RATIONALE MANAGEMENT IN PROJECT PLANNING AND CONTROL

by

ZAKIA RAHMOUNA ZERHOUNI
B.S. Computer Science, The George Washington University (1989)

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degrees of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
and
MASTER OF SCIENCE IN COMPUTER SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1994

Signature of Author ....
Department of Mechanical Engineering
January, 1994

Certified by...
Warren P. Seering
Professor, ME Department
Thesis Supervisor

Certified by...
Patrick H. Winston
Professor, EECS Department
Thesis Supervisor

Accepted by...
Ain A. Sonin
ME Departmental Committee
on Graduate Studies

Accepted by...
Frederic F. Morgenthaler
EECS Departmental Committee
on Graduate Studies

RATIONALE MANAGEMENT IN PROJECT PLANNING AND CONTROL

by

ZAKIA RAHMOUNA ZERHOUNI
B.S. Computer Science, The George Washington University (1989)

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degrees of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
and
MASTER OF SCIENCE IN COMPUTER SCIENCE

Abstract

The construction project domain is a meeting point of various professions, constraints, and tasks. Therefore, both the volume and the complexity of information flow throughout the construction process are accordingly increased.
This research focuses on the development of an information management system that makes it possible, through a set of computational services, to capture and manage dependency relationships in the project, hence facilitating information re-use in both concurrent and subsequent activities.
For this purpose, an existing decision rationale representation scheme is augmented with an event-based knowledge representation. The resulting system provides a forum for efficient storage of and easy access to informal and unstructured information generated throughout the construction project.

Thesis Supervisor..Warren P. Seering
Professor, Mechanical Engineering Department

Thesis Supervisor..Patrick H. Winston
Director, Artificial Intelligence Laboratory

ACKNOWLEDGMENTS

First and foremost, I am grateful to the Ministry of Education of Algeria for giving me the opportunity to further my formal education. The scholarship I was granted allowed me to fully concentrate on my studies throughout both my undergraduate and graduate years.

Likewise, I would like to thank my thesis advisors for their support. Both Pr. Patrick H. Winston and Pr. Warren P. Seering provided guidance, advice, and patience all along.

I would also like to acknowledge the help I had from colleagues at the Artificial Intelligence Laboratory, and more particularly within Pr. Winston' s Learning Group. More specifically, my thanks go to Lukas Ruecker for his friendly advice and his insightful remarks and discussions throughout the development of this work, and Lisa Freedman for her extensive help during the editing stages of the thesis. Leslie Regan from the Mechanical Engineering Department Graduate Office always made administrative procedures smoother and easier than otherwise.

I also deeply appreciate the support and encouragement I received from all my friends. It is my hope that the friendships I have developed throughout these years will be pleasant and lasting. In particular, I thank Zahra Hayat Tazir for her continuous advice on the organization of the thesis, as well as Karima Benatchba, Lyna Brahimi, Samar Ayoubi, and Rabah Boudiaf for their unconditional support throughout this endeavor.

Last but not least, my eternal gratitude goes to my family. To my Father, Nourredine Y. Zerhouni, whose dedication for and commitment to a job well done have always been, and continue to be, an inspiration. To my Mother, Fatiha Boualga Zerhouni, whose constant smile has always been a sign of encouragement and approval. To my Brother, Amine Zerhouni, whose confidence and pride in my achievements is an effective remedy to moments of doubt. And to my Sister, Samira K. Zerhouni, whose invariable cheerfulness always manages to lighten heavier moments.

DEDICATION

To My Family, My Friends, and My Country

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

PREFACE

Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information upon it.
Samuel Johnson.

Throughout our lifetimes, we have been accumulating information. It seems that the more we learn, the more we realize our ignorance. And, in these modern times, what we learned, applied, processed, and analyzed yesterday, may be obsolete tomorrow. This excruciating need for efficient ways to manage this ever increasingly high volume of information resulted in the emergence of a new field that is often referred to as Information Technology.

More attention is given to the development of information management systems every day. Because of the breadth of the type of information that needs to be managed, various systems with different scope of use are studied to provide different services in different domains of application.

For example, our research is concerned with the conceptualization and design of a system that would assist managers keep track of the rationales behind the decisions they make throughout a construction project. Such a system not only maintains the project's schedule along with the resource allocations constraints, but it also provides its users with various tools that helps them manipulate the information captured in the system.

The rationale dimension of the system enables it to capture informal information generated throughout decision-making processes in a formal environment that is accessible to the system. Furthermore, we use an event-based representation to capture the knowledge acquired by the system during the deliberation process. In particular, we use this representation to capture and manage dependencies that occur in the construction domain.

