HUMAN-COMPUTER INTERACTION, 1991, Volume 6, pp. 251-280

What's in Design Rationale?

Jintae Lee and **Kum-Yew Lai**
Massachusetts Institute of Technology

ABSTRACT

A few representations have been used for capturing design rationale. To understand their scope and adequacy, we need to know how to evaluate them. In this article, we develop a framework for evaluating the expressive adequacy of design rationale representations. This framework is built by progressively differentiating the elements of design rationale that, when made explicit, support an increasing number of the design tasks. Using this framework, we present and assess DRL (Decision Representation Language), a language for representing rationales that we believe is the most expressive of the existing representations. We also use the framework to assess the expressiveness of other design rationale representations and compare them to DRL. We conclude by pointing out the need for articulating other dimensions along which to evaluate design rationale representations.

Authors' present address: Jintae Lee and Kum-Yew Lai, Center for Coordination Science, Massachusetts Institute of Technology, E40-175, 1 Amherst Street, Cambridge, MA 02139.

CONTENTS

1. INTRODUCTION

As the articles in this issue point out, an explicit representation of design rationales can bring many benefits. Such a representation can lead to a better understanding of the issues involved (Conklin & Yakemovic, 1991 [this issue]; Lewis, Rieman, & Bell, 1991 [this issue]), of the design space (MacLean, Young, Bellotti, & Moran, 1991 [this issue]), and of the principles underlying human-computer interaction (Carroll & Rosson, 1991 [this issue]). It can also provide a basis for learning, justification, and computational support in design (Fischer, Lemke, McCall, & Morch, 1991 [this issue]; Lee, 1990a). The extent to which we can actually reap these benefits, however, depends largely on the language we use for representing design rationales. If, for example, design rationales were represented in free text, the benefits we obtained from it would not be different from what we already get from the notes on paper that we take in design meetings. Also, the kinds of computational support that we can provide depends on what a representation makes explicit and how formal the representation is. A few systems have been built and actually used to capture design rationales or arguments (Conklin & Begeman, 1988; Fischer, McCall, & Morch, 1989; Kunz & Rittel, 1970; Lee, 1990a, 1990b; McCall, 1987), and most of them used representations based on the earlier studies of design activity (Kunz & Rittel, 1970) or of argumentation (Toulmin, 1958). However, there is no systematic attempt to justify the choice of these representations or to discuss the rationale for using them.

This article is motivated by the following questions: How adequate are the existing representations? Do they allow us to represent easily what we want to

represent? In general, how do we evaluate a language for representing design rationales? This article attempts to answer these questions by identifying the elements of design rationale that could be made explicit and by exploring the consequences of making them explicit. Laying out these elements provides a framework for placing the existing meanings of design rationale in perspective, as well as providing a framework for evaluating a representation language for design rationales, as we hope to show.

We proceed in the following way. In the next section, we identify the tasks that we might want to support using a design rationale representation. Throughout the article, we use these tasks as a reference against which we evaluate the representations that we discuss. In Section 3, we characterize design rationale by presenting progressively richer models. We start with a simple model of design rationale, where an artifact is associated with a body of reasons for the choice of that artifact. We then elaborate this simple model by incrementally differentiating and making explicit what is implicit in the body of reasons. As we do so, we discuss what each resulting model allows us to do. These models of design rationale provide a framework in which to define the scope of a representation and its adequacy within its scope.[1] In Section 4, we present a language called DRL for representing rationales and use the models to evaluate DRL. In Section 5, we consider other existing languages for representing design and explore some open problems for future research.

2. WHAT DO WE WANT TO DO WITH DESIGN RATIONALE?

To evaluate a representation, we need to know what tasks it is designed to support. The tasks that a design rationale representation can or should support can be described in many ways at different levels of abstractions. For example, Mostow (1985) listed the following tasks: documentation, understanding, debugging, verification, analysis, explanation, modification, and automation. Fischer et al. (1991 [this issue]) point out that documenting design rationales can support maintenance and redesign of an artifact, reuse of the design knowledge, and critical reflection during the design process. MacLean et al. (1991 [this issue]) list two major benefits from design rationale representation: aid to reasoning and aid to communication. The tasks of achieving these benefits are elaborated further in terms of subtasks,

[1] In this article we use the terms *model* and *representation* in the following way. A model is a conceptual structure, and a representation is a linguistic manifestation of a model. Because the same structure can be described in many ways, a model can have several representations. Therefore, when we want to discuss a structure independent of a particular way of describing it, we use the term *model*. On the other hand, we use the term *representation* to refer to a particular notation for describing the structure.

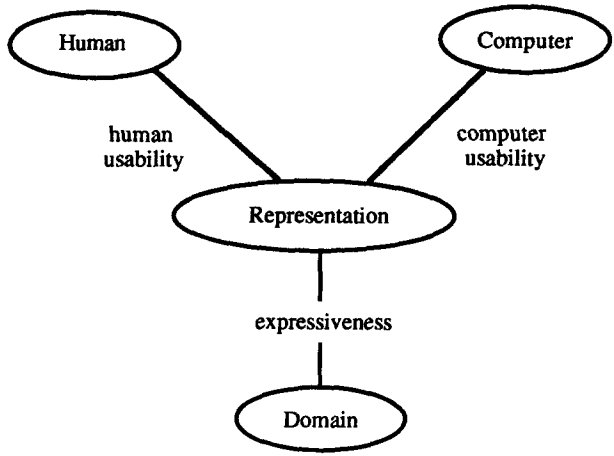
such as enabling the designers to envisage the available design alternatives in a more structured way, with the arguments for and against them.

Another way of characterizing the tasks is to list the questions we often need to answer to accomplish the general tasks mentioned before in the design process. To the extent that we want our design rationale representation to help answer these questions, answering these questions becomes the task that the representation should support. The following is a set of representative questions that we gathered from our experiments with design rationale (Lee, 1991), from walking through examples (Lewis et al., 1991 [this issue]), and from creating scenarios (Carroll & Rosson, 1990).

- What is the status of the current design?
- What did we discuss last week, and what do we need to do today?
- What are the alternative designs, and what are their pros and cons?
- What are the two most favorable alternatives so far?
- Sun Microsystems just released their X/NeWs server. How would the release change our evaluations?
- What if we do not consider portability?
- Why is portability important anyway?
- What are the issues that depend on this issue?
- What are the unresolved issues? What are we currently doing about them?
- What are the consequences of doing away with this part?
- How did other people deal with this problem? What can we learn from the past design cases?

This list of questions is by no means complete, as there are many possible paths we did not walk through and many scenarios we did not construct. We also left out those questions that, although important, do not seem to be the job of design rationales to answer (e.g., How can we compute the total cost of this design?). Nevertheless, the questions in the list provide a useful framework for assessing the expressiveness of the different representations. When we discuss the limited or increased expressiveness of a given representation, we refer to those questions that can or cannot be answered as a result. If our task includes answering a question that is not represented in the list, then we can always evaluate the representations by asking whether they would support answering the question and, if not, what additional objects, attributes, or relations would have to be made explicit.

Figure 1. **Elements in computer-supported activities.**



We want to emphasize further that we are assessing only the expressiveness of the existing representations of design rationales. That it is desirable to answer the questions in the list does not mean that any representation for design rationales should support answering all of these questions. Each representation must weigh the costs and benefits involved in tradeoffs among three general dimensions: expressiveness, human usability, and computational tractability (see Figure 1). These tradeoffs should in turn be motivated by the tasks that are intended to be accomplished using the representation. In short, we are not dictating what any existing representation should or should not have. However, we do hope that the analysis presented in this article will help designers of representations for design rationales be more conscious of what their languages can or cannot express and why.

To be sure, we cannot separate our concern with expressiveness entirely from other concerns such as human usability or computational tractability. For example, if a language is meant to be used by people to capture design rationales but is too complex for people to manage, then there is not much point in evaluating its expressiveness. Whether any of the representations we discuss falls into that category is an empirical question. All the languages discussed here actually have been used by people, but that is no guarantee that they will all succeed at their "industrial strength" use (Conklin & Yakemovic, 1991 [this issue]). Nevertheless, we believe it would be difficult to evaluate tradeoffs among the three dimensions without calibrating individual dimensions such as expressiveness (cf. Levesque & Brachman, 1985, on tradeoffs between expressiveness and computational tractability for general knowledge representation).

