamodel starts with an existing Artifact (e.g., an informal wish-list). An Artifact can raise a number of different Issues. Different Alternatives may then be proposed for the resolution of an Issue. Some of the Alternatives may suggest the creation of new Artifacts (in this example, a functional specification); others may demand modifications to existing ones. The Alternatives are supported or challenged by Justifications. When an Alternative is

³A *way-of-working* is a way for carrying out a task. Under the definition of methods given in Section 1 a way-of-working is not a method, but a method may contain various ways-of-working.

⁴A slightly different version has also been presented [Potts 1996], where the terminology is closer to IBIS. We stick to the original presentation here.

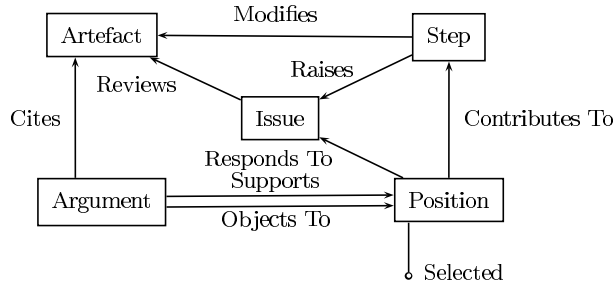


Fig. 5. The Potts metamodel.

selected, a new *Artifact* is derived (either a completely new one, or a modified version of an existing one) that may raise more *Issues*, and so on.

2.2.4 Potts. The Potts approach [Potts 1989] is an evolution of the Potts and Bruns approach presented above. In Figure 5, a number of differences with its ancestor can be seen: it follows IBIS more closely and it introduces a specific entity for capturing the design actions. The following entities are employed by the approach:⁵

- Issues that are raised by design Steps.
- Positions that resolve Issues.
- Arguments that support or object to Positions.
- Artifacts that represent the objects of the design process.
- Steps that represent actions in the design process.

The way-of-working proposed is similar to the one outlined for the Potts and Bruns approach, although implicit actions are externalized by means of the *Step* concept.

2.2.5 Procedural Hierarchy of Issues. The Procedural Hierarchy of Issues (PHI) approach [McCall 1991], shown in Figure 6, adopts the IBIS conceptual structure, its differences with the latter laying mainly in the specific process model employed, and not in the intrinsic properties of the entities provided. The following entities are used:

- Issues that have a broader definition than in IBIS: in the latter, an *Issue* requires deliberation; in PHI it represents any design problem, whether it requires deliberation or not.
- Answers that are alternative ways of resolving Issues.
- Arguments that provide the grounding for selection among Answers.

⁵An updated version of the approach has also been proposed [Potts 1996]. The differences, though, are minor: for example, instead of the *Step* entity a specialized *Refinement* entity is proposed; we keep the more general one here. Also, a catch-all *Comment* entity is added.

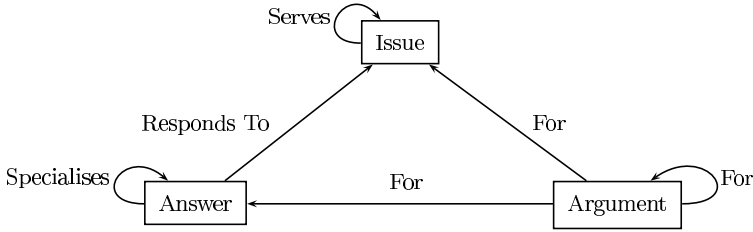


Fig. 6. The PHI metamodel.

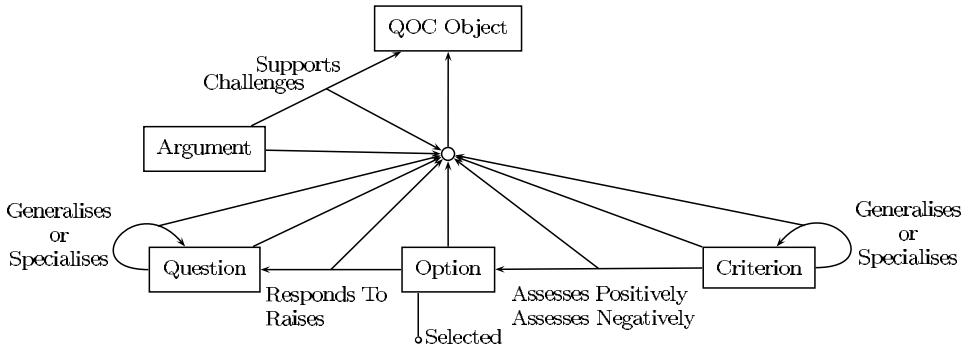


Fig. 7. The QOC metamodel.

The relationships proposed by the PHI metamodel form a minimal set; this reflects one of the approach’s tenets, that the metamodel should be as simple as possible.

PHI is the only DR approach to advocate a specific process model in the sense of algorithmic steps to be followed; in order to deliver good results, the design activity should be structured. The process begins with an Issue (called the *Prime Issue*, phrased as “What should this design be?”), which only refers to the project without defining it (since the definition is part of the problem). Then this Issue is decomposed to sub-Issues in a top-down breadth-first manner. A tree hierarchy is produced, in whose leaves Answers may be posted. Arguments can be linked either to Answers, or directly to Issues during the decomposition process.

2.2.6 Questions, Options, and Criteria. Questions, Options, and Criteria (QOC) [MacLean et al. 1991] has as main objective the discussion of alternatives on specific artifact features; in essence, to provide a structured representation of the design space to which an artifact belongs (this space includes all the alternative possibilities and options arising in the process of design together with the criteria and the considerations used for selecting among them). To achieve this objective, the following entities and a set of relationships are proposed (see Figure 7; the circle-arrow construct represents is-a relationships):

—Options that are artifact features.

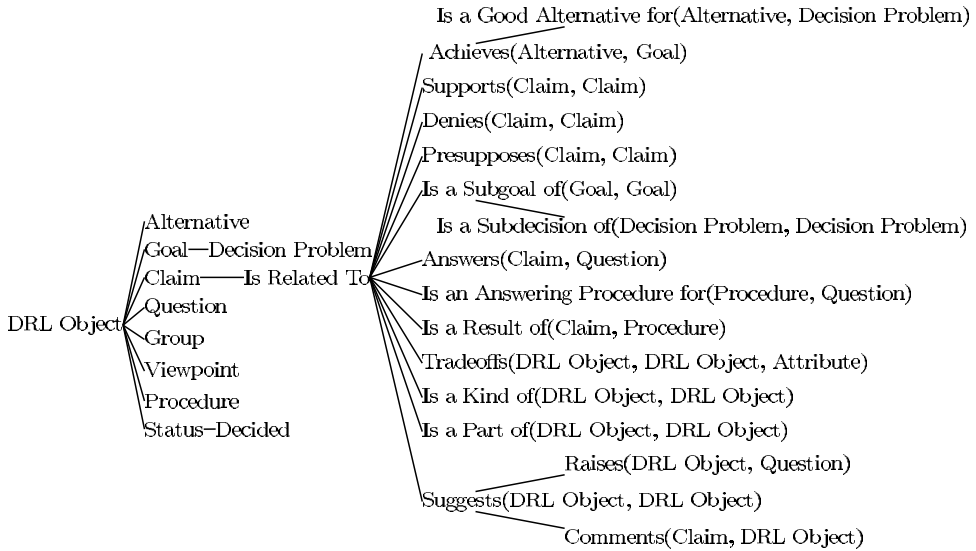


Fig. 8. The DRL specialization hierarchy.

—Questions that are means of organizing the various Options, since every artifact feature responds to a specific design issue that can be framed as a Question.

—Criteria that are used to determine the choice between Options; equivalently, they can be seen as requirements (goals) that have to be accomplished.

—Arguments that can support or object to entities in QOC models.

A QOC process starts by establishing problems (Questions) impinging on an artifact. It then proceeds by establishing Options addressing Questions. Options are evaluated and selected in terms of specific Criteria describing the requirements (goals) to be fulfilled. Arguments can be attached to any element in an ongoing QOC discussion.

2.2.7 Decision Representation Language. As indicated by its name, Decision Representation Language (DRL) [Lee and Lai 1991] was developed as a language for representing decision making. Computer usability was a primary concern; this entailed increased expressiveness and functionality but also increased complexity, as can be seen in Figure 8.⁶ All links in Figure 8 represent is-a relationships; the descendants of the Is Related To object are objectified relationships, with the related objects in parentheses. From Figure 8 and the presentation of DRL in the literature, the basic entities are:

⁶Although it would be advantageous, for comparison purposes, to present DRL in the arrows-and-boxes format used above, the attempt results in very cluttered diagrams; hence the original representation, as presented in Lee and Lai [1991], was kept.

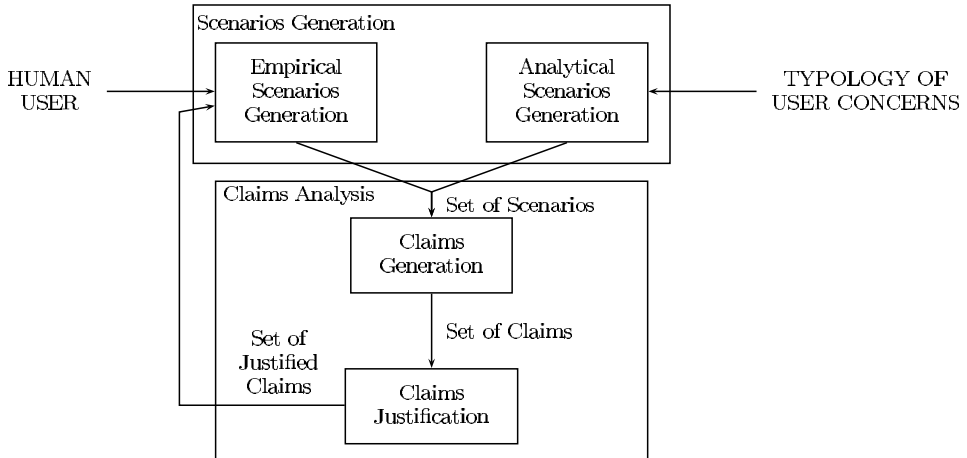


Fig. 9. The PDR way-of-working.

- Decision Problems and Goals; a Decision Problem is a design issue and a Goal, since implicitly in its definition lies the requirement (goal) of resolving it.
- Alternatives representing possible solutions to Decision Problems.
- Claims that are the means for argumentation; since all relationships are subclasses of the Claim object, argumentation can be built on top of other argumentation.

The rest of the entities (not the objectified relationships) in DRL are of secondary importance; only Questions present interest, since they are the means for carrying out discussions on the design process per se, i.e., metadiscussions. As for relationships, albeit objectified, conceptually they are still treated as relationships; their rendering as objects affects their capacity for being related upon and not their semantics. DRL was embedded in a decision rationale tool called SIBYL. In SIBYL the user starts by creating instances of a Decision Problem, and then proceeds by entering Goals and Alternatives. Evaluation and argumentation ensue, supported by appropriate modules.