In this thesis, we present the major concepts behind the construction project management system we propose. Chapter 1 describes our motivation behind studying the problem of project management. Chapter 2 explains our choice of the construction domain through illustrative scenarios. Chapter 3 presents the technology this work is based on. Chapter 4 describes the extensions we made to this existing technology to fit this system. Chapter 5 represents local dependencies management in the system. Chapter 6 presents a list of the computational services available. Finally, chapter 7 discusses of the overall performance, limitations, and contributions of our work, as well as a tentative direction for future research.

CHAPTER 1

INTRODUCTION

This chapter presents the motivation behind our research efforts in information management for project monitoring and control. Before concentrating on the particular problem of dependency management, we first discuss the importance of decision rationale management. We then narrow down the discussion to the specifics of dependency management in the construction domain, as this environment provides a good wealth of case studies for the issues that we need to address. At that point, we present our framework for a project management system, along with the underlying assumptions and insights. We conclude the chapter with a presentation of the material presented in the subsequent chapters of the thesis.

1.1. Decision Rationale Management

1.1.1. Benefits of Rationale Management

Decision rationales refer to the deliberation process decision-makers must go through when making decisions. A structured representation of such deliberations allows better use of previous decisions in the current deliberation process. It also provides a good forum of communication between decision-makers and knowledge holders. In addition, it is a good basis for documentation. Furthermore, a good decision rationale representation facilitates the provision of other capabilities, such as maintaining dependency relations among decisions, and handling changing events. Moreover, the value of these capabilities increases tremendously as the decisions' environment becomes more complex.

1.1.2. Problems of Rationale Management

The main concern in information management in general, and in rationale management in particular, is to get *the right amount of information, at the right level of detail,* and *at the right time*. However, we usually don't know where to look for that information in the first place. Most times, we don't even know who or what originated that piece of information. Even when these answers are available, they may be irrelevant because of how they are presented. Finally, more often than not, we obtain the requested information when it is no longer needed, or when it has already become obsolete. Such a delay could have several causes. For instance, the search operation itself may be slow, and/or the retrieval request may involve too long a chain of such search operations. We will therefore rate a decision rationale representation based on its capability to alleviate these difficulties, and on the efficiency with which it does so.

1.1.3. Dimensions of Rationale Management

Depending on the context in which they are introduced, decision rationales are handled differently. In the following, we classify rationale management into four categories. For each category, we present the main issues a good deliberation representation should be able to capture.

1.1.3.1 Managing Rationales within a Session

Managing rationales within a single session consists of relating together issues brought up within one single deliberation session, and keeping track of the dependency relationships. Such issues may, for instance, refer to the relationship between two decision problems, the effect of a new goal on the evaluation of the different alternatives, or the description of the same problem under different circumstances.

1.1.3.2 Managing Rationales across Sessions

Managing rationales across sessions deals with bookkeeping issues that are not necessarily inherent to rationale management. Such problems may consist of transferring the status of the decision-making process across sessions, remembering the previous argumentation, scheduling meetings convenient to all participants, as well as maintaining up-to-date information at all times.

1.1.3.3 Sharing Rationales across Decisions

Sharing rationales across decisions becomes especially important as the domain of deliberation becomes more complex. The main concern is to maintain global consistency across decisions. Indeed, in a deliberation process, one cannot make decisions independently of one another, because they may be affected by some, and/or affect others. Similarly, different people that work on similar problems may have different perspectives and requirements that may result in inconsistencies.

1.1.3.4 Reusing Rationales from Past Decisions

Finally, rationales from past decisions can often be reused in current decision problems. In most discussion domains, issues to be resolved have probably been previously discussed. In some cases, they may have been already resolved. Elements from both the argumentation process and the deliberation outcome itself can be mapped onto the current decision problem. Furthermore, rationales from past decisions may also be used for justification and documentation.

1.1.4. Desirable Services of Rationale Management

In order for decision rationale management to be performed properly, some services should be available to the decision-makers, so that they can consider the issues previously presented. *Precedent Management* allows participants to efficiently access, and therefore reuse previously acquired knowledge from similar, or closely related deliberation processes. *Viewpoint Management* allows participants to develop different scenarios of the same problem. It therefore enables a better analysis and evaluation of the various alternatives. *Dependency Management* is of critical importance to rationale systems, as it provides the ability to efficiently respond to a changing world, and to maintain an overall consistency among different, yet related, decisions. *Evaluation Management* is the ability of a rationale management system to assist decision-makers evaluate the alternatives brought up throughout the deliberation process, and to effectively propagate the decisions they make. *Communication Management* is of particular relevance when the parties involved in related deliberation processes are numerous and scattered, and when consistent and rapid propagation of changes throughout the deliberation process is consequently primordial. Finally, *Documentation Management* is essential for capturing and storing acquired knowledge. Such a feature allows for both the reuse of information, and the justification and review of decisions.