3. MODELS OF DESIGN RATIONALE

What is design rationale? The term *design rationale* is currently used in at least three different ways: a historical record of the reasons for the choice of an artifact (Yakemovic & Conklin, 1990), a set of psychological claims embodied by an artifact (Carroll & Rosson, 1990), and a description of the design space (MacLean, Young, & Moran, 1989).[2]

Design rationale often means the historical record of the analysis that led to the choice of the particular artifact or the feature in question. To illustrate, let us take as an example a particular feature of the Macintosh operating system, namely, the placement of all the window commands in the global menu bar at the top of the screen. By a window command, we mean a command specific to a window; for example, SAVE is a window command that saves the contents of the window. A design rationale for this feature in the sense of historical record would be something like:

> The issue of where to put the window commands was raised by Mark on January 20. Kevin proposed the idea of incorporating them into the global menu bar at the top of the screen and pointed out that it saves screen space (e.g., as opposed to putting the commands on each window, as in the Star environment). Julie objected because it requires a long mouse travel from the currently active window in executing a command. But, we decided to have the global menu bar anyway because people generally agreed that the advantage, together with others such as more efficient implementation, outweigh the objection.

We can provide more structure to this historical record, as we discuss in the rest of the article. Such structure is usually designed to make explicit the logical structure (e.g., an argument supports a proposal) and/or the historical structure (e.g., a proposal replaces another proposal).

Another meaning of design rationale is the set of psychological claims embodied by an artifact (Carroll & Kellogg, 1989; Carroll & Rosson, 1991 [this issue]), that is, claims that would have to be true if the artifact is to be successful or claims about psychological consequences for the users of the artifact. These claims are different from the historical record; the claims need not be present in the historical record; even if they were, they would have to be extracted from the record and formulated in a testable form. For example,

[2] The representation used in Yakemovic and Conklin (1990) describes logical as well as historical aspects of design rationale, as we discuss later. We associate their work, as well as that of Lee (1990a) and Potts and Bruns (1988), with the historical record only because one of their goals is to capture and document the actual process of design.

the design rationale in this sense would be something like: "The global menu makes the environment easier to use because it reduces screen clutter," or "Dimming the irrelevant items in the global menu makes it easier to learn about the commands."

The third meaning of design rationale is one used by MacLean et al. (1989), namely, how a given artifact is located in the space of possible design alternatives: What are the other possible alternatives? How are these alternatives related? What are the tradeoffs among them? In our Macintosh example, the design rationale in this sense would be some description of the logically possible alternatives for placing window commands, how they are related, and what the tradeoffs are. It is often difficult to provide such a description in a systematic way, but an example is found in Card, Mackinlay, and Robertson (1990) and in Mackinlay, Card, and Robertson (1990), which provides a vocabulary of the primitives and a set of composition operators for describing the design space of possible input devices. This meaning of design rationale seems different from the first meaning in its emphasis on design rationale not being a record but a construction and from the second in its emphasis not on a particular artifact but on the relation among possible alternatives.

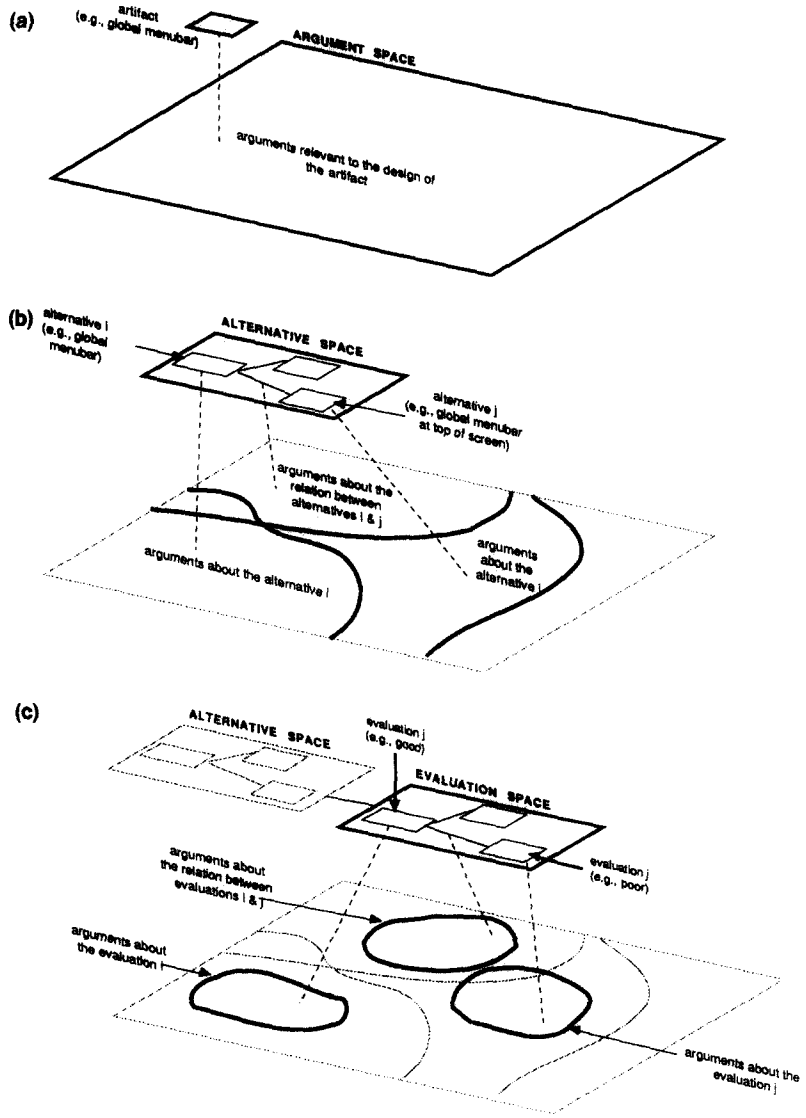
We now develop a series of progressively richer models of design rationale, which provides a framework in which we can place the three different meanings of design rationale. The first two meanings are discussed next. The third meaning of design rationale, as a possibility space, is discussed shortly after.

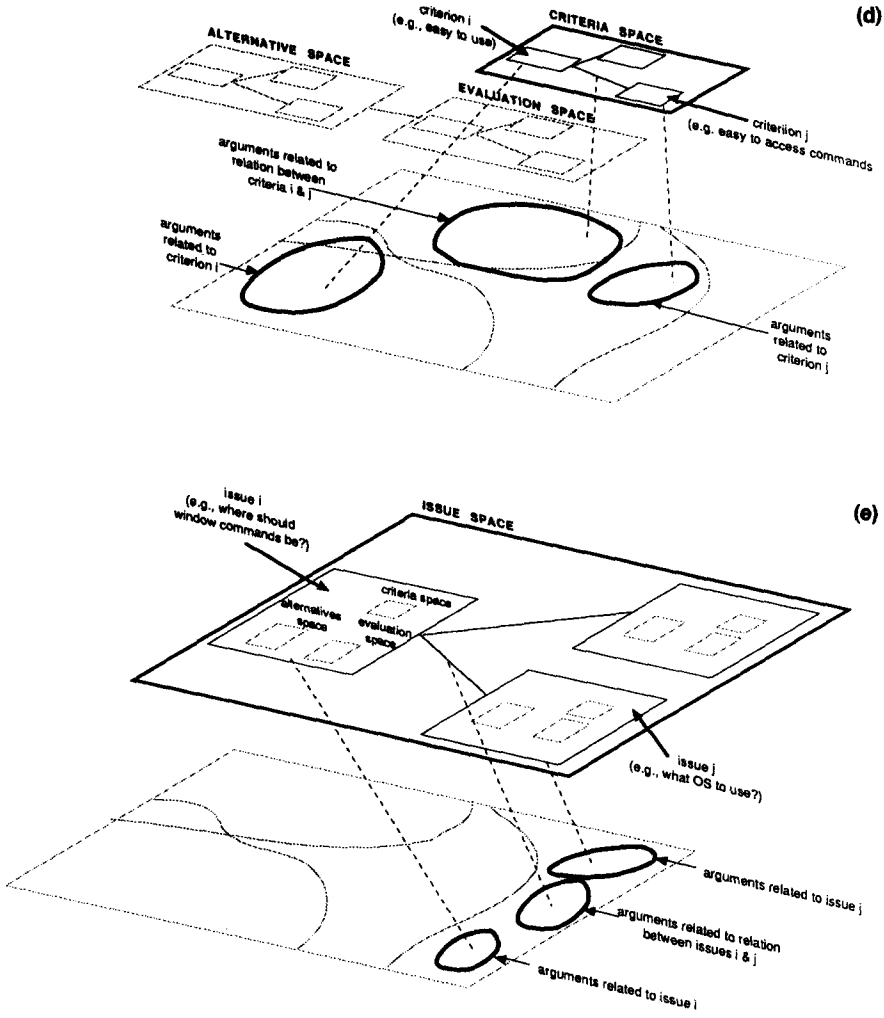
Design rationale in the most general sense is an explanation of why an artifact is designed the way it is. So, in our first model of design rationale, an artifact is associated with a body of reasons as shown in Figure 2a.