2.2.8 Psychological Design Rationale. Psychological Design Rationale (PDR) is not as much a language or representation for documenting DR, as an approach for design in which the recording of a special kind of rationale (i.e., psychological) plays a key role [Carroll and Rosson 1991; 1992]. The PDR way-of-working is shown in Figure 9; it consists of a repeating loop of the following structure:

- Scenarios Generation, i.e., the creation of use scenarios of the envisaged artifact. This can be done in two ways:
 - Analytically (Analytical Scenarios Generation), where scenarios are

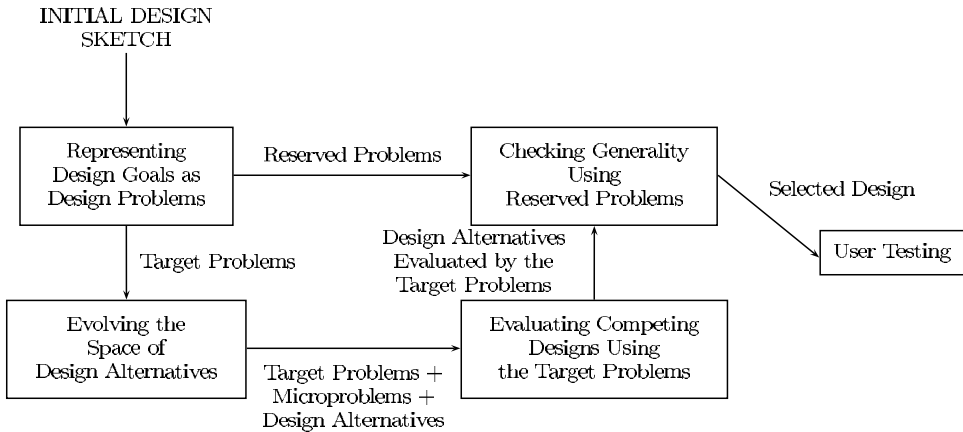


Fig. 10. The PCD way-of-working.

created starting from a typology of typical user concerns and instantiating from this typology specific scenario instances.

- Empirically (Empirical Scenarios Generation), where scenarios are based on observations of the users' behavior with similar artifacts.
- Claims Analysis, i.e., the analysis of the constructed scenarios using claims that link features in specific situations with desirable and undesirable psychological consequences. Claims Analysis consists of:
 - Claims Generation, i.e., the generation of claims purporting to explain the constructed scenarios.
 - Claims Justification, i.e., the justification of the generated claims.

The cycle is repeated as deemed appropriate until a satisfactory solution is derived. The set of claims used in the process constitutes the DR.

2.2.9 Problem-Centered Design. Problem-Centered Design (PCD) [Lewis et al. 1991] is, like PDR, more a design method than a formalism for representing rationale. Still, in the course of the proposed method DR is recorded, although in an implicit and idiosyncratic way. Specifically, PCD proposes placing *problems* at the center of the design process, where problems are concrete goals that are to be fulfilled (that is, the term “problem” is not used here with the sense of something being wrong, but of something demanding resolution). Figure 10 shows the way-of-working proposed. PCD proposes starting by representing design goals as target problems to be resolved. These are then partitioned in two categories:

- Target Problems that are used to guide the evolution of the space of design alternatives; to this end, the designers try to develop a description of the intended artifact that satisfies the target problems. During this phase additional problems of narrower scope, called microproblems, are derived. The competing design alternatives are then evaluated with respect to the original target problems.

—Reserved Problems. These problems are not used for the evolution of design alternatives, but for checking their generality. The idea is that designs may well resolve the target problems used for their creation, but may not be general enough to handle others; hence, an evaluation using separate problems is needed.

A specific design is selected and put to user testing. The result can be either the envisaged artifact or a new application of the approach starting with a more detailed design sketch. Throughout the process, the DR is captured implicitly by the relationships between the various design alternatives and the problems they solve poorly or well.

2.2.10 *Other Approaches.* There is a number of other DR approaches in the literature, which have been analyzed as part of the work reported in this paper, but have not been of direct influence to the design of the generic model. We present them summarily, along with the reasons for their exclusion in the synthesis of semantic and operational affinities that follows in Section 3.

Functional Representation (FR) [Chandrasekaran et al. 1993] is an approach for a formal representation of the artifacts functioning. FR can support diagnostic knowledge generation, simulation, design verification, and case-based design. The same focus on automation in DR is presented by Generative Design Rationale [Gruber and Russel 1992], which aims at an automated generation of the rationale behind artifacts; for this, it proposes reasoning ontologies and processes for use by computers. The idea of actively generating documentation is also pursued in Active Design Documents (ADD) [Garcia and Howard 1992]; an improved version of the system, ADD+ [Garcia and de Souza 1997], produces better rationale by using an approach for rhetorical organization. ADD was developed to assist a single designer; MultiADD [Vivacqua and Garcia 1997] extends ADD to deal with group design. Automation in DR can go all the way to automated software construction and maintenance: in Design Maintenance Systems [Baxter 1992] it is proposed that design goals, in the form of formal design specifications, be captured and used as input to a process of automatic decomposition and implementation in code. The artifact, then, is the outcome of its rationale. In a different vein, the artifact can be reflective in itself. That is, reflectiveness can be built on the system level. Hence, the idea of constructs that can reflectively inspect and interfere in their state has been explored [Dourish 1995]. None of these approaches, however, addresses directly the aspects of design as a communicative process between humans.

Some approaches aim at providing the basis for computer support in DR, instead of providing the DR themselves. OSC (from the French *outil de suivi de conception*) is a tool aiming at helping designers track and reuse decisions [Arango et al. 1991]. It is not intended, however, to be employed by the designers, but as a core for developing extensions for more specialized tools. DRCS is a structured language with explicit semantics for capturing design rationale in concurrent engineering teams [Klein 1993]

(extensions for capturing rationale about geometric features of designs have additionally been proposed [Klein 1997]). Its two main features, formality and expressiveness, make it especially suitable for computer implementations; but the focus is on improving computer support for the design process, rather than improving the design process directly. A similar goal is pursued in an approach for automatic conflict resolution of formally expressed requirements based on a metamodel similar to REMAP [Robinson and Volkov 1997]; again, the aspects of design as a human collaborative activity is outside the scope of the approach.

CADS [Favela et al. 1993] is a system that supports collaboration in engineering design. The approach is tightly coupled to a specific engineering design method. This helps the users of the method, but limits CADs to this specific constituency. DRIVE (Design Rationale for the Information Phase of Value Engineering) is a “path-finder” based on building dependency networks (paths) between objects parameters and constraints and their rationale [de la Garza and Alcantara 1997]; for us, DR is more about communication than about dependencies. DRIM (Design Recommendation-Intent Model) [Peña-Mora et al. 1995] is an ontology for representing DR. It is outside our concerns, since it aims at automating conflict mitigation and on actively providing part of the rationale for this purpose. In a setting closer to Software Engineering, DRIM is combined with design patterns to enhance their reusability by using the rationale behind them [Peña-Mora and Vadhavkar 1997].

The idea of combining design rationale with design patterns has also been pursued in the area of end-user tailoring of applications. Argumentation, combined with pictures, diagrams, stories and scenarios, provides the rationale behind application units; the user can tailor the units while consulting the related rationale [Mørch 1994; 1995]. Also, although the idea of using bare video for capturing DR is not very practical, it has, with appropriate modifications, led to an approach for documenting design history. In *Raison d’Etre* [Carroll et al. 1994], interviews with the designers are videotaped. The footage is broken into small clips with specific themes. The clips are transcribed; clips and transcriptions are entered into a multimedia environment allowing searching and browsing. The idea of using multiple media to capture history in the making is employed in the development of the Blacksburg Electronic Village [Carroll and Rosson 1996], with the assumption that the use of history can improve the design process. Our interests focus on using DR capture itself for improving the design process, and not in using its outcome, i.e., the rationale, for documentary and historical purposes.

2.3 General Characteristics of the Selected Design Rationale Approaches

Before we proceed on the derivation of a model for reflective reasoning in design, it is useful to take a fresh view and elicit some general characteristics and trends of the selected approaches. This will contribute to our research in two ways: first, we shall obtain a better understanding of our

working material; and second, this understanding will inform our own results. Three caveats are in order. These characteristics and trends are *general*; they are shared by the majority of the selected approaches, but not by all. Moreover, they pertain to the *selected* approaches only, and are not necessarily shared by others. And finally, they reflect the approaches developers' intentions; it is not always possible to gauge from the existing literature whether the approaches live up to the expectations; the characteristics are still useful, though, as a guide to what these approaches strive for.

Participation, Collectivity, and Plurality. All approaches are participative. *Participation* can be analyzed on two orthogonal notions. First, participation involves group working; hence DR approaches involve *collectivity*. Although they can be used by individuals working alone (an—unsuccessful—example can be found in the literature [Buckingham Shum and Hammond 1994]), DR approaches were developed to be used in groups. Second, participation involves *plurality*, i.e., the inclusion in the design team of participants who are not traditionally labeled as “designers.” Plurality, though, is a plea, rather than a feature. It rests primarily on extraneous factors such as corporate culture, group relations and group dynamics. Although DR approaches are collective, they can also be pluralistic, if the right conditions prevail.

Deliberation, Argumentation, and Simplicity versus Functionality. Most of the DR approaches aim at supporting *deliberation* and *argumentation*, especially those characterized by simplicity and minimality of concepts. Approaches with an extensive set of concepts are more demanding. It is argued, though, that this is balanced by their increased *functionality*, offered by the additional concepts introduced. These ease the way for automated operations on the data that is captured by using them. Moreover, their proponents claim that automated support can palliate their intricacy. However, we are interested in the approaches per se, and not with what they can become by using additional support, i.e., tools. With this proviso, most approaches opt for simplicity (REMAP and DRL are the exceptions).

Synchronic versus Diachronic Communication. Interpersonal communication can be *synchronic*, or *diachronic*. Synchronic communication is the live communication taking place in real time, e.g., face-to-face conversation and computer chat. Diachronic communication is relayed communication and is primarily conducted via recorded material, e.g., documentation and email. Approaches that opt for simplicity and ease of use stress argumentation and participation and therefore focus on supporting synchronic communication. The situation is reversed in approaches that opt for complexity and increased functionality. DRL and REMAP emphasize diachronic communication, whereas the other approaches emphasize synchronic communication.

Intrusive versus Nonintrusive. An approach is *intrusive* if it alters the design process. It is *nonintrusive* if it is *intended* to be used transparently, with little interference to it (we assume a complete absence of interference to be impossible). Nonintrusiveness is related to an approach's intentions, since nonintrusiveness could only be objectively asserted if one had proof that the process proposed by an approach is fundamental to the human nature. No approach has such proof. The criterion, although subjective, is still useful. If an approach is admittedly complex, or if it defines rigidly how to be applied, it is certainly intrusive; if not, it is presumably nonintrusive. Complex approaches requiring tool support (DRL, REMAP) are intrusive. PHI, PDR, and PCD are simple approaches but require a rigorous way-of-working. Hence, they become intrusive.

Prescriptive versus Nonprescriptive. An approach is *prescriptive* when it proposes a detailed set of steps to be followed during its deployment (a rigorous way-of-working); it is *nonprescriptive* if it offers only some guidelines, allowing considerable leeway in their application (a generic way-of-working). PHI, PDR, and PCD are prescriptive. The relation between intrusiveness and prescriptiveness is subtle. A nonintrusive approach is also nonprescriptive. A prescriptive approach is also intrusive; the contrary is not necessarily true. An approach can be intrusive without being prescriptive. This happens when, although no rigorous way-of-working is prescribed, the complexity of the approach renders it *ipso facto* intrusive (DRL, REMAP).