1.2. Dependency Management

1.2.1. Definition of Scope

As previously stated, dependency management consists of the capability a rationale management system provides to its users, in order to efficiently respond to a changing world, and to avoid making inconsistent decisions. In this work, we will use the term *dependency management* in its general sense, i.e., referring to the task of both capturing relationships and constraints existing between various elements of a deliberation process, and propagating changes that occur to the various elements through these relationships and constraints. For instance, the argument A1 that argument A2 motivates a decision problem D1 is an illustration of the first case, whereas the argument A3 that if A2 is no longer valid then D1 may not be relevant anymore illustrates the second situation. Just as we consider A1 a relationship between elements A2 and D1, we could also interpret A3 as a relationship between A2 and D1. This similarity of interpretation greatly simplifies the tasks of capturing and managing dependencies, as we will see in subsequent chapters.

1.2.2. Motivation for Scope Selection

We have decided to concentrate our research mostly on dependency management. However, rather than discussing dependency management independently, we chose to approach the problem in the context of decision rationale management in project monitoring and control. The reason for our choice is twofold. First, stand-alone dependency management systems are not very practical. In fact, we view dependency management tasks as a complement to other services that should be provided by more general rationale management systems for construction projects. Since our ultimate goal is to produce such a system, we decided to conceive dependency management in that context. Second, we strongly believe that dependency management is indeed a basic element of such other services. Indeed, it enables the accurate transfer of previous experience to current decision problems, the representation of constraints in order to answer what-if questions, the propagation of the effects of decisions made during a deliberation process, the update of the system, after the different parties involved contribute to the discussion, and finally, the indexing of the information kept for justification and review purposes.

1.3. The construction Project Domain

1.3.1. Features of the Construction Domain

Today's construction industry has become complex for multiple reasons. High competition among construction companies results in shorter projects. Also, people from different backgrounds come together and interact throughout the project's lifetime. Furthermore, as new technology develops, construction managers are faced with more choices that are more specific and more detailed than ever before. Finally, new construction techniques such as subcontracting, concurrent engineering, and fast-tracking are emerging in the industry. Subcontracting consists of allocating specific tasks to different consultants, or subcontractors, that specialize in a particular field. For example, in a single construction project, different subcontractors may be hired to deal with the plumbing work and the masonry work. Concurrent engineering is the technique of having several teams work simultaneously on related tasks of a project. Hence, while architects are designing the building, structural engineers study the soil properties of the building site in order to determine the appropriate construction materials to use. Fast-tracking is the technique according to which activities are started before all their requirements are provided. In order to shorten the overall project's lifetime, the construction activity is sometimes started before all the drawings are completed.

1.3.2. Motivation for Construction Project Management Systems

The inherent complexity of the construction industry suggests a strong need for appropriate construction management systems. The fragmented character of the industry affects the project development from various perspectives. As more teams interact with one another, different objectives and evaluation criteria need to be included in the decision-making process. Many meetings need to be scheduled to accommodate all the participants. Communication and documentation become increasingly complex, since people use different terminology, and different levels of details. Also, dependency management is harder, as information now needs to be updated frequently. Furthermore, in construction projects, time is often the most expensive resource. A higher volume of both the constraints and the tasks makes scheduling even more complex, and the new construction techniques tend to exacerbate these problems, as they reduce the lag time between tasks. Therefore, the overall complexity of the construction environment provides a good forum for the study of dependency management issues. Construction projects can host different types of dependencies to be considered. We also felt that they would provide a good test bench for our dependency management system, because of the important impact dependency management has on the overall performance of the project.

1.3.3. Dependencies in the Construction Domain

In the construction project domain, dependency relationships associate elements of construction project management across different dimensions.

Dependencies across Professions are those dependencies that relate the various parties involved in the project. For example, plumbing and electrical installations usually compete for space. The involved teams therefore need to negotiate and deliberate in order to come up with a configuration that is feasible and satisfactory to all. Also, electricians cannot start working on the interior installations before the masonry team has finished pouring the slab. Another constraint concerns the different objectives different teams have. When choosing the location of the main entrance door to the building, for instance, interior designers will be concerned with organizational issues such as accessibility and space efficiency, while civil engineers will be more concerned with structural issues such as safety and support.