There are different kinds of reasons that we can give for an artifact. The reasons can be historical or logical, roughly corresponding to the meanings of design rationale, respectively, as a historical record and as a set of claims embodied by an artifact.[3] The record of the process that led to the choice of an artifact tells us one kind of reason why that artifact was chosen. The previously shown free text example about the window commands is an example. If we wanted a logical justification, we would have to extract it from the record, but at least such a record tells us the historical circumstances and sequence that led to the design and provides a basis from which to infer the logical reasons. On the other hand, we can represent the logical reasons

[3] We believe that the distinction between historical and logical reasons breaks if we push it too far because a purely historical record per se does not really give us a reason. It would give us a reason only to the extent that we can extract some logical structure out of it. Nevertheless, we believe that the distinction is useful for the purpose of evaluating representations because, for a given representation, we would like to know what it makes explicit and what we have to infer from it.

Figure 2. **Progressively more differentiated models of design rationale. (a) Model 1: An artifact is associated with a body of all the arguments relevant to the design of the artifact. (b) Model 2: Alternatives and their relations are made explicit, and the arguments about individual alternatives can be differentiated. (c) Model 3: Evaluation measures used and their relations are made explicit, and the arguments about them can be differentiated. (d) Model 4: Criteria used for evaluation and their relations are made explicit, and the arguments about them can be further differentiated in the argument space. (e) Model 5: Individual issues are made explicit, each of which contains the alternatives, evaluations, and criteria used in discussing the issue. A part of the argument space includes the meta-arguments about the issues and their relations.**





directly, that is, the reasons justifying the choice of an artifact no matter how or in what order they were articulated. The set of claims embodied by an artifact is an example because these claims would justify the design of the artifact. These claims are logical also in the sense that the context in which they are true has to be made explicit. For example, a claim should not say, "The global menu is better because it leads to smaller implementation" if it really means, "The global menu is better in the context of the Macintosh because it leads to smaller implementation. This is important in a system like the Macintosh, which has a small memory." Extracting these logical reasons

is not an easy task; once identified, however, they provide the advantages of being testable and general.

The internal structure of these reasons can be made explicit to different degrees. At one extreme, the reasons can be completely undifferentiated. An example is the natural language description that we gave earlier as an example of a historical record. If we were to make the historical relations more explicit, we could differentiate further by making explicit these roles and relations such as: **Initiator, Second Motioner, Initiates,** and **Replaces.** An example that is not historical is the representation used by Carroll and Rosson (1991 [this issue]) for describing the claims embodied in an artifact. In this representation, the claims themselves are represented in natural language, but the claims are grouped by the questions they answer: What can I do? How does that work? and How do I do this? We can also imagine a representation where the logical support relations can be made more explicit by providing such constructs as **Logically Implies, Supports, Denies, Qualifies,** and **Presupposes.** We use the term *argument space* to refer what we have called a body of reasons, because the reasons are captured either as a historical record of the various arguments relevant for the design of the artifact or logical arguments underlying the design.

There is much we can do with our first model of design rationale. A representation based on this model can help us answer the question, What did we discuss last week, and what do we need to do today? Such a representation can also help us answer the questions: How did other people deal with this problem? and Can we learn from the past design cases? Carroll and Rosson (1991 [this issue]) provide a good example. They report in detail how their representation of design rationales, mentioned earlier, suggested many issues for redesigning an artifact (the View Matcher in Smalltalk). They discuss how these issues can be couched as a design hypothesis, which can be tested and compiled to form, in the long run, "a contextualized science out of practice" of human-computer interaction.

Our first model, however, does not help very much with the other questions, although we should qualify this statement immediately. Saying that it does not help much is not to say that we cannot answer these questions. Of course, if the user works hard enough, and as long as the representation based on the model has enough information captured, even in the form of natural language free text, we can answer these questions. So the real issue is how much the model itself helps us answer these questions either by helping us see the structure better or by enabling us to define computational services that help us answer the questions. We will see how more differentiated models allow us to answer these questions more easily, although they increase the cost in some other ways (see Conklin & Yakemovic, 1991 [this issue]).

Our second model (see Figure 2b) differs from the first by making multiple alternatives and their relations explicit. Design involves formulating several

alternatives, comparing them, and merging them, as many of the questions in our list indicate. In our first model, only a single alternative is made explicit at a given time, and the multiple alternatives are present only implicitly in the argument space. Our second model makes these alternatives explicit, including the ones that have been rejected. Once the alternatives become explicit, we can talk about their attributes (e.g., current status such as "rejected" or "waiting for more information"), make the relations among the alternatives explicit (e.g., specialization, historical precedence), define computational operations on them (e.g., comparing alternatives, displaying the alternatives that specialize this alternative), or even argue about whether an alternative is worth considering. The alternatives, other than the one finally chosen, are interesting because many of the issues and the knowledge used in evaluating them are useful in other contexts, for example, when situational constraints change. We use the term *alternative space* to refer to this set of multiple alternatives and their relations.

These relations among the alternatives can also be historical or logical. Historical relations may be not only the linear sequence that we usually describe as versions but also more complex relations such as layers and contexts (Bobrow & Goldstein, 1980). The logical relations may include **Specializes, Generalizes, Elaborates,** or **Simplifies.** Or alternatives can be related through a design space (Mackinlay et al., 1990). To the extent that we want a representation to stand for these different alternatives and their relations, we say that the alternative space is within the scope of the representation. gIBIS, for example, seems to include the alternative space within its scope because one of its goals is "to capture alternative resolutions (including those which are later rejected), [and] trade-off analysis among these alternatives" (Conklin & Begeman, 1988, p. 304). The constructs in gIBIS for representing the alternative space consist of: **Position,** with which we can describe multiple alternatives, and the specialization relation among the **Positions.**

By now, we have an alternative space connected to the argument space, as shown in Figure 2b. For each of the alternatives, there are arguments describing the reasons for its current evaluation, just as in our first model there are arguments describing the evaluation status of that single alternative, that is, that it was chosen. Some of the arguments can be shared; for example, an argument can support an alternative while denying another; so it is better to think of the arguments about the different alternatives forming a single large argument space, as shown in Figure 2b.

With the representation of the alternative space, we can imagine how we can make a system help us answer some of the questions posed in Section 2. To answer "What are the alternative designs, and what are their pros and cons?," we can associate an argument space with each of the alternatives through the links such as **Supports** or **Objects To,** as in gIBIS. To answer

"Why do we even consider this alternative, and how is it related to the one that we discussed last week?," we need to use some historical relations (e.g., **Replaces**) or structural relations (e.g., **Is a part of**) among the alternatives.

Once we make explicit multiple alternatives, however, we need to articulate more carefully what the argument space is about (Figure 2b). In our first model, when we had a single artifact (i.e., the chosen one), the argument space contained reasons for the choice of that artifact. Similarly, the arguments for the other alternatives are about why they were not chosen or, to generalize, why they have their particular evaluation status (e.g., "still in consideration," "waiting for more information," "rejected"). These evaluation statuses could be nominal categories (e.g., like the previous examples), ordinal categories (e.g., "very good," "good," and "poor"), or a continuous measure (e.g., the probability that the alternative will achieve a given set of goals).

Therefore, we introduce the *evaluation space* (Figure 2c), where the evaluation statuses are made explicit and interrelated. Usually, we do not and need not specify any elaborate relation among the evaluation measures we use. Often, the implicit ordinal relation among these values (e.g., "very good," "good," "poor," "very poor") is sufficient when we leave it for the human user to assign these values to the alternatives. However, if we want to define any computational service that manages these values, for example, that automatically propagates and merges them to produce a higher level summary, then we need to be very careful about what these values mean. We need to specify the units of measurement, a calculus for combining them, and a model specifying what they mean. Even in the case where these actions are left to humans, for example, if the human user is expected to combine these values to produce a higher level summary measure, then we need to set down what these values mean so that their interpretation does not become arbitrary. To the extent that a representation makes explicit the parts of the evaluation space needed for merging individual evaluations into overall evaluations, we can now answer questions such as: "What are the two most favorable alternatives so far?" and "Sun Microsystems just released their X/NeWs server. How would the release change our evaluations?" We can also explain how an evaluation was made by pointing to the arguments in the argument space behind the artifact in question and by explaining how this particular evaluation measure is derived or computed from them or related to other measures.

Making the evaluation space explicit allows us to differentiate two components of the argument space: (a) arguments about why an alternative has its current evaluation status and (b) arguments about the alternatives themselves (e.g., why we should or should not even consider an object as an alternative or whether this alternative is really a special case of another alternative). That

is, as shown in Figure 2c, these different kinds of arguments can be differentiated in the argument space.