By-Product versus Coproduct. The DR produced by an approach can be a coproduct or a by-product of the design process. The DR is a *by-product* of design if its capture is a side-effect of the approach. It is a *coproduct* if it is seen as a separate product, as an artifact in itself, to the production of which special effort should be awarded. QOC regards DR as a coproduct, whereas the other approaches regard it as only a by-product. Note that the distinction does not imply a qualitative difference in the DR derived; the term by-product is employed here without negative connotations.

Process-Oriented versus Product-Oriented. An approach that aims at unveiling the justifications, arguments, and assumptions lying behind artifact features is *product-oriented*; it is *process-oriented* if it aims at unveiling the justifications, arguments, and assumptions behind process steps. In broad terms, the rationale recorded in product-oriented approaches answers questions of the type "why was this feature selected?", "why is this feature like this?", and so on, while the rationale recorded in process-oriented approaches answers questions of the type "why was this decision taken?", "why was this action taken?", and so forth. QOC, PDR, and PCD are product-oriented. The Potts and Bruns approach is more to the product-oriented end of the spectrum; the Potts approach is more process-oriented.

Informal versus Semiformal versus Formal. There are no completely informal nor completely formal DR approaches among the ones used here.

All are *semiformal*, i.e., they propose a formalism defining concepts, but they leave the concepts' contents informal. This formality is only of a structuring kind. More complex approaches, i.e., REMAP and DRL, are implemented in tools that increase their formality and extend it to operational aspects: the DR models produced can be manipulated and operated upon according to formal properties introduced by the implementations.

Summarizing this discussion, the selected DR approaches are designed in the light of the following characteristics:

- They are collective.
- They argue for plurality.
- They support deliberation and argumentation.
- They opt for simplicity and ease of use rather than complexity and increased functionality.
- They support interpersonal communication, mostly of a synchronic, rather than a diachronic kind.
- They aim at being nonintrusive.
- They are mostly nonprescriptive.
- They view DR as a by-product, rather than a coproduct, of the design process.
- Equal importance is given to product and to process rationale.
- They are semiformal, employing formality as a structuring mechanism.

These characteristics are taken up again in Section 6, where they provide a concise characterization of our approach as well.

3. TOWARD A MODEL FOR REFLECTIVE REASONING IN DESIGN PROCESSES

The development of a model for reflective reasoning in design is now conducted by abstracting common points among the different approaches. We show, in spite of surface differences, that common principles and similarities can be found.

Common principles and similarities may relate to the analogies and correspondences between the concepts introduced by the various approaches, i.e., their semantic affinities. These focus on the static aspects of DR approaches and on the rationale models resulting by their use. They are *static affinities*, i.e., *affinities of the invariant semantic concepts of the various approaches*. Static affinities have formed the locus of previous comparative research [Lee and Lai 1991] and are the subject of Section 3.1.

Common principles and similarities may also relate to the operational semantics of the various approaches; in effect, their proposed ways-of-working. Such affinities are *operational*, or *dynamic affinities*, and must

not be neglected. They are the subject of Section 3.2. A synthesis of the results follows in Section 4.

3.1 Analysis of the Static Affinities of Design Rationale Approaches

All approaches make some assumptions on the concepts to be employed and their use, but not all of them explicitly. We denote those approaches that propose an explicit metamodel of concepts and the way these are combined as *metamodel-based* approaches, and those that do not provide such explicit metamodels as *non-metamodel-based* approaches. More succinctly, a metamodel-based approach provides a metamodel specifying entities and relationships to be used for representing the rationale (in Conceptual Modeling parlance, they are Entity-Relationship metamodels). IBIS and its descendants (gIBIS, itIBIS), REMAP, the Potts and Bruns and the Potts approaches, PHI, QOC, and DRL are metamodel based; PDR and PCD are non-metamodel based. An analysis of the static affinities of DR approaches must focus on the metamodel-based approaches, since these provide such static constructs in an unequivocal and accessible way.

Insofar as entities are concerned, the differences among DR approaches are not extensive; the situation is altogether different in what regards the proposed relationships among them. Even in approaches akin to each other (e.g., the Potts and Bruns and the Potts approaches), the differences in the proposed relationships are important, both in their number and in their semantics. A possible reason could be that it is the way the fundamental building blocks of reasoning are combined together that provides the flexibility and the ingenuity concomitant with them; in other words, that what is most important is the creation of a web formed by the instances of the entities established by each approach, and not the instances considered in isolation: for example, the full meaning and impact of an instance of an Argument entity is not given by its contents but by the relationships it enters with the other instances of the entities in the specific context.

We return to the problem of relationships in Section 4.2. It is, however, possible to arrive at a synthesis of the entities provided by these approaches. The following common set can be derived:

—Goals, which *designate any objective to be reached, demand to be satisfied, problem to be resolved, issue to be discussed, in general anything that may arise in the design process and requires effort to be effected, anything that one would like to achieve*. This entity comprises a number of similar concepts found in DR approaches and covers the Goal object of DRL and the Requirement object of REMAP. The Goal entity's semantics comprise the semantics of Decision Problem (DRL), Issue (IBIS, PHI, Potts and Bruns, Potts), Question (QOC), and Criterion (QOC). A Decision Problem, an Issue, and a Question are concepts that require a resolution, hence, we have a Goal of resolving them. A Criterion is something used to evaluate alternatives. It represents desirable features, or requirements (goals) of the envisaged artifact. Goals also encompass demands for clarification in the design process (Question objects in

- DRL). Finally, a Constraint (REMAP), although it represents the result of resolving an Issue, is still something that exerts demands on the designers for the features of the evolving artifact; it is, therefore, a Goal.
- Hypotheses, which designate *any suggestion, proposal or idea about the resolution of a problem in the design process*. Hypotheses are ways of tackling (satisficing) Goals. Hypotheses here comprise Options (QOC), Positions (IBIS, Potts), and Answers (PHI); they also cover the semantics of the Alternative object (Potts and Bruns, DRL).
 - Design Actions, which represent *any action taken in the course of a design project*. The entity comprises the Decision object (REMAP) and the Step concept (Potts).
 - Justifications, which represent *any statement in the design process that can be conceived as a claim for something being the case*. This entity comprises the Claim object (DRL), the Argument object (IBIS, Potts, PHI, QOC), and the Justification object (Potts and Bruns). Assumptions in REMAP are metaarguments (arguments for arguments) and therefore justifications.
 - Artifacts, which designate *any object that is used, produced, or worked on during the design process*. This entity comprises both the Artifact object (Potts and Bruns, Potts, and DRL extensions [Lee 1991]) and the Design Object (REMAP).

3.2 Analysis of the Dynamic Affinities of Design Rationale Approaches

A problem with looking for the dynamic affinities of the various DR approaches is that explicit models for their application are usually not given. To compensate, we follow the examples of their use found in the literature. We show that, the surface appearances notwithstanding, a common way-of-working model emerges. Specifically, all ways-of-working are variations of a common theme that forms a skeleton for reflective reasoning in design.

Such a skeleton is not a model, or a formalism, of reasoning. What we are after is a model or formalism of an *aid for reasoning*. We have no aspirations of mirroring the actual cognitive processes of reasoning per se. This may actually unfold in any way that may be analyzed by the relevant research. But this does not compromise the usability of the outcome of our work. For example, a word processor has no pretences of mirroring the cognitive processes of composition; but it is, nevertheless, a very useful tool.

Starting with IBIS and REMAP, the process starts by establishing problems requiring resolution. It proceeds with seeking alternatives for the problems. Arguments for the alternatives are provided, and, based on the argumentation carried out, an alternative is chosen. This is similar to what the Potts and Bruns and the Potts approaches propose. The process starts by establishing problems on account of existing artifacts. It proceeds by establishing alternatives addressing the problems. Then, arguments for the alternatives are provided and an alternative is chosen.

PHI seems to differ, since a structured top-down breadth-first decomposition process is advocated. But the process starts again by establishing problems; problem decomposition, the fundamental procedural construct in PHI is problem setting. It proceeds by proposing answers to the problems. Then, arguments for the proposed answers are provided. Answers are selected based on the argumentation.

Similarly, QOC starts by establishing questions related to an artifact. It then proceeds by proposing options addressing the questions. These are selected based on argumentation that takes into account specific requirements. In DRL, the process starts by establishing goals. It continues by giving alternatives to achieve these goals. Alternatives are evaluated by arguments, and suitable candidates are selected.

Proceeding to non-metamodel-based approaches, PDR presents more surface differences. The process starts by generating scenarios. This is followed by the generation of claims explaining the scenarios. The claims are justified, and the existing scenarios are modified, or new ones generated. But note that scenarios are created and modified in response to perceived user problems and they propose alternatives for the envisaged artifacts. Claims justification is an argumentation procedure, where claims represent arguments on analyzed scenarios. The procedure then becomes a variant of the previous ones. PCD is more straightforward. First, problems are established, then alternatives, which are evaluated for a selection to take place.

This discussion suggests the following common structure consisting of four *reasoning primitives*:

- Problem Setting (establishing issues for resolution).
- Problem Analysis (analysis of the problem domain, in order to find alternatives for resolving the issues established).
- Solutions Evaluation (evaluation of the proposed alternatives).
- Problem Resolution (resolution of the issues by choosing among the alternatives).

4. THE REASONING LOOP MODEL

4.1 Description

A reasoning loop model for reflective design processes is a composition of a set of static primitives capturing the information content of the design process (Section 3.1), and a set of operational primitives capturing its dynamic component (Section 3.2). The former is concerned with what is done; the latter with how this is done. A synthesis of the two shows the interaction between process and substance. The following correspondences are identified:

- Problem Setting generates *Goals*. Designers set the objectives to be reached, the demands to be satisfied, the problems to be resolved, the

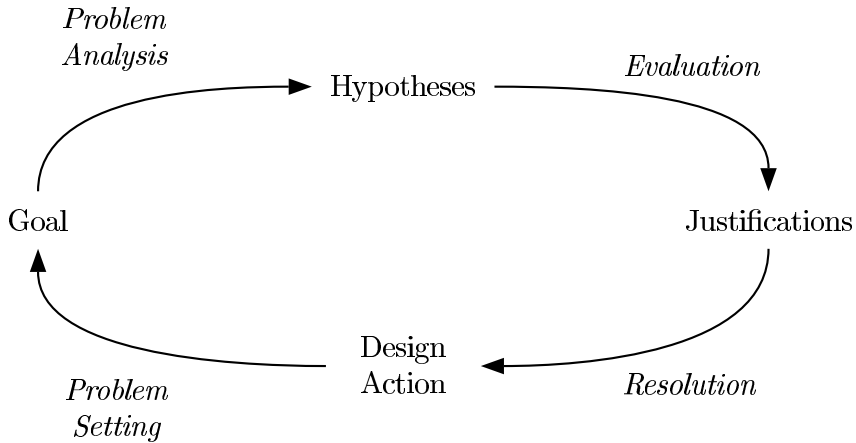


Fig. 11. The reasoning loop.

issues to be discussed. These include anything that may arise in the design process and requires effort to be effected, anything they would like to achieve.