Our models so far do not make explicit the criteria used in producing an evaluation. However, the criteria used for the evaluation and their relations are usually quite important to represent explicitly. For example, it is important to know that the argument "We do not need to duplicate menu items" is an argument for the alternative "Global menu at the top of the screen" because of the goal of reducing screen clutter, which is used as a criterion for evaluation. By making this criterion explicit, we can group all the arguments that appeal to this criterion and weigh them against one another. If the criterion changes or becomes less important, then we can do appropriate things to all the arguments that presuppose the goal (e.g., making these arguments less important). Knowing how this criterion is related to others (e.g., "reducing screen clutter" is a way of achieving "easy to use") also allows us to assign proper importance to this criterion or to change its importance when the related criteria changes. We use the term *criteria space* to refer to these criteria and their relations. As Figure 2d shows, once we have the criteria space explicit, we can further differentiate the argument space by grouping those arguments that are about the criteria and their relations.

Hence, it is important that a language whose scope includes the criteria space represent the different attributes of the criteria and the relationship among them. For example, it should allow us to represent the importance of these criteria and the synergistic or tradeoff relations among them. Some criteria can be subcriteria of another in the sense that satisfying them facilitates the satisfaction of the latter. These subcriteria can be related among themselves in various ways. They can be mutually exclusive in the sense that satisfying one makes it impossible to satisfy others. They can be independent of each other in the sense that satisfying one does not change the likelihood of satisfying others. These subcriteria can be related to their parent criterion in various ways as well. They can be exhaustive in the sense that satisfying all of them is equivalent to satisfying the parent.

With the criteria space represented, we can now see how the system might be able to help us answer questions such as: What if we do not consider portability? or Why is portability important anyway? The answer might be, "If we give up the goal of portability, then the evaluation of the alternative X changes to 'High' because all these claims that argue against X were based on the importance of portability." Or, "Portability is important because it is a subgoal of another important goal, 'Have a wide distribution.' " These answers can be derived from a representation if the representation makes explicit the relation among evaluations, criteria, and arguments. Of course, representation of the criteria is not sufficient for answering these questions. It is not obvious how these questions can be answered, even if some parts of the

criteria space are represented explicitly. However, the explicit representation of the criteria space seems a necessary condition if we are to answer these questions. At least, we would have the information necessary to define an operation that will give or suggest the answers to these questions. We briefly describe a few examples of such operations in Section 4.

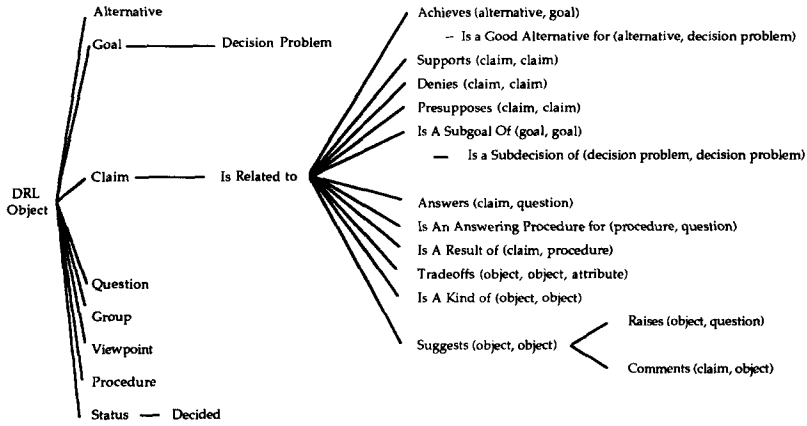
So far, we have identified and discussed the structure of a single decision underlying an artifact; namely, Which of the alternative designs should we choose? However, with the representation of such local structures alone (viz., its argument space, alternative space, and criteria space), we still cannot ask some of the questions in the list such as: What are the unresolved issues, and what are we currently doing about them? What are the issues that depend on this issue? To answer such questions, we need a more global picture of how individual issues are related. A decision often requires and/or influences many other decisions. For example, a decision can be a subdecision of another if the latter requires making the first decision. A decision can be a specialization of another if the first decision is a more detailed case of the second. It is important to capture how these decisions are related, and we use the term *issue space* to refer to them. A unit in this issue space is, therefore, a single decision that has as its internal structure the other spaces, as shown in Figure 2e. Once we have an issue as an explicit element, we can associate the attributes such as "status" and "actions taken" with issues and answer questions such as "What are the unresolved issues, and what are we currently doing about them?" Representing the dependency relation among the issues will allow us to answer the question "What are the issues that depend on this issue?"

There are still some questions that we have not yet covered, such as: How did other people deal with this problem? Can we learn from the past design cases? We argue, however, that the five spaces identified so far—the spaces of arguments, alternatives, evaluations, criteria, and issue—can contain enough information to answer these questions. We support this argument by showing how these questions can be answered with a language that we have developed for representing design rationales. This language, called DRL, is presented in the next section and is evaluated with respect to the five spaces described in this section.

4. DRL (DECISION REPRESENTATION LANGUAGE)

DRL (see Lee, 1990a) is a language that we have developed for representing and managing the qualitative elements of decision making: for example, the alternatives being considered, their current evaluations, the arguments responsible for producing these evaluations, and the criteria used for the evaluations. We call *decision rationale* the representation of these qualitative elements, and we call a *decision rationale management system* a system

Figure 3. **The DRL vocabulary.**



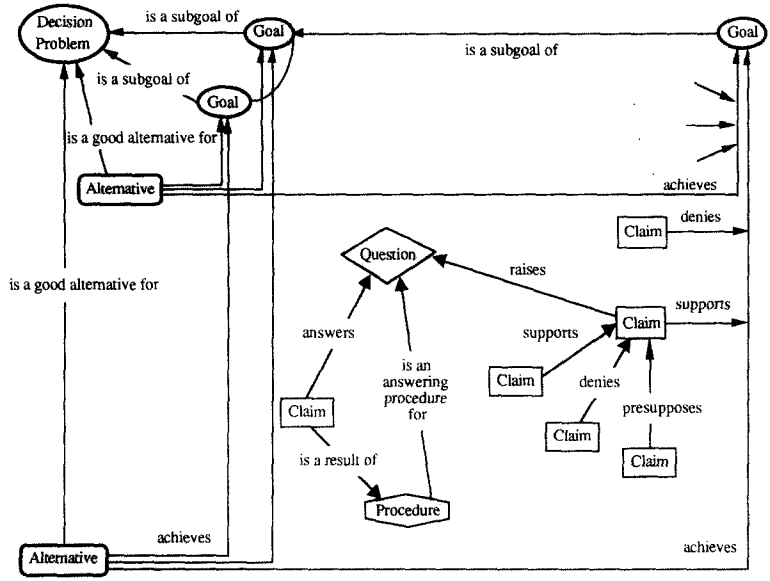
that provides an environment for capturing decision rationale and computational services using it. Decision rationale in our sense does not capture some important aspects of design rationale. For example, a design rationale may include the deliberations about how to generate the alternative designs. The scope of DRL, at least for now, does not include the representation of such deliberations.[4] The exact relation between decision rationale and design rationale, however, has yet to be articulated. Nevertheless, we believe that DRL is the most expressive language that has been used for representing design rationales and that it overcomes many of the limitations in the existing languages in a way that is still simple enough for the user. In this section, we evaluate DRL as a design rationale representation language and point out its strengths and limitations as such.

4.1. Description of DRL

Figure 3 shows the object types that form the vocabulary of DRL. Objects of type **Is Related to** and its subtypes can be used to link other objects. For instance, an achieves relation can be used to link an **Alternative** object to a **Goal** object. The legal types that can be linked are shown inside the parentheses following the names of the relations. Figure 4 shows graphically

[4] That is not to say that DRL does not help us generate alternative designs. It does in a couple of ways. DRL allows people to argue about the existing alternative designs, thus helping them to see more clearly their strengths and limitations. It also helps people retrieve the past decisions that contain useful alternative designs that are still useful for the current decision or those that can be so adapted. Furthermore, DRL can represent the relationship between the existing alternative and the new alternative that may have been derived from it. However, being able to generate a new design alternative is still different from representing the rationales for how it was generated.