- Problem Analysis generates *Hypotheses*. Designers analyze the problem domain to arrive at suggestions, proposals, or ideas about the resolution of identified problems.
- Solutions Evaluation generates *Justifications*. Designers produce claims on the status of the generated alternatives.
- Problem Resolution generates *Design Actions*; these alter *Artifacts* while generating and resolving *Goals*. Designers take actions affecting the objects used, produced or worked on during the design process; existing problems may be resolved and new problems may be generated.

With the help of the above, we arrive at Figure 11 showing a *reasoning loop* which contains the relationships among reasoning and conceptual primitives.

The process commences with the statement of the problem (it need not be a precise one, since its ramifications can form a large part of the design). This is followed by *abduction*, i.e., the putting forward of hypotheses for its resolution. Then *retroduction*, i.e., the testing of the competing hypotheses, ensues in order to select some (or one) of the hypotheses, thus altering the situation and beginning a new cycle.

The activity is goal-driven, in that it stems from and aims at resolving goals. It is important to emphasize that a *Goal* is anything that arises during the design effort and the designers try to resolve or to achieve. They subsume enterprise or project goals, but are not limited to them. In short, *Goals* are the designers' intentions with respect to the project at each point during the project.

The creation of new cycles results in the construction of a network of reasoning loops, each one of them spawned to resolve a problem arising in

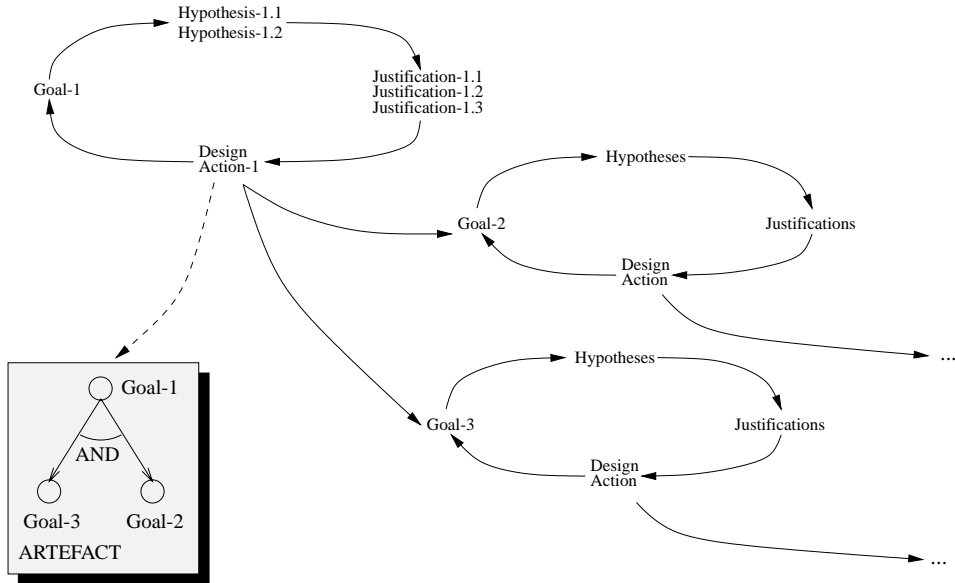


Fig. 12. An example design process.

the resolution of another problem. It has been argued that work is a closed loop process, resulting in a network of commitments [Medina-Mora et al. 1992; Denning 1992]; analogously, reasoning can proceed in a closed loop process, starting with the definition of a problem and closing with its resolution. The notion of a *closed loop* has two connotations that clarify its meaning here. First, a closed loop is a sequence of steps that must be closed, in the sense of being completed; if it is not, the problem has not been solved. Second, like a programming loop, it is a sequence of steps that is used repeatedly throughout the design process and whose contents take up different values in each application.

To render the discussion more concrete, a simple example of a design process is shown in Figure 12. In this example, two Hypotheses are established in order to tackle a certain Goal. Some argumentation ensues, and one of the Hypotheses is chosen. This entails two new Goals, and the process will continue until all goal exigencies have been met. The Design Action undertaken alters some artifact. In the example the problem is to arrive at the requirements, in the form of an AND/OR graph, for a given system under development. The framed AND/OR graph is the Artifact being considered, and it is connected to the reasoning loop network with a dashed line (this is only a convention: any other means for connecting Artifacts to the reasoning loop network could be used).⁷

⁷Just as the Goal, Hypothesis, and Design Action concepts are instantiated by real-world goals, hypotheses, and design actions in the use of the model, the Artifact concept is instantiated by real-world artifacts. These may not exist on the same medium side by side with the reasoning loops, hence they do not actually form part of reasoning loops; but Design Actions nevertheless refer to them.

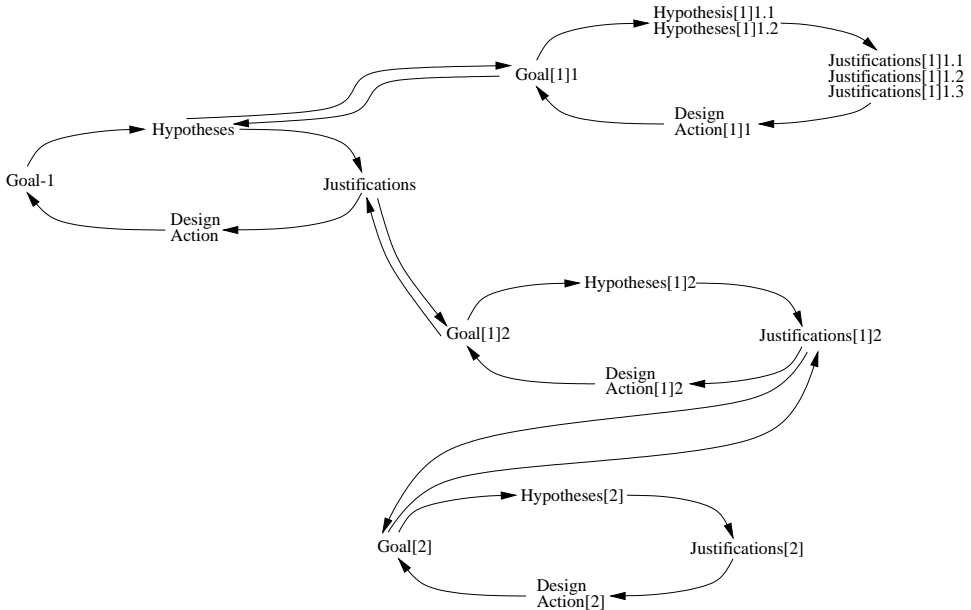


Fig. 13. An example process including metadiscussions.

We see that the Design Action results in a new goal graph consisting of two subgoals whose joint resolution will result in the fulfillment of the first goal.

To reiterate, we argue that *an entire design process can be seen as a repeated application of the reasoning loop wherever problems are raised*. The word “wherever” in the above statement is of special importance: Goals arise in the definition of any kind of entity, be it a Goal, Hypothesis, a Justification, or a Design Action: this entails the creation of secondary loops aiming at the closing of a primary one *in contexts other than action enactment*. This enables the design process to include *metadiscussions*, i.e., discussions on the design process per se, discussions referring to the ongoing discussions. Figure 13 clarifies this, showing a possible abstract model of such a design process (concrete examples are presented in Section 5); in this figure, and in what follows, we add an index corresponding to the (meta) level removed from the primary loop (these indices are used here for presentation purposes but not in actual applications). The process started with the statement of Goal-1. During the analysis of Goal-1, Goal[1]1, shown in the top-right loop in the figure, arose, regarding the identification of a Hypothesis (e.g., “is...really a Hypothesis?”). In this loop, two Hypotheses were identified for its resolution (Hypothesis[1]1.1, “yes it is”, and Hypothesis[1]1.2, “no, it is not”); after an evaluation based on three Justifications, one of them was chosen and carried out (e.g., “accept it as a Hypothesis”), driving the process back to the first loop. Once the Hypothesis had been settled down Goal[1]2 arose in the course of Justification definition (“is this Jus-

tification relevant?”). A solution for its resolution was proposed (“yes, it is”), but during its evaluation Goal[2] arose, leading to another loop where the validity of a Justification was questioned.

We call the loop from which a metadiscussion arises the *primary loop* and the loops constituting metadiscussions *secondary loops*. Secondary loops can be numerous, and metadiscussions can form the bulk of a design process. Indeed, it has been shown that among a classification of design activities, clarifications (in which metadiscussions are subsumed) consume more time than any other activity, and about a third of all time [Olson et al. 1996]. Even in extended metadiscussions, though, the evolution of the artifact can be readily traced by keeping only the primary loops and exempting all the secondary ones. In the example of Figure 12 this can be done by exempting indexed loops.

Interestingly, since the artifact under construction in this example is a goal graph, there is a close synergy between the goals comprising the goal graph and the goals pursued by the designers. By internalizing the project’s goals, the designers make them their own intentions. Hence, the primary loops Goals comprise the goal graph under construction. In general, though, there is no such relation between the designers’ goals and the artifact. Of course, the designers aim at producing something, which is the Artifact on which primary loop Design Actions and Goals impinge, but these Goals do not necessarily form part of it. An Artifact may be a set of specifications, a plan for future action, a method specific document, and so on. If it is, for instance, a process model, Goals refer to it, but do not form a part of it.

In summary, the four reasoning primitives identified in Section 3.2 are found to entwine with the four conceptual primitives established in Section 3.1 to produce a reasoning loop. Reasoning processes can be mapped to repeated applications of this loop, which in this way functions as the atom of the design activity.

4.2 The Problem with Relationships⁸

In contrast to *fixed-link* approaches that define *a priori* fixed semantics for the relationships linking the entities in their ontologies, we propose a *free-link* approach that leaves these semantics free, to be determined during actual use.

We argue that no DR approach can sufficiently and comprehensively capture all possible relations among concepts. They are too many to be exhaustively enumerated, and sometimes they have implicit meanings that cannot be specified at all, until later on, after subsequent discussion. A fixed set of links limits both the expressive and the functional capabilities of the model.

⁸We thank one of the anonymous reviewers for pointing out that the criticism in this section is related to other criticisms of the fixed metamodels that plagued many failed CASE tools, a parallel that is not pursued here.

Regarding the expressive weaknesses of any such attempt, the sheer number of the proposed relationships in the various DR approaches and the differences in their semantics from approach to approach indicate that it is difficult to arrive at a widely accepted set of predefined links. Each approach commits to a certain set, but there is no reason to believe that one of these sets is innately better than the others. Even the union of all these different sets would not be enough. For instance: a *Justification* may support a *Justification*, or it may object to it; it may depend on a *Justification*, or it may be suggested by it; it may be questioned by another *Justification*, and so on.

Moreover, a link among two concepts can often be established, but its nature cannot be fully elucidated. Demands to do so require premature analysis of the relationship; sometimes links have implicit meanings that are difficult to make explicit unless more analysis is undertaken. Commitment to an explicit meaning can be enforced, but this is unnatural and contrary to the flow of the design process. For example, when providing *Justifications* it can be more natural to go through bursts of generating criteria that are thought to be, in one way or another, relevant to the alternatives. Their exact relationship to them, like how positively or negatively they assess them, is decided upon after they have been generated. The same effect can be observed with *Hypotheses* related to a given *Goal*.