Figure 4. **The structure of a decision graph.**

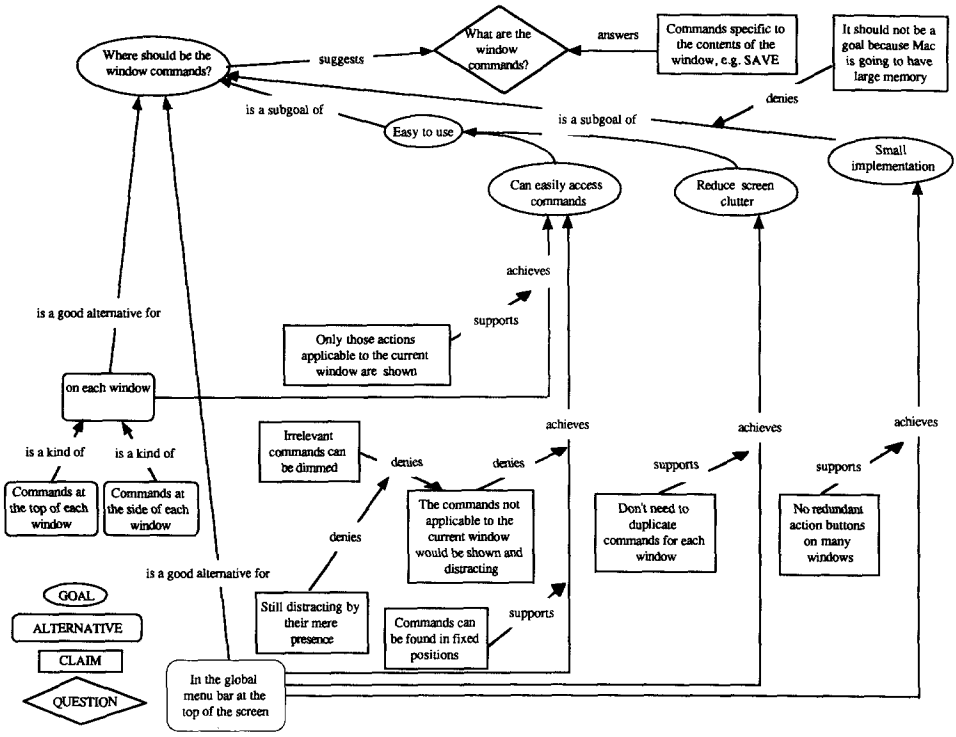


how DRL objects can be linked to each other. Figure 5 shows an example decision rationale represented in DRL. We describe the basic features of DRL briefly and discuss in detail how they allow us to represent the five spaces of design rationale. One should note that the following is not a description of the way the user would use DRL. The actual interface is briefly described later.

A qualification is in order before we proceed further. The vocabulary of DRL presented in this section is a minimal set in the following sense. The initial set included those objects and relations (e.g., alternatives, goals, and claims) needed to represent the important elements of the spaces discussed in the previous section. Then the initial set has been augmented with additional constructs only when they were needed and judged to be useful for *generic* decision tasks, that is, useful in decisions independent of any particular domain. As such, the DRL constructs presented here are meant to be specialized and augmented further for a specific set of tasks. For example, Lee (1991) describes an extension of DRL that supports the task of managing rationales in software engineering.

A **Decision Problem** represents the problem that requires a decision (e.g., where to place the window commands). An **Alternative** represents an option being considered: for example, "In the global menu at the top of the screen." A **Goal** represents a desirable state or property used for comparing the alternatives. A **Goal** such as "Where should the window be?" is elaborated

Figure 5. **An example rationale in DRL.**



further in terms of its subgoals. For example, "Easy to use" is elaborated into two subgoals, "Can easily access command items" and "Reduce screen clutter."[5] Every relation in DRL is a subclass of **Claim**, as shown in Figure 3. For example, the rightmost **Achieves** link in Figure 5 represents the **Claim** that the **Alternative**, "the global menu at the top of the screen," achieves the **Goal**, "Reduce screen clutter."

We evaluate an **Alternative** with respect to a **Goal** by arguing about the **Achieves** relation between the **Alternative** and the **Goal**, that is, the claim that the **Alternative** achieves the **Goal**. We argue about a **Claim** by producing other **Claims** that **Support** or **Deny** the **Claim** or by qualifying the **Claim** by pointing out the **Claims** that it **Presupposes**. Each **Claim** has the following attributes: evaluation, plausibility, and degree. The evaluation of a **Claim**, represented by the value of its evaluation attribute, is a function of both of its plausibility and degree attribute values. The plausibility of a **Claim** tells us

[5] Because goals are described by desired states of the world, the exact meaning of the "Easy to use" subgoal should be "The window commands should be at a place where they are easy to use."

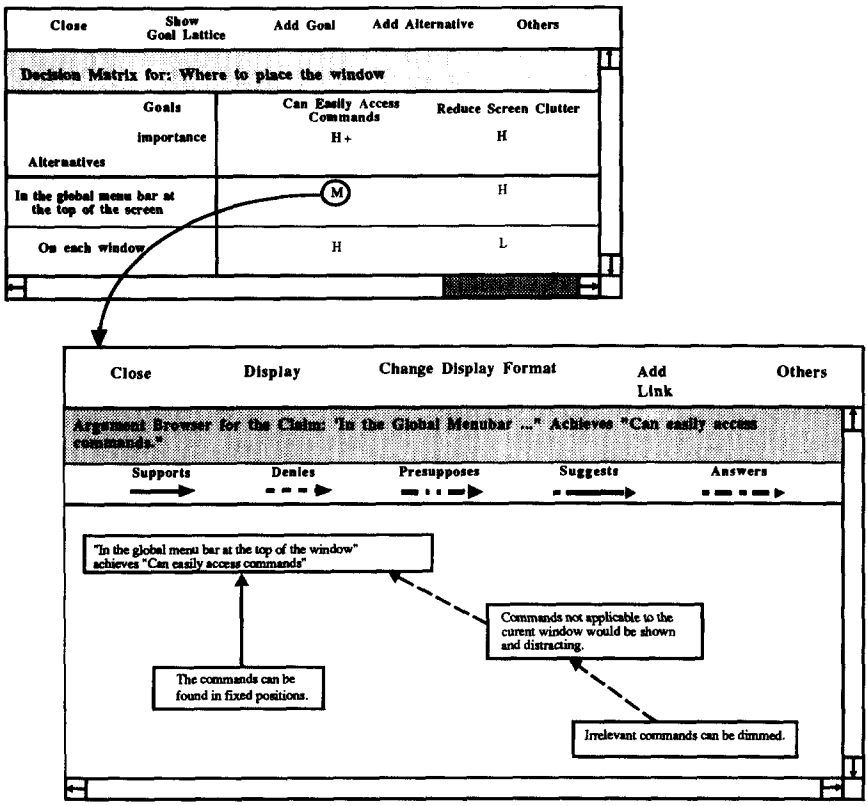
how probable it is for the claim to be true, and the degree of a **Claim** tells us to what extent it is true. For example, the degree of the **Achieves** link between the **Alternative** and the **Goal** tells us to what degree the alternative achieves the goal in question. The overall evaluation of an alternative is represented by the degree attribute value of the **Is a Good Alternative for** link between the **Alternative** and the **Decision Problem**, that is, the claim that the alternative is a good alternative resolution for the issue. This degree is a function of the degrees of the **Achieves** claims that link the **Alternative** to the different **Goals**. Not all of the three attributes have to be used for the evaluation. For example, we might require that a **Claim** be entered only if its plausibility is above a certain threshold and ignore the plausibility once the **Claim** has been entered. In that case, we can do away with the plausibility attribute, and the evaluation and the degree attributes become synonymous.

There are other auxiliary objects in DRL. A **Group** object groups a number of objects and has the attribute, "member relations," which tells us how the objects are related. A relation can take a **Group** of objects rather than a single object. For example, a **Goal** may be related to a **Group** of other **Goals** through a **Is a Subgoal of** link. The other objects in DRL, such as **Question**, **Procedure**, and **Viewpoint**, represent somewhat auxiliary aspects of decision making such as the questions raised and the procedures used for answering the questions. More details of DRL may be found in Lee (1990a).

DRL has been partially implemented in a system called SIBYL, which runs on top of Object Lens (Lai, Malone, & Yu, 1988). Although the previous description of DRL may seem complex, the actual user interface provided by SIBYL for using DRL is quite simple, and SIBYL has been used for real group decision tasks such as designing a workplace layout. For example, SIBYL makes it easy to create objects like a **Decision Problem**, **Goals**, and **Alternatives** by providing context-sensitive menus and template editors. Once a **Decision Problem** and some of its **Goals** and **Alternatives** are specified, SIBYL displays them in a matrix such as that shown in Figure 6. By mouse clicking on a cell of the matrix, the user gets the menu of all the actions that can be performed on the selected object. For example, by mouse clicking on a **Goal**, the user can get a menu containing action items such as creating a new subgoal or displaying a goal tree showing how this **Goal** is related to other **Goals**.

Figure 6 also shows an argument browser displayed when the user chooses the action, "Display Arguments," from a pop-up menu that appears when one of the evaluation cells is selected. The argument browser shows, in a network format, all the **Claims** that provide reasons for the evaluation, that is, all the claims related to the surrogate claim that the system automatically generates to be argued about, namely, that a given alternative achieves a given goal. By mouse clicking on an object in the argument browser, the user gets a pop-up menu of all the operations that can be performed on the object: such as "Add a Supporting Claim," "Add a Denying Claim," and "Add a Qualifying Claim."

Figure 6. User interface for SIBYL.