As for the functional weaknesses of fixed-link approaches, the current line of research in DR accepts that the functionality provided is proportional to the wealth of concepts, and especially relationships among them. Generally, increased functionality presupposes some kind of enhanced formality of the concepts and relationships provided. By agreeing on specific semantics of the approach's links, appropriate support tools can be developed. It can therefore be argued that a free-links approach, like the one advocated here, can hamper the approach's potential for such kinds of computerized support. On the contrary, though, a rigid semantic definition of concepts and relationships limits functionality, along with expressiveness, to the preconceived notions of the approach's developers.

A free-links approach can accommodate both expressiveness and functionality. We do not advocate *shedding* the information carried by relationships provided by traditional models (although even this has been proposed by PCD in the context of small-scale projects involving a limited number of participants). Provided this information can be explicitly stated, we propose not casting it into prefabricated molds. This leaves two possibilities: first, links can be named at will, in the course of the design process, by choosing names on the fly by the designers themselves depending on the context at hand. Or second, the information carried by the links can be transferred to the concepts: for example, a *Justification-supports-Hypothesis* structure can be represented as a "supporting *Justification*"-*Hypothesis* structure. This is shown in the *Design Actions* in our examples. There is no closed set of possible *Design Actions*, and the contents of each one of

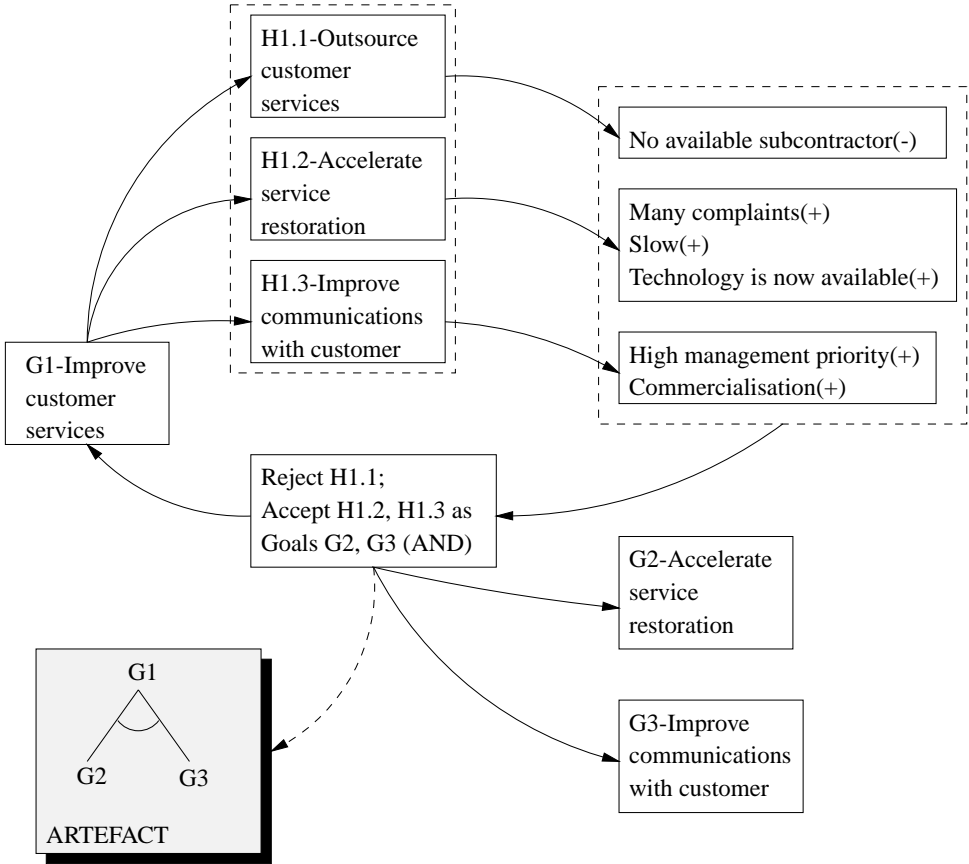


Fig. 14. The ESC case study: 1.

them determine the exact nature of the relationship linking it to the Goals they engender.

5. EXAMPLES IN THE USE OF THE REASONING LOOP MODEL

We now demonstrate the use of the reasoning loop model by means of real-world examples. For reasons of space, no complete example can be given, as the resulting loop networks are very large for anything but toy problems. Hence, we present here only parts of real-world problems and design processes. They are self-contained, and can be understood independently of the excluded discussion.

The first example concerns a major European Electricity Supply Company, to which we refer as ESC. Because of pressures for commercialization and liberalization of the electricity supply industry, ESC has begun a series of projects for analyzing and understanding the present situation, devising scenarios for the future, and proposing steps that can be taken in advance to facilitate the transition to the forthcoming business environment.

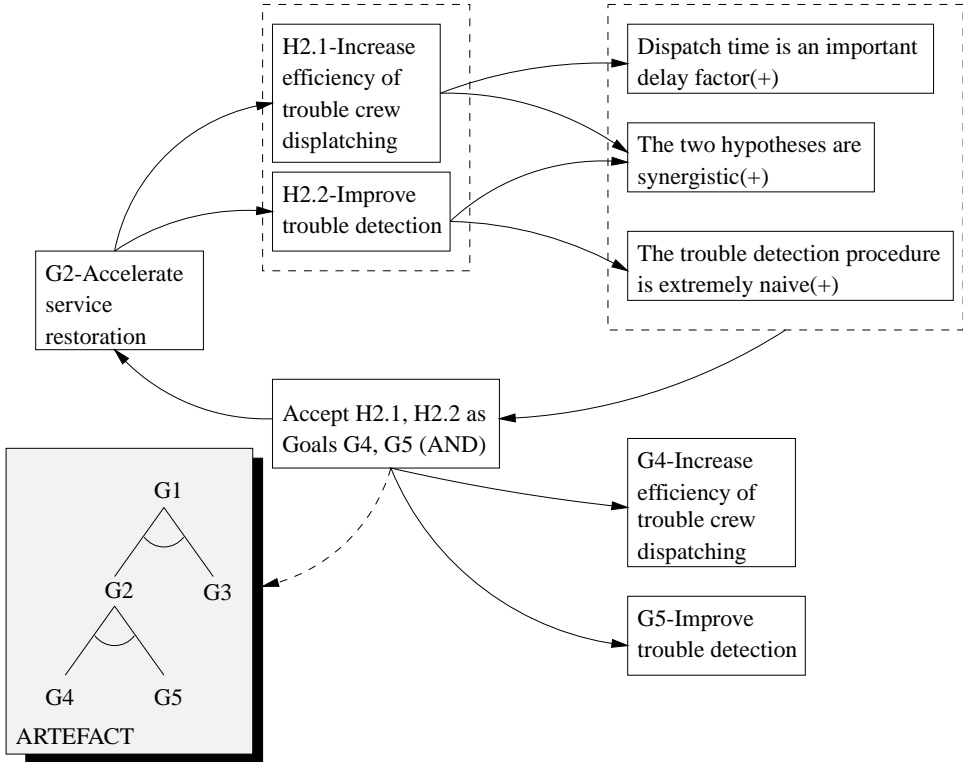


Fig. 15. The ESC case study: 2.

One of the key concerns of ESC is improving their customer services (see Figure 14; labels standing for full discussion items are employed to facilitate presentation). This, therefore, represents a goal that has to be resolved. In the discussion that followed, three possible hypotheses were proposed: customer services could be outsourced, service restoration could be accelerated (since it was felt that service restoration is a vital aspect of customer services and room for improvement exists), and customer communication could be improved (since it was argued that customer services are not that bad after all, and that the problem was rather in communicating with the customers). A round of argumentation ensued. The first hypothesis was rejected, due to the unavailability of subcontractors (an objecting/supporting justification is marked by a minus/plus sign respectively, see the discussion on relationships in Section 4.2). The second hypothesis had some points in its favor: there were many complaints for faults repair, restoration was often slow, and new technology made improvements feasible. The third hypothesis was supported by the fact that improved customer communication was championed by the management and by the demands of the impending commercialization. Hence the decision was taken to try to resolve the goal of improving customer services by both accelerating service restoration and improving customer communication.

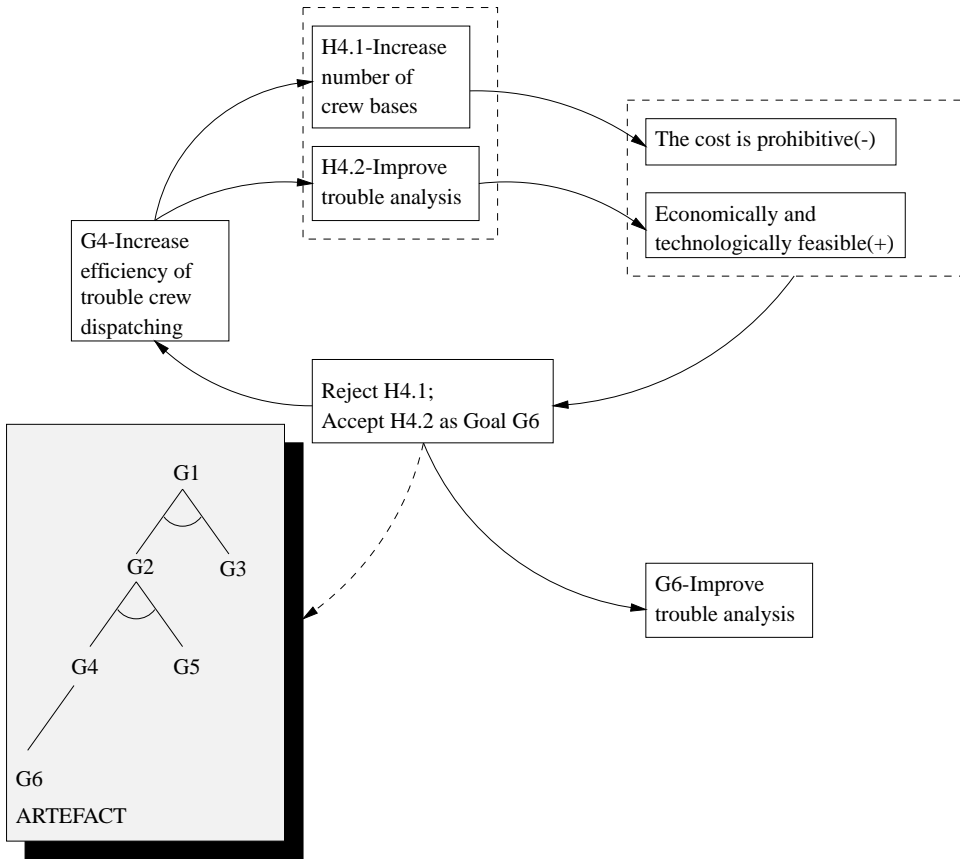


Fig. 16. The ESC case study: 3.

The discussion continued for each of the two new goals. Figure 15 shows the progress on the goal of accelerating service restoration. Here two hypotheses on possible ways to fulfil the goal were formulated; since all justifications were in favor of both, they were both accepted as joint ways to attain the goal. (Discussion on the other sibling goal of improving customer communications was left for later consideration.)

The discussion went on for the goal of increasing the efficiency of trouble crew dispatching (see Figure 16). The goal of improving trouble detection was handled as shown in Figure 17. Here a minute metadiscussion arose, with G[1] questioning the nature of a Justification that challenges H5.

It must be noted that, although for reasons of space and presentation each reasoning loop has to be presented separately and in an imposed sequence, all are parts of the same loop network; moreover, the order of the loops presented in Figures 16 and 17 is inconsequential. The discussion of the goal to increase the efficiency of trouble crew dispatching could be conducted before, after, or in parallel with the goal of improving trouble detection.

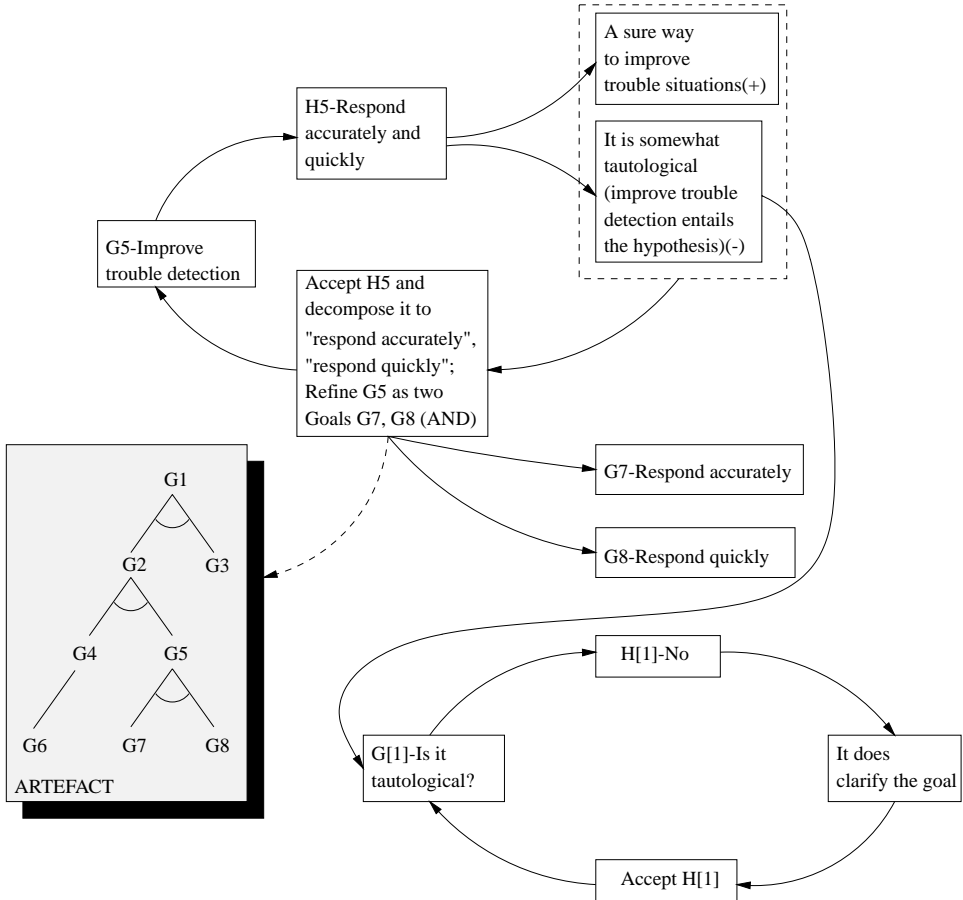


Fig. 17. The ESC case study: 4.

Metadiscussions played only a minor role in this part of the ESC project because little controversy arose. Although this suits our presentation purposes, since the diagrams remain clear and uncluttered, the situation can be completely different in controversial issues. The importance of metadiscussions in capturing them is highlighted in the next example, concerning the evolution of the C++ programming language. This is an instance of *retrospective* DR, where the objective is to uncover the reasons and rationale behind the current status of a given artifact. In effect, we are reverse-engineering the artifact, tracing its features to their requirements.

By consulting material on the development of C++ [Waldo 1993; Stroustrup 1994], we constructed a goal graph (Figure 18) showing the evolution from the initial motivation to develop C++ to the requirement of providing multiple inheritance in the language. At that point, the desirability of including multiple inheritance in C++ was questioned and incited a heated discussion, some of the main points of which are summarized in Figure 19. Although only a few pieces of the enormous amount of data that can be

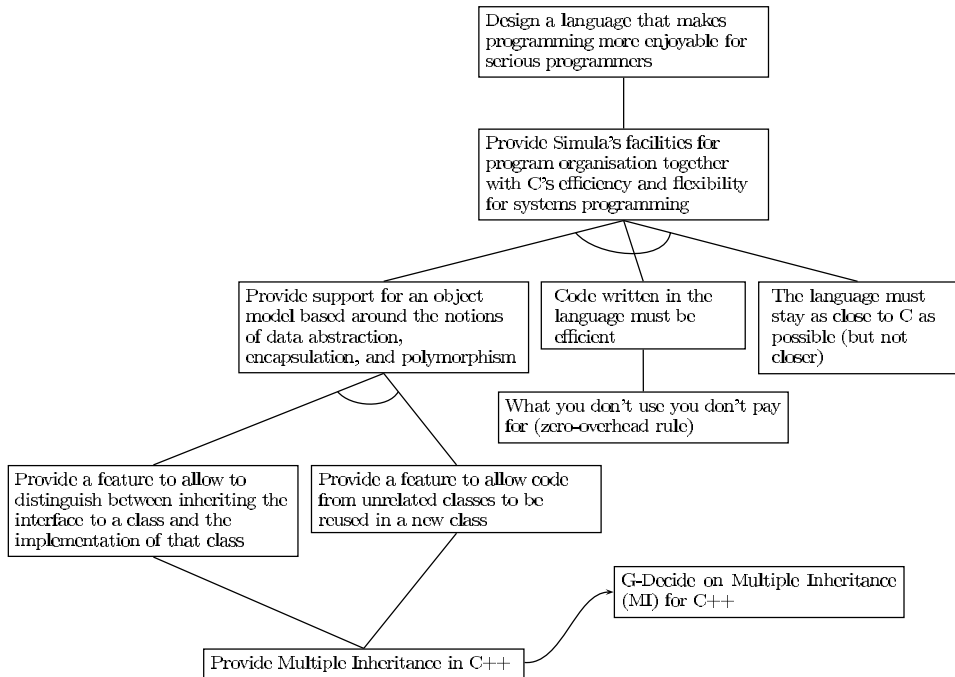


Fig. 18. The C++ case study: 1.

amassed by trawling on the lengthy controversies on the subject are presented here, it is easy to observe that the effort to decide on multiple inheritance in C++ is in the form of metadiscussions. To see that, one needs to fill in the omitted reasoning loops emanating from the “Many equate OOP with Smalltalk” or “Several of the techniques...” justifications: it is clear that the number of levels in such situations reaches great depths. We have noted, in Section 4.1, that clarifications, which are represented by metadiscussions, make up a major part of the design process. In highly charged and controversial problems (like this one), questions about the relevance and the validity of goals, alternatives, arguments, in general of any assumptions and claims, arise frequently, and their resolution is essential for the resolution of the original problem. The metadiscussion mechanism is useful not only in quantitative terms (the amount of the design process it can represent), but in qualitative terms (the importance of the part of the design process it can represent) as well.

The Artifacts of our examples are goal graphs. This, as has been remarked in Section 4.1, produces a nice synergy between Goals and Artifacts. It is not a rule, however, as there may be any kind of Artifact. The synergy is nice but not compelling, and it was not the reason for using a goal modeling approach. That was an independent decision based on our experience in using such approaches [Loucopoulos and Kavakli 1995; Kavakli and Loucopoulos 1998; Kardasis and Loucopoulos 1998].

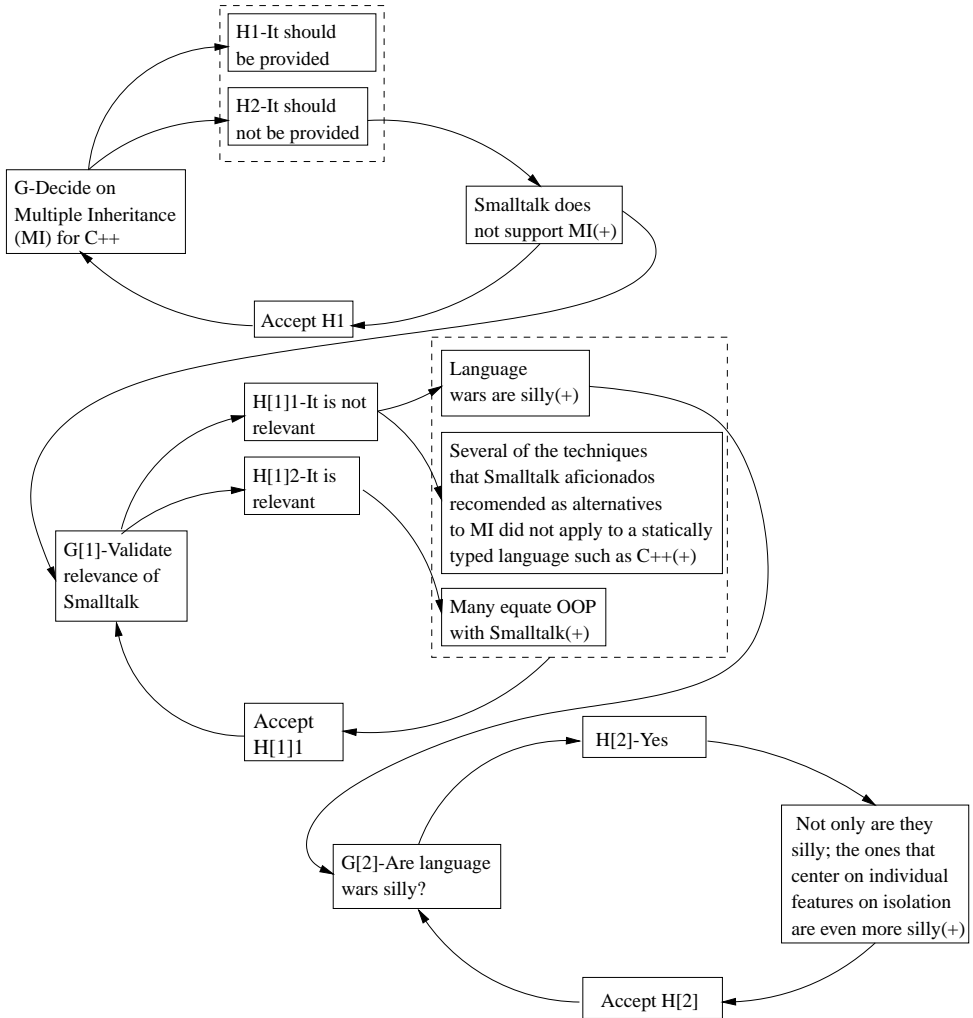


Fig. 19. The C++ case study: 2.