When the user chooses one of these actions, the template editor containing a new Claim object appears, and the new object is added to the argument browser with an appropriate link to the object chosen. The user interface that SIBYL provides for using DRL is described in more detail in Lee (1990b). Using the decision rationale represented in DRL, the computer can provide many services, such as managing the dependencies among claims, propagating and merging the plausibilities automatically, providing multiple viewpoints, and retrieving useful knowledge from past decisions (for more details, see Lee, 1990a).

4.2. Evaluation of DRL as a Design Rationale Language

The Argument Space

An argument is represented in DRL as a set of related Claims. A Claim subsumes what other people might call facts, assumptions, statements, or

rules. Instead of making these distinctions, which is sometimes arbitrary and difficult to make, a DRL **Claim** has the attribute, plausibility, which indicates how much confidence we have in the claim. This has the advantage of not imposing a set of predetermined categories on the user and avoiding the ambiguity resulting from the disagreement among people on what facts or assumptions are. When it is desirable to make the distinction, say, between facts and assumptions, we can do so simply or by specializing a claim or by using nominal categories like "fact" and "assumption" as values for the plausibility attribute in different **Claims**. We can do so after the fact or dynamically by using a numeric measure as the plausibility value and mapping between this measure and the measure based on the nominal categories such as factors or assumptions. Therefore, users do not have to conform to static categories prespecified by the designer of the vocabulary. We discuss different plausibility measures when discussing the evaluation space.

A **Claim** can be **Supported, Denied,** or **Presupposed** by another **Claim**. These relations among the **Claims** allow us to respond to a **Claim** directly without, as in IBIS, having to respond indirectly to the **Position** that responds to the second **Claim**. For example, the **Claim**, "Irrelevant commands can be dimmed," directly denies the **Claim**, "The commands not applicable to the current window would be shown and distracting," rather than having to be formulated as a **Claim** for the **Alternative** in question. These direct relations among the **Claims** allow us to see the logical and the dynamic structure of the argument more easily. All DRL relations are special types of **Claims**. For example, when we link a **Claim 1** to **Claim 2** through a **Supports** relation, we are making the claim that **Claim 1** supports **Claim 2**. Likewise, an **Achieves** relation from an **Alternative** object to a **Goal** object represents the claim that the alternative achieves the goal. Hence, any DRL relation, like **Supports, Denies, Achieves, Is A Subgoal Of**, is a **Claim** and can be argued about; that is, people can support, deny, or qualify them. For example, "Commands not applicable to the current window would be shown and distracting" is denied by "Irrelevant commands can be dimmed." That the first **Claim** is denied by the second itself is a relational **Claim**, which is then denied by "Dimmed commands are still distracting by their mere presence."

The Criteria Space

DRL represents the criteria space fairly well. In DRL, criteria are represented by **Goals**. DRL uses the term *Goal* rather than *Criterion* because, for each criterion, we can always define a corresponding goal (viz., the goal of achieving the criterion) and because we want to convey the richer relationship among these goals than what the term *criteria* usually conveys. For example, a **Goal Is a Subgoal of** another **Goal** if achieving the first **Goal** facilitates the achievement of the second. Subgoals can be related among themselves in various ways; they can be mutually exclusive, independent of

each other, or partially overlapping. These relationships are represented by creating a **Group** object and specifying these **Goals** to be its members; the relations among these **Goals** are specified in the "member relations" property of the **Group**.

Decision Problem represents the goal of choosing the best alternative. All the other goals for the decision problem are subgoals of the decision problem in the sense that they elaborate what it means to choose the best alternative. For example, the **Goal** "Easy to use" is a subgoal of the **Decision Problem** of our example if we interpret it to mean "Choose the alternative that has the property 'Easy to use.' " In other words, satisfying this goal facilitates the achievement of the goal of choosing the best alternative.[6]

Because the **Is a Subgoal of** relation is a **Claim**, as is any other DRL relation, we can argue about whether a goal is desirable or whether it contributes to achieving another goal by arguing about this relational claim. For example, we can argue about whether small implementation should be a goal at all. In Figure 5, there is an argument, "It should not be a goal because Mac is going to have large memory soon," denying that it is a subgoal of the decision problem; that is, small implementation is not a desirable property that should be used to compare alternatives. The record of these **Claims** and their relations represents the argument space for the goal space. Lee (1990a) discussed how this representation of **Goals** in DRL allows us to create multiple viewpoints and to extract from past decisions knowledge useful to the current decision.

The Alternative Space

DRL represents only parts of the alternative space well. DRL can represent alternatives and the specialization relation among them through the **Is a Kind of** relation. Thus, we can say that "Commands at the top of the window" is a special case of the alternative, "On each window." However, design alternatives may be related in much more complex ways than through the specialization relation. An **Alternative** can be related to another **Alternative** via, for example, the following relations: **Elaborates, Simplifies,** or **Is the Next Version of. Alternatives** can be related in a more complex way, for example, in the context of a design space. These relations are beyond the current expressive power of DRL.

DRL represents the arguments about the alternative space the same way it represents the arguments about the goal space. We can argue about whether

[6] The precise semantics of the model underlying DRL are more complicated and are fully explained in Lee (1991). Roughly, for a given decision problem of the form, "What is the best alternative for *X*?," its underlying interpretation is, "the goal of choosing the best alternative for *X*." Its subgoal of the form, *G*, is strictly speaking "the goal of choosing the alternative that satisfies *G*." An alternative of the form, *A* (e.g., the global menu bar at top of the screen), is to be interpreted as, "choosing the alternative, *A*." It is in this sense that a decision problem is the parent of the other goals and that an alternative achieves a goal. This nicety, although important for computational purposes, can be ignored by human users.

an alternative should be an alternative at all or whether an alternative is really a specialized version of another alternative by creating **Claims** that deny or support the appropriate relations, such as **Is a Good Alternative for** or **Is a Kind of**. The relations also can be qualified by linking them to another **Claim** via a **Presupposes** relation. For example, we can say that "commands on each window" is an alternative only if the window system allows the attachment of menu windows to the main window by linking the two claims with a **Presupposes** relation. One can of course object to this **Claim**, in turn, by pointing out another way of implementing the window commands at the top of the window.

The Evaluation Space

In DRL, each **Claim** has the following attributes: evaluation, plausibility, and degree. The evaluation attribute tells us how important the claim is, and its value is a function of both plausibility (how likely the claim is true) and degree (to what extent the claim is true). The overall evaluation of an alternative is represented as the evaluation attribute value of the relational claim, **Is a Good Alternative for**, between the **Alternative** and the **Decision Problem**. This value represents the extent to which the alternative satisfies the overall goal. This value, in turn, is a function of the evaluations of the **Achieves** relations that link the **Alternative** to the subgoals of the **Decision Problem** (i.e., the extent to which the alternative satisfies the subgoals). It is also a function of how the subgoals interact to satisfy the parent goal, such as the extent to which tradeoffs and synergies exist among these goals.

DRL does not commit to a particular measure of evaluation. Users of DRL can use nominal categories, numeric measures, or whatever they devise for evaluation. However, such evaluation measures should come with the algorithm for propagating and merging them to produce evaluation measures at a higher level. For example, if we want to use probability as the measure of plausibility, then we should also know how the probability of the two **Claims**—"The alternative 'In the global menu bar . . .' achieves the goal 'Can easily access commands' " and "The alternative 'In the global menu bar . . .' achieves the goal 'Reduce screen clutter' "—combines over the subgoal relations to produce the probability of the **Claim**, "The alternative 'In the global menu bar . . .' achieves the goal 'Easy to use,' " given that we also know how these subgoals are related among themselves and to the parent goal. We might try to work out such an algorithm based on Bayes's theorem, for example.

However, as discussed in Section 2, the exact algorithm is important only to the extent that the user can trust the algorithm. That is, if the algorithm is based on many assumptions that the user feels are seriously violated, then the exactness of the algorithm does not contribute much. We might as well concentrate on how to support people for making these judgments. DRL

takes this philosophy and tries to help by modularizing and helping to make explicit the relationships that need to be considered for these judgments.

The Issue Space

In DRL, the unit of the issue space is a decision problem. A **Decision Problem** corresponds to an **Issue** of gIBIS and a **Question** in "Questions, Options, and Criteria" (QOC). A decision problem **Is a Subdecision of** another decision problem if a decision for the first requires a decision for the second. For example, deciding where to place the window commands might require deciding what the window layout algorithm is (e.g., tilting or overlapping). A decision problem **Is a kind of** another decision problem if the first decision problem is a special case of the second: For example, "Where to place the emacs window commands?" is a special case of "Where to place the window commands?" Of course, we can relate the decision problems through the generic relations such as **Is a kind of** and **Is a part of.** We are sure that there are many other possible relations. For example, the **Replaces** relation in IBIS seems important for describing the dynamic aspect of the issue space. DRL, however, is based on the philosophy that the vocabulary should be extended to tailor the task in hand and that it is better to provide a method for extending the vocabulary as needed rather than to provide constructs that may not be useful in general.