The generic character of the approach is also reflected on the Design Actions that can occur. These resolve Goals. The resolution is based on the Hypotheses stated and their evaluation through Justifications. As mentioned in Section 4.2, there is no closed set of possible Design Actions. The simplest Design Action is the acceptance or rejection of a Hypothesis. More subtle Design Actions involve the acceptance and combination of more than one Hypotheses. Even subtler Design Actions involve the combination, not of whole Hypotheses, but of parts of them; and so on, with different forms of Design Actions. In fact, the most successful resolutions are those that manage to read the proposed alternatives in a new light, combine opposing views, and derive a solution that moves beyond differences. Problem resolution, as expressed by Design Actions, is a creative, open-ended process.

6. DISCUSSION AND CONCLUSIONS

6.1 Usability

The use of DR approaches is hampered by two main problems: the size of the resulting rationale records, and the overhead placed on the users of the approaches when producing them. The graphical form of the reasoning loop, shown in Figure 11, the absence of fixed links, and the limits of our ambitions temper these problems.

The graphical form of the reasoning loop is derived from similar graphical structures employed in the context of workflow-business process modeling with good results [Medina-Mora et al. 1992]. The elliptical loops connected with loose curves seem elegant, and they have been shown to work in the composition and management of large models and networks. The differences between process modeling and the construction of reasoning loop networks put a check on cross-fertilization (in process modeling the network is the stable end-product, while the reasoning loops are fluid and evolving by-products); yet, the reasoning loop by itself does possess some good characteristics.

The spatial form of the loop lets us manage without introducing special symbols for the representation of each concept. The position of a concept in a loop shows its type (left side: Goal, topside: Hypothesis, right side: Justification, underside: Design Action), which is therefore recognizable by simple inspection. The resolution of a problem is indicated by loop closure, i.e., the arc connecting Design Actions with Goals. The reasoning loop keeps relevant parts of the discussion together; the discussion unit is the loop itself, and not its elements. As a result, loop networks deviate from the highly interconnected, cluttered, and undifferentiated networks of nodes typical of DR records.

The position of loops distinguishes metadiscussions (they are tangential), which become immediately apparent. Lengthy argumentations take the form of metadiscussions. The ability to make them out functions as a feedback mechanism controlling the size of loop networks.

The absence of fixed links contributes to the ease of use. Fixed links result to the introduction of a closed set of possible moves that can be made during a design process. This leads to problems of validity of moves, which plague the use of DR approaches.

Finally, an important factor mitigating the problem of the size of loop networks has to do with the limits of our intentions. The approach focuses on synchronic communication. Past loop networks are not grown and accumulated in archives to form the substrate of present decisions. They grow only until the objectives that initiated the design process are resolved. In this context, the simplicity and the terseness of the model make possible its application with no more resources than a whiteboard or pen and paper. At the same time it limits its potential and usefulness, since taking into account past decisions and their rationale can lead to better decision making.

6.2 Automation

Automation can be potentially useful in extending the capabilities of the approach both in synchronic and in diachronic communication. Some desirable characteristics of computer support can be posited here in brief.

The merits of diachronic use are those of experience: past rationale can function as a sort of collective memory that can be brought to bear on present problems. Moreover, this memory is evolving, incorporating new rationale and new decisions produced by the use of the approach. If artifacts are also stored, their features can be traced back to the relevant decisions. Advice can be offered by mapping the present situation to the knowledge stored; this can be passive, with the designers asking the system for it, or active, with the system offering it by following the designers' steps and trying to establish points of intervention. Concomitant with these possibilities come challenges: effective use of loop networks across time requires effective storage and retrieval; the provision of pertinent and effective advice is an intriguing and topical problem (e.g., as attested by the current interest in Intelligent Agents).

Synchronic communication can also benefit. With universal connectivity proceeding apace, appropriate tools can support collaborative design projects not only in a local, or intranet level, but in much wider virtual communities. Automation can support conflict detection: contradictory and mutual cancelling decisions and argumentation are likely to occur. And automation can also support the detection of relapses: the participants may repeat explorations that they have performed before, even during the same design session.⁹ These problems are endemic to all collaborative design processes. The approach can, by externalizing the discussion, make them more conspicuous, but it cannot actively assist in their detection, and this provides a window of opportunity for automation.

The link-based structure of the loop networks suggests an affinity with hypertext, which has been the backdrop of most implementations of DR approaches. However, hypertext, if implemented simply as a collection of linked components, will compromise the benefits offered by the graphic notation of the reasoning loop. The spatial properties of our approach, i.e., the clustering of relevant discussion elements in a closed loop and the tangential character of metadiscussions, should be preserved and capitalized upon. Lessons can be drawn from spatial hypertext tools like Aquanet [Marshall et al. 1991] and VIKI [Marshall et al. 1994].

This brings us to the semiformal nature of the approach, which complicates computerization. Formality, though, presents problems of its own by increasing the cognitive burden placed on the user [Shipman and Marshall 1994a; Shum 1991]. If simplicity is not to be compromised, any formalization should remain hidden. An interesting idea is incremental formalization [Lee 1997], where the information is captured using a semiformal

⁹Of course, depending on how far in the past such goes, it can be a matter of diachronic, rather than synchronic use.

framework and is subsequently submitted to a gradual formalization process (e.g., Shipman and McCall [1994b]).

Our initial aim was to find a way to improve design processes through increased reflectiveness. This does not necessarily imply tool support, and this was not among our primary objectives: to our view, tool support should add to our approach, and not be an essential part of it. It is true that such support could improve the approach's potential in many aspects. The independence of the approach from automation support, however, makes it applicable in a wide range of user environments. Moreover, by separating the two we obtain a methodological advantage: tools can be developed to address specific needs that are discovered empirically, during the approach's application.

Indeed, talk is cheap. Whether extensive computer support is better than simple recording of loop networks with electronic means; whether the benefits to be accrued by computer support outweigh the burden of recording, organizing, and handling the information; and whether they justify a departure from a bare pen-and-paper approach (plainness is often more alluring), cannot be settled *a priori*, but on empirical data.

6.3 Concluding Remarks

The approach proposed here is intended to be employed in organizational environments during collaborative design sessions. By designers we do not designate only the information technology professionals, but all those who are affected by the evolution of the artifact under development. This raises the most serious problem regarding the use of the approach, namely, the politics involved. The approach is useless without a commitment to openness and participation.

The politics problem is exogenous. Automation can offer some help, e.g., by offering anonymity with appropriate interfaces. But the issue remains primarily one of the prevailing culture in the given context. For example, the participants are more wary to express their opinions when they think, or know, that these may be used to assign responsibilities and blame if the decisions they supported prove, in hindsight, to be wrong. Despite pronouncements for employee empowerment and continuous process improvement, corporate reality can be very different.

The importance of collaboration, negotiation, and the management of conflicts is prevalent in most Software Engineering settings. It is most acute in the initial phases of systems development, i.e., Requirements Engineering, when the limits of the problem to be resolved must be set, and a set of requirements to be met must be composed. Other researchers in the field have worked on similar problems, but our approach differs in several ways: the NATURE [Grosz et al. 1997; Pohl 1996] metamodels are rather suitable for use with appropriate tools, the Inquiry Cycle [Potts et al. 1994] focuses more on artifacts rather than the process, and WinWin [Boehm et al. 1998] is oriented toward automated support for conflict resolution.

The approach can be viewed in the light of Section 2.3, i.e., in terms of the general characteristics of DR approaches. The reasoning loop can be characterized as follows:

- The reasoning loop was developed to be used in collective, collaborative design processes. There is no innate reason to prevent it from being used by individuals in a private manner, but this was not among our goals during its development.
- We argue that it be used as pluralistic as well.
- It supports deliberation and argumentation.
- It is simple and aims primarily for human, not computer usability.
- Hence, formality was kept at a low level.
- It focuses on synchronic communication. Effective diachronic communication requires automatic support. This, however, must be done at a lower level, so as not to compromise the users' conception of the model. The complexity entailed by increased functionality must remain hidden.
- The approach is nonprescriptive. Being simple and nonprescriptive it aims at being nonintrusive.
- DR is produced by design itself, and hence it is a by-product of the design activity.
- It captures both process and product rationale. The process rationale (the rationale behind the sequence of events during design) is captured, since the creation of loop networks proceeds chronologically, mirroring the design process itself.¹⁰ The product rationale (the rationale behind specific artifact features) is also captured, since the Design Actions connect the artifacts to the reasoning loop.¹¹

These characteristics determine the application context of our approach. The reasoning loop is to be applied to help resolve wicked problems, i.e., real-world problems with no clear statement, no clear final state, and no clear set of means to transform the initial state to the final state [Rittel and Webber 1984]. Argumentative approaches in straightforward situations can be hampering [Buckingham Shum and Hammond 1994].

The approach is generic in two senses. First, wicked problems represent a large class of real-world problems. Secondly, it is generic by being the result of an abstraction process. Argumentative approaches can be customized to the method they help to apply [Potts 1996]. We believe that this

¹⁰This process subsumes, but is not limited to, any process steps advocated by any specific process model that may be followed. Process capture is not a fortuitous side-effect. Process capture is the means by which the design process is externalized and can be reflected upon (thus enabling a metadesign process, cf. Section 1).

¹¹This is not fortuitous either: Design Actions were defined so as to ground the design process to its artifacts.

constrains the reasoning processes during design by allowing only a given set of goals, alternatives, and argument classes to occur. This might be desirable in relatively well-known and ordered contexts, but these do not fall into our area of applications.

We intend to use our approach in Change Management initiatives involving the use of information technology in corporate settings. These projects involve a large number of stakeholders in heterogeneous groups with diverging, and often conflicting interests. They do not have a clearly defined initial state, hence problem setting is a difficult problem by itself; the final state and the way to get there are also a matter of negotiation and agreement. Many information technology projects share these characteristics nowadays. Such applications can help flesh out our findings and provide some guidance on how to address some of the open issues discussed here.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and the editors of this journal for their insightful and helpful comments that have greatly improved the present work.

REFERENCES

- ALEXANDER, C. 1971. The state of the art in design methods. *DMG Newslett.* 5, 3, 3–7.
- ARANGO, G., BRUNEAU, L., CLOAREC, J.-F., AND FEROLDI, A. 1991. A tool shell for tracking design decisions. *IEEE Software* 8, 2, 75–83.
- BAXTER, I. D. 1992. Design maintenance systems. *Commun. ACM* 35, 4 (Apr. 1992), 73–89.
- BOEHM, B., EGYED, A., KWAN, J., PORT, D., SHAH, A., AND MADACHY, R. 1998. Using the winwin spiral model: A case study. *IEEE Computer* 31, 7 (July), 33–44.
- BUCKINGHAM SHUM, S. AND HAMMOND, N. 1994. Argumentation-based design rationale: What use at what cost?. *Int. J. Hum.-Comput. Stud.* 40, 4 (Apr. 1994), 603–652.
- CARROLL, J. M. AND ROSSON, M. B. 1991. Deliberated evolution: Stalking the view matcher in design space articles. *Hum. Comput. Interact.* 6, 3-4, 281–318.
- CARROLL, J. M. AND ROSSON, M. B. 1992. Getting around the task-artifact cycle: How to make claims and design by scenario. *ACM Trans. Inf. Syst.* 10, 2 (Apr.), 181–212.
- CARROLL, J. M. AND ROSSON, M. B. 1996. Developing the Blacksburg electronic village. *Commun. ACM* 39, 12 (Dec.), 69–74.
- CARROLL, J. M., ALPERT, S. R., KARAT, J., VAN DEUSEN, M., AND ROSSON, M. B. 1994. Raison d’Etre: capturing design history and rationale in multimedia narratives. In *Proceedings of the ACM Conference on Human Factors in Computing Systems: “Celebrating Interdependence”* (CHI ’94, Boston, MA, Apr. 24–28), ACM Press, New York, NY, 192–197.
- CHANDRASEKARAN, B., GOEL, A. K., AND IWASAKI, Y. 1993. Functional representation as design rationale. *IEEE Computer* 26, 1 (Jan. 1993), 48–56.
- CONKLIN, J. AND BEGEMAN, M. L. 1988. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Trans. Inf. Syst.* 6, 4 (Oct. 1988), 303–331.
- DE LA GARZA, J. M. AND ALCANTARA, P. T., JR. 1997. Using parameter dependency network to represent design rationale. *J. Comput. Civil Eng.* 11, 2 (Apr.), 102–112.
- DENNING, P. J. 1992. Work is a closed-loop process. *Am. Sci.* 80, 4 (July-Aug.), 314–317.
- DOURISH, P. 1995. Developing a reflective model of collaborative systems. *ACM Trans. Comput. Hum. Interact.* 2, 1 (Mar. 1995), 40–63.
- FAVELA, J., WONG, A., AND CHAKRAVARTHY, A. 1993. Supporting collaborative engineering design. *Eng. Comput.* 9, 3, 125–132.

- GARCIA, A. C. B. AND DE SOUZA, C. S. 1997. ADD+: Including rhetorical structures in active documents. *Artif. Intell. Eng. Design Anal. Manufact.* 11, 2 (Apr.), 109–124.
- GARCIA, A. C. B. AND HOWARD, H. C. 1992. Acquiring design knowledge through design decision justification. *Artif. Intell. Eng. Design Anal. Manufact.* 6, 1 (Jan.), 59–71.
- GROSZ, G., ROLLAND, C., SCHWER, S., SOUVEYET, C., PLIHON, V., SI-SAID, S., BEN ACHOUR, C., AND GNAHO, C. 1997. Modelling and engineering the requirements engineering process: An overview of the nature approach. *Requir. Eng.* 2, 3, 115–131.
- GRUBER, T. R. AND RUSSEL, D. M. 1992. Derivation and use of design rationale information as expressed by designers. Tech. Rep. KSL-92-64. Knowledge Systems Laboratory, Stanford University, Stanford, CA.
- JONES, J. C. 1977. How my thoughts about design methods have changed during the years. *Des. Meth. Theor.* 11, 1, 50–62.
- KARDASIS, P. AND LOUCOPOULOS, P. 1998. Aligning legacy information systems to business processes. In *Proceedings of the 10th International Conference on Advanced Information Systems Engineering (CAiSE '98, Pisa, Italy, June)*, B. Pernici and C. Thanos, Eds. Springer-Verlag, New York, 25–39.
- KAVAKLI, V. AND LOUCOPOULOS, P. 1998. Goal-based business process analysis: Application on electricity deregulation. In *Proceedings of the 10th International Conference on Advanced Information Systems Engineering (CAiSE '98, Pisa, Italy, June)*, B. Pernici and C. Thanos, Eds. Springer-Verlag, New York, 305–324.
- KLEIN, M. 1993. Capturing design rationale in concurrent engineering teams. *IEEE Computer* 26, 1 (Jan. 1993), 39–47.
- KLEIN, M. 1997. Capturing geometry rationale for collaborative design. In *Proceedings of the 6th International IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE '97, Cambridge, MA, June)*, IEEE Computer Society Press, Los Alamitos, CA.
- LANDAUER, T. K. 1995. *The Trouble with Computers*. MIT Press, Cambridge, MA.
- LEE, J. 1991. Extending the Potts and Bruns model for recording design rationale. In *Proceedings of the 13th International Conference on Software Engineering (ICSE '91, Austin, TX, May 13–17)*, L. Belady, D. Barstow, and K. Torii, Eds. IEEE Computer Society Press, Los Alamitos, CA, 114–125.
- LEE, J. 1997. Design rationale systems: Understanding the issues. *IEEE Expert/Intell. Syst. Appl.* 12, 3 (May/June), 78–85.
- LEE, J. AND LAI, K.-Y. 1991. What's in the design rationale?. *Human-Comput. Interact.* 6, 3-4, 251–280.
- LEWIS, C., RIEMAN, J., AND BELL, B. 1991. Problem-centered design for expressiveness and facility in a graphical programming system. *Human-Comput. Interact.* 6, 3-4, 319–35.
- LOUCOPOULOS, P. AND KAVAKLI, V. 1995. Enterprise modelling and the teleological approach to requirements engineering. *Int. J. Intell. Coop. Inf. Syst.* 4, 1, 45–79.
- MACLEAN, A., YOUNG, R. M., BELLOTTI, V. M. E., AND MORAN, T. P. 1991. Questions, options, and criteria: Elements of design space analysis. *Human-Comput. Interact.* 6, 3-4, 201–250.
- MARSHALL, C. C., HALASZ, F. G., ROGERS, R. A., AND JANSSEN, W. C. 1991. Aquanet: a hypertext tool to hold your knowledge in place. In *Proceedings of the 3rd Annual ACM Conference on Hypertext (San Antonio, TX, Dec. 15–18)*, J. J. Leggett, Ed. ACM Press, New York, NY, 261–275.
- MARSHALL, C. C., SHIPMAN, F. M., AND COOMBS, J. H. 1994. VIKI: spatial hypertext supporting emergent structure. In *Proceedings of the 1994 ACM Conference on Hypermedia Technology (ECHT'94, Edinburgh, Scotland, Sept. 18–23)*, I. Ritchie and N. Guimarães, Eds. ACM Press, New York, NY, 13–23.
- MCCALL, R. J. 1991. PHI: A conceptual foundation for design hypermedia. *Des. Stud.* 12, 1, 30–41.
- MEDINA-MORA, R., WINOGRAD, T., FLORES, R., AND FLORES, F. 1992. The action workflow approach to workflow management technology. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92, Toronto, Canada, Oct. 31–Nov. 4)*, M. Mantel and R. Baecker, Eds. ACM Press, New York, NY, 281–288.

- MORAN, T. P. AND CARROLL, J. M. 1996. Overview of design rationale. In *Design Rationale: Concepts, Techniques, and Use*, T. P. Moran and J. M. Carroll, Eds. LEA computers, cognition, and work series. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 1–19.
- MÖRCH, A. 1994. Designing for radical tailorability: Coupling artefact and rationale. *Knowl.-Based Syst. J.* 7, 4 (Dec.), 253–264.
- MÖRCH, A. 1995. Application units: Basic building blocks of tailorable applications. In *Human-Computer Interaction: 5th International Conference (EWHCI '95)*, B. Blumenthal and C. Unger, Eds. Springer-Verlag, New York, 45–62.
- OLSON, G. M., OLSON, J. S., STORRØSTEN, M., CARTER, M., HERBSLEB, J., AND RUETER, H. 1996. The structure of activity during design meetings. In *Design Rationale: Concepts, Techniques, and Use*, T. P. Moran and J. M. Carroll, Eds. LEA computers, cognition, and work series. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 217–239.
- PENA-MORA, F. AND VADHAVKAR, S. 1997. Augmenting design patterns with design rationale. *Artif. Intell. Eng. Design Anal. Manufact.* 11, 2 (Apr.), 93–108.
- PENA-MORA, F., SRIRAM, D., AND LOGCHER, R. 1995. Design rationale for computer-supported conflict mitigation. *J. Comput. Civil Eng.* 9, 1, 57–72.
- POHL, K. 1996. *Process Centered Requirements Engineering*. John Wiley and Sons Ltd., Chichester, UK.
- POTTS, C. 1989. A generic model for representing design methods. In *Proceedings of the 11th International Conference on Software Engineering* (Pittsburgh, PA, May 15–18), L. Druffel, Ed. ACM Press, New York, NY, 217–226.
- POTTS, C. 1996. Supporting software design: integrating design methods and design rationale. In *Design Rationale: Concepts, Techniques, and Use*, T. P. Moran and J. M. Carroll, Eds. LEA computers, cognition, and work series. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 295–321.
- POTTS, C. AND BRUNS, G. 1988. Recording the reasons for design decisions. In *Proceedings of the 10th International Conference on Software Engineering* (Singapore, Apr. 11–15), T. N. Nam, Ed. IEEE Computer Society Press, Los Alamitos, CA, 418–427.
- POTTS, C., TAKAHASHI, K., AND ANTÓN, A. I. 1994. Inquiry-based requirements analysis. *IEEE Software* 11, 2 (Mar. 1994), 21–32.
- RAMESH, B. AND DHAR, V. 1992. Supporting systems development by capturing deliberations during requirements engineering. *IEEE Trans. Softw. Eng.* 18, 6 (June 1992), 498–510.
- RITTEL, H. AND WEBBER, M. M. 1984. Planning problems are wicked problems. In *Developments in Design Methodology* John Wiley and Sons, Inc., New York, NY, 135–144.
- ROBINSON, W. N. AND VOLKOV, S. 1997. A meta-model for restructuring stakeholder requirements. In *Proceedings of the 1997 international conference on Software engineering (ICSE '97, Boston, MA, May 17–23, 1997)*, W. R. Adrion, Ed. ACM Press, New York, NY, 140–149.
- SCHÖN, D. A. 1983. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, Inc., New York, NY.
- SHIPMAN, F. M. AND MARSHALL, C. C. 1993. Formality considered harmful: Experiences, emerging themes, and directions. Tech. Rep. CU-CS-648-93. Department of Computer Science, University of Colorado at Boulder, Boulder, CO.
- SHIPMAN, F. M. AND MCCALL, R. 1994. Supporting knowledge-base evolution with incremental formalization. In *Proceedings of the ACM Conference on Human Factors in Computing Systems: "Celebrating Interdependence"* (CHI '94, Boston, MA, Apr. 24–28), ACM Press, New York, NY, 285–291.
- SHUM, S. 1991. Cognitive dimensions of design rationale. In *People and Computers VI: Proceedings of HCI '91*, D. Diaper and N. V. Hammond, Eds. Cambridge University Press, New York, NY, 331–344.
- STROUSTRUP, B. 1994. *The design and evolution of C++*. Addison-Wesley Longman Publ. Co., Inc., Reading, MA.
- VIVACQUA, A. S. AND GARCIA, A. C. B. 1997. MultiADD: A multiagent active design document model to support group design. In *Artificial Intelligence*, WALDO, J., Ed. 1993. *The evolution of C++: language design in the marketplace of ideas*. MIT Press, Cambridge, MA.

- WINOGRAD, T. AND TABOR, P. 1996. Software design and architecture. In *Bringing design to software*, T. Winograd, Ed. ACM Press, New York, NY, 10–15.
- ZUBOFF, S. 1988. *In the Age of the Smart Machine: The Future of Work and Power*. Basic Books, Inc., New York, NY.

Received: February 1998; revised: June 1998, November 1998, and September 1999;
accepted: October 1999