5. RELATION TO OTHER STUDIES

In Lee and Lai (1991), we described other existing representations for design rationales and assessed them by using the framework developed in this article. We also discussed the relation between these representations and DRL; here, we provide a brief summary.

First, there are several representations based on IBIS (Kunz & Rittel, 1970), whose goal is to represent designers' argumentation activities. The most well known among them is gIBIS (Conklin & Begeman, 1988; Conklin & Yakemovic, 1991 [this issue]). The units of the **Issue Space,** the **Alternative Space,** and the **Argument Space** are, respectively, **Issue, Position,** and **Argument.** gIBIS provides no constructs for representing the **Criteria Space.**[7] Because criteria are not explicit, we cannot argue about them; we cannot represent the reasons for having these criteria; and we

[7] When we say that a representation cannot express some information, we do not mean that people cannot infer that information from the representation. For example, if we keep a detailed enough record in natural language of what happened, or even a video recording of the whole design process, we can always retrieve the information that has been recorded by working hard enough. When we say that a representation cannot express some information, we mean that the representation does not provide constructs that make the information explicit in such a way that help people easily see the structure or make it amenable to computational manipulation.

cannot indicate any relationship, such as mutual exclusiveness, among the criteria. Further, when criteria change, there is no easy way to accommodate the changes. It would be more difficult to isolate the real disagreements among people, because the criteria they use in their arguments remain implicit. The explicit representation of goals can also provide modular representations of arguments, multiple viewpoints, and can serve as a basis for relevance matching (cf. Lee, 1990a).

gIBIS's constructs for describing the argument space is also limited in several ways, as we pointed out in Lee and Lai (1991). For example, you cannot qualify an argument. Furthermore, because relations are not claims, as in DRL, there is no way of saying that we agree with *A* and *B* but not that *A* **Supports** *B*. The ability to argue about relational claims is important. For example, one may agree that the global menu bar is a bad idea and that seeing irrelevant commands is distracting but not that the second claim supports the first; for instance, the global menu bar does not have to show the irrelevant commands. Also, arguments cannot directly respond to other arguments. Therefore, a query such as "Show me all the arguments that respond to this argument" cannot be computed by the system.

The gIBIS structure has the advantage of being simple at least from the representation standpoint. However, we believe that the foremost criteria for a representation is not whether it is simple, but whether it helps users accomplish their tasks. This capacity to help is in turn determined by the trifactors of human usability, machine usability, and expressiveness. As Figure 1 indicates, there is a certain directionality among these three factors. An expressive language can be made easy to use with an appropriate user interface, but it is impossible to make a usable language more expressive. Therefore, it seems that a good starting point is to design an expressive language rather than one that is simple to use. DRL can be viewed as extending gIBIS in several ways: an explicit representation of the criteria space, a richer representation of the argument space, and the provision of an infrastructure for defining evaluation measures.

Procedural Hierarchy of Issues (PHI; McCall, 1987) overcomes some of the gIBIS limitations by allowing a quasi-hierarchical structure among issues, answers, and arguments. The semantics of the hierarchical relation are different for the different spaces. In the issue space, if **Issue A** is a child of **Issue B**, that means *A* "serves" *B*—That is, resolving *A* helps resolving *B*. In the answer space (i.e., the alternative space), an **Answer** is a child of another if the first is a more specific version of the second. In the argument space, an **Argument** is a child of another if the first is a response to the second. Hence, unlike in gIBIS, we can respond to an **Argument** directly by making it a child node of the **Argument**. Furthermore, PHI is *quasi*-hierarchical and allows the sharing of nodes (i.e., multiple parents) and cyclic structures. Although the quasi-hierarchy increases the expressiveness of PHI, some important relations

cannot be easily expressed in this structure. An **Issue** cannot specialize another **Issue**, and **Answer** cannot serve another **Answer**, and so forth. Therefore, the same comments about not explicitly representing relations in gIBIS apply to PHI. DRL can be viewed as pushing further the extensions that PHI made to IBIS by generalizing the hierarchical structure to more complex relations and by making explicit some other elements, especially those in the criteria space.

Potts and Bruns (1988) extended IBIS to represent the derivation history of an artifact design. One starts with an abstract **Artifact** (e.g., a plan for a formatter), associates with it the **Issues** that arise in making the plan more concrete, associates with each of the **Issues** the **Alternatives** considered (some of which lead to a more concrete plan), and so on until the plan is concrete enough to be implemented. Associated with each **Alternative** may be a **Justification**, which is the unit of the **argument space.** This representation is interesting because it allows us to describe yet another space that we might call the **Artifact Space**, where the evolving versions of an artifact over time are related. An extension of DRL that incorporates this additional space was presented in Lee (1991).

JANUS (Fischer et al., 1989) is interesting as an attempt to bridge two representations. One of its components, CRACK, uses a rule-based language for representing domain-specific knowledge (e.g., about kitchen design). The other component, ViewPoints, uses PHI to represent the rationale for the decisions made. JANUS integrates the two representations by finding the appropriate rationales represented in PHI for the particular issue that designers face in the construction phase, that is, while using CRACK. Although the current interface is limited to that of locating the relevant parts of the representations, bridging a design rationale representation and a domain representation is a very important topic of research because such a bridge can allow us to represent the relations among the alternatives or the criteria in more domain-specific ways.

QOC is a representation proposed by MacLean et al. (1991 [this issue]) whose constructs map clearly to the framework proposed in this article. **Question**, **Option**, and **Criterion** are, respectively, the units of **the issue space**, **the alternative space**, and **the criteria space.** A **Criterion** (e.g., "reduce screen clutter") is said to be a "bridging criterion" if it is a more specific one that derives its justification from a more general one (e.g., "easy to use"). The units of the **evaluation space** are links labeled with "+" and "−," corresponding to whether an option does or does not achieve a given criterion. Some constructs for representing the argument space are **Data**, **Theory**, and **Ad Hoc Theory**. One supports the evaluation ("+" or "−") of an **Option** with respect to a **Criterion** by appealing to empirical **Data** (e.g., "The mouse is a Fitts's law device") or to an accepted **Theory** (e.g., "Fitts's law"). When there is neither relevant data at hand nor existing theory to draw on,

the designers may have to construct an **Ad Hoc Theory**, which is an approximate explanation of part of the domain. MacLean et al. provide an illuminating discussion of the other forms of justifications for design, such as various forms of dependencies and metaphors, although no specific constructs are discussed for representing them.

QOC as we understand it has a number of limitations as a representation language, as we pointed out in Lee and Lai (1991). First, in the argument space, constructs like **Data**, **Theory**, or **Ad Hoc Theory** do not seem to capture many aspects of arguments. For example, it is not clear how an argument such as "Irrelevant commands can be dimmed" should be treated, given that it is neither a piece of empirical data nor a theory. And it is not clear whether and how we can argue about theories or individual claims in theories. In the alternative space, there is a reference to cross-option dependency, but no specific constructs are discussed for representing it. Given the ambiguity about what exactly constitutes the vocabulary of QOC, however, QOC seems to be more of a model than a fully developed representation language.[8] That is, it seems to be an attempt to understand and categorize the elements of design rationale without providing a specific vocabulary for expressing them. This observation is also consistent with the authors' warning against premature commitment to a specific representation (MacLean et al., 1989). Considering the similarity between QOC and DRL in the underlying structure, we hope that DRL provides a representation language adequate for representing most of the elements that the QOC research has been articulating.

6. CONCLUSIONS

A large body of research in the last two decades or so points to the importance of choosing the right representation for a given task (Amarel, 1968; Bobrow, 1975; Brachman & Levesque, 1985; Lenat & Brown, 1984; Winston, 1984). The task of using and reusing design rationales is no different. The benefits we can get and how easily we can get them depends heavily on the representation we use. The choice of representation is especially important when a human is the user of the representation, as in design rationale capture, because a wrong representation can turn people away from the task altogether, attributing the failure and frustration to the task itself rather than the inadequacy of the representation used. People might conclude that capturing design rationales is not worth the trouble because it is so hard and because it does not provide enough rewards for the efforts. But the real problem might be that the representation does not allow us to represent easily what we want to represent in a way that can provide much

[8] See footnote 1 about the distinction between *model* and *representation*.

benefit. Thus, it is important that we know how to evaluate a representation for a given task, in our case, for capturing design rationales.

In this article, we made a step forward by characterizing the domain of design rationale, that is, by identifying the kinds of elements that form the rationale as well as the relations that hold among them. Characterizing this domain is important because we then know what we can represent, what we have decided not to represent, and what the consequences will be. It also helps us to map the different meanings of design rationale by associating them with the different parts or aggregates of the domain. In other words, it provides a framework for defining the scope and assessing the expressive adequacy of a representation. Using the framework, we defined the existing representations' scope and discussed their adequacy. We have also presented a language, called DRL, which we believe is more expressive than most of the existing languages and overcomes many of their limitations in a way that is still natural to human users. However, that is a testable claim that we plan to investigate empirically by using DRL with many tasks by many users. We also discussed the limitations of DRL, which we hope will be explored by us and others in future research.

The step we made, however, is a small one, and we have a long way to go before we fully understand the important issues in designing an ideal representation for representing design rationales. We provided a framework for evaluating a design rationale representation along one dimension—its expressive power. Even then, expressiveness involves more than being able to represent the elements in the domain explicitly or not. There are many other characteristics, such as the ability to provide abstractions, that are important (see Bobrow, 1975). We need to think about whether these characteristics matter much for the task we have in hand and in what way they matter. Then, there are other dimensions to a representation than its expressiveness. In Section 2, we mentioned two other categories: human usability and computational tractability. We need to articulate the characteristics that make a representation more usable. We also need to identify the computational services we can provide with design rationales so that we know what they require and any tradeoff between their requirements and other requirements such as human usability. In addition to the articles in this issue, there are some existing studies that address these problems (Lee, 1989; Newman & Marshall, 1990; Shum, 1991; Yakemovic & Conklin, 1990). We need more such studies, more focus on the representation being used, and more systematic categorization of the results. We believe that the benefits from explicit representation of design rationales would more than pay for the efforts that we put into such studies.

Acknowledgments. The project of articulating the elements of design rationale grew out of our frustration with not knowing exactly how to evaluate the existing

representations and how to relate them to DRL. Tom Malone suggested that we define the scope of a representation by thinking about which components of a decision matrix it makes explicit. That insight triggered much of the analysis here. Jintae Lee thanks Frank Halasz for providing him with the great environment in which he wrote this article. The comments from the anonymous reviewers, Jack Carroll, Tom Moran, Jeff Conklin, Randy Trigg, and Austin Henderson were valuable. The article was influenced much by the comments from the members of the Argumentation Reading Group at Xerox PARC: Danny Bobrow, Frank Halasz, Bill Janssen, Cathy Marshall, Susan Newman, Dan Russell, Russ Rogers, Mark Stefik, and Norbert Streitz. We also thank the members of the learning group at the MIT AI Lab, especially Patrick Winston, Rick Lathrop, and Gary Borchardt.

Support. This work was supported, in part, by Digital Equipment Corporation, the National Science Foundation (Grant Nos. IRI-8805798 and IRI-8903034), and DARPA (Contract No. N00014-85-K-0124).

REFERENCES

Amarel, S. (1968). On the representation of problems of reasoning about actions. In B. Webber & N. Nilsson (Eds.), *Readings in artificial intelligence* (pp. 2-22). Palo Alto, CA: Tioga.

Bobrow, D. (1975). Dimensions of representation. In D. Bobrow & A. Collins (Eds.), *Representation and understanding: Studies in cognitive science* (pp. 1-34). San Francisco: Academic.

Bobrow, D., & Goldstein, I. (1980). Representing design alternatives. In I. Goldstein & D. Bobrow (Eds.), *An experimental description-based programming environment: Four reports* (Tech. Rep. No. CSL-81-3, pp. 19-29). Palo Alto, CA: Xerox Palo Alto Research Center.

Brachman, R., & Levesque, H. (Eds.). (1985). *Readings in knowledge representation.* Los Altos, CA: Morgan Kaufmann.

Card, S., Mackinlay, J. D., & Robertson, G. G. (1990). The design space of input devices. *Proceedings of the CHI '90 Conference on Human Factors in Computing Systems,* 117-124. New York: ACM.

Carroll, J. M., & Kellogg, W. A. (1989). Artifact as theory-nexus: Hermeneutics meets theory-based design. *Proceedings of the CHI '89 Conference on Human Factors in Computing Systems,* 7-14. New York: ACM.

Carroll, J., & Rosson, M. B. (1990). Human computer interaction scenarios as a design representation. *Proceedings of the Hawaii International Conference on System Sciences,* 555-561. Los Alamitos, CA: IEEE Computer Society.

Carroll, J. M., & Rosson, M. B. (1991). Deliberated evolution: Stalking the View Matcher in design space. *Human-Computer Interaction, 6,* 281-318. [Included in this Special Issue.]

Conklin, J., & Begeman, M. L. (1988). gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems, 6,* 303-331.

Conklin, E. J., & Yakemovic, KC B. (1991). A process-oriented approach to design rationale. *Human-Computer Interaction, 6,* 357-391. [Included in this Special Issue.]

Fischer, G., Lemke, A. C., McCall, R., & Morch, A. I. (1991). Making argumentation serve design. *Human-Computer Interaction, 6,* 393-419. [Included in this Special Issue.]

Fischer, G., McCall, R., & Morch, A. I. (1989). Design environments for constructive and argumentative design. *Proceedings of the CHI '89 Conference on Human Factors in Computing Systems,* 269-276. New York: ACM.

Kunz, W., & Rittel, H. (1970). *Issues as elements of information systems* (Working Paper No. 131). Berkeley: University of California, Berkeley, Institute of Urban and Regional Development.

Lai, K.-Y., Malone, T., & Yu, K.-C. (1988). Object lens: A "spreadsheet" for cooperative work. *ACM Transactions on Office Information Systems, 6,* 332-353.

Lee, J. (1989). Task-embedded knowledge acquisition through a task-specific language. *Proceedings of IJCAI Workshop on Knowledge Acquisition.*

Lee, J. (1990a). SIBYL: A qualitative decision management system. In P. H. Winston & S. Shellard (Eds.), *Artificial intelligence at MIT: Expanding frontiers* (pp. 104-133). Cambridge, MA: MIT Press.

Lee, J. (1990b). SIBYL: A tool for managing group decision rationale. *Proceedings of the Conference on Computer-Supported Cooperative Work,* 77-92. New York: ACM.

Lee, J. (1991). Extending the Potts and Bruns model for recording design rationale. *Proceedings of the 13th International Conference on Software Engineering,* 114-125. New York: ACM.

Lee, J., & Lai, K.-Y. (1991). *A comparative analysis of design rationale representations* (CCS Tech. Rep. No. 121). Cambridge, MA: MIT, Center for Coordination Science.

Lenat, D. B., & Brown, J. S. (1984). Why AM and Eurisko appear to work. *Artificial Intelligence, 23,* 269-294.

Levesque, H. J., & Brachman, R. J. (1985). *A fundamental tradeoff in knowledge representation and reasoning* (rev. ed.).

Lewis, C., Rieman, J., & Bell, B. (1991). Problem-centered design for expressiveness and facility in a graphical programming system. *Human-Computer Interaction, 6,* 319-355. [Included in this Special Issue.]

Mackinlay, J., Card, S. K., & Robertson, G. G. (1990). A semantic analysis of the design space of input devices. *Human-Computer Interaction, 5,* 145-190.

MacLean, A., Young, R. M., Bellotti, V. M. E., & Moran, T. P. (1991). Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction, 6,* 201-250. [Included in this Special Issue.]

MacLean, A., Young, R. M., & Moran, T. P. (1989). Design rationale: The argument behind the artifact. *Proceedings of the CHI '89 Conference on Human Factors in Computing Systems,* 247-252. New York: ACM.

McCall, R. (1987). PHIBIS: Procedurally hierarchical issue-based information systems. *Proceedings of the Conference on Planning and Design in Architecture,* 17-22. Boston: American Society of Mechanical Engineers.

Mostow, J. (1985, Spring). Toward better models of the design process. *AI Magazine,* pp. 44-57.

Newman, S., & Marshall, C. (1990). *Pushing Toulmin too far: Learning from an argument representation scheme.* Palo Alto, CA: Xerox Palo Alto Research Center.

Potts, C., & Bruns, G. (1988). Recording the reasons for design decisions. *Proceedings of the 10th International Conference on Software Engineering,* 418-427. Washington, DC: IEEE Computer Society Press.

Shum, S. (1991). Cognitive dimensions of design rationale. In D. Diaper & N. V. Hammond (Eds.), *People and computers VI* (pp. 331-344). Cambridge, England: Cambridge University Press.

Toulmin, S. (1958). *The uses of argument.* Cambridge, England: Cambridge University Press.

Winston, P. (1984). *Artificial intelligence.* Reading, MA: Prentice-Hall.

Yakemovic, KC B., & Conklin, J. (1990). Report on a development project use of an issue-based information system. *Proceedings of the Conference on Computer Supported Cooperative Work,* 105–118. New York: ACM.

HCI Editorial Record. First manuscript received July 13, 1990. Revisions received February 18, 1991; May 8, 1991; and May 21, 1991. Accepted by John M. Carroll. — *Editor*