Dependencies across Stages are dependencies that come about along the sequential dimension of the project. They relate decisions and arguments from different stages of the construction process. Decisions made at an early stage can affect deliberations that occur at later stages, but also changes made at the monitoring stage can result in changes in earlier stages. In deciding on where to set up the site offices, for example, the construction manager must keep in mind operational issues, such as accessibility to the building. Also, if the foundation sheets the engineers suggested are not strong enough to resist the ground load, another kind must be selected, ordered, and provided.

Finally, *Dependencies across Scope* are dependencies that occur along different dimensions of the information generated and used during the project. Schedule Dependencies relate schedule objects together. They affect only scheduling attributes of a project task, such as its duration time and the resources it requires. Rationale Dependencies relate rationale objects together along the deliberation dimension alone. They are represented in terms of objectives, alternatives, their evaluations, and other rationale entities. They too, are independent of the semantics of the issues and alternatives being discussed. Finally, Semantics Dependencies are those dependencies that are inherent to the definition of the objects being discussed. They are invariant across both the scheduling and the rationale dimensions.

In this research, we consider dependencies across scope, as this distinction directly maps the distinctions between representations. Scheduling objects, rationale objects, and semantics objects represent different entities and therefore have different structures. Dependencies across scope describe dependencies that arise within the different types of representations, i.e., scheduling, rationale, and semantics representations. Therefore, each type of dependencies across scope, i.e., scheduling, rationale, and semantics dependencies, consists of dependency relationships that relate objects of the corresponding dimensions together.

1.4. Desirable Services of Project Management

In previous sections, we introduced the motivation behind decision rationale management. In the following, we describe how the construction industry can benefit from decision rationale systems. For that purpose, we review the desirable services of rationale management systems, and illustrate how they may improve construction project management.

1.4.1. Precedent Management

When a new subcontractor is brought into the project, he/she can easily follow the deliberation process that had previously taken place to understand past decisions of the previous contractor. He/she can then decide, for example, whether to get the same tools from a cheaper supplier, or whether to order different tools altogether. Similarly, when an inexperienced engineer joins a construction company, he/she can access the previous knowledge and experience of other engineers by accessing previous projects. He/she can then decide which information is still relevant to the current situation, adapt it to this situation, and integrate it into the current deliberation process.

1.4.2. Viewpoint management

As the construction project gets closer to its end, the construction manager has more alternatives to choose from, and these alternatives become more detailed. The level of detail at that point is relevant only to the team dealing directly with that particular task. However, such a task is now more sensitive to external changes. Viewpoint management allows the users to view a deliberation process in different contexts. They can therefore analyze different alternatives independently, and then compare them with one another. At that point, argumentation elements may be transferred across viewpoints, and hence across alternatives as well. For example, the same argument may be used in different viewpoints to support different alternatives.

1.4.3. Dependency Management

The ability of the project management system to respond to changing external events is of crucial importance. For example, when the excavation crew hits an underground water pool, the structural engineers must redesign the building's foundation system in a relatively short period of time. These changes have to be adequately propagated, and other teams have to be informed promptly so they can adjust to the changes. If the decision rationales are available in the system, all parties that might be affected by the changes can be notified accordingly, and they can have access to all the information they need to adjust to this event.

1.4.4. Evaluation Management

This capability helps users evaluate the alternatives brought up throughout a decision-making process, and effectively propagate the decisions they make. For example, if one party feels that a particular objective has been overrated by others, at the expense of a more important objective, he/she can propose a change of evaluation in the first one, by presenting its arguments to the other participants. After proper deliberation and negotiations, an agreement is reached concerning the relative importance of the overall goals. The dependency management is then used to properly update the alternatives affected by that change. The deliberations regarding the different alternatives suggested to resolve the corresponding decision problem may now have different outcomes that need to be propagated across decisions.

1.4.5. Communication Management

Project management systems become more important when modern construction techniques are used to reduce the overall project's lifetime. It therefore becomes even more critical, as well as more difficult, for the different teams to maintain a constant flow of communication. When the architects make a last minute change to the interior design of the building, the masonry team must be notified immediately, before they have a chance to commit to the old design, and thus be forced to redo the job later on. Communication management allows the different teams to interact promptly and exchange their objectives, choices, and perspectives. It also reduces tremendously the need for meetings that are often difficult to schedule, and not always productive.

1.4.6. Documentation management

Documentation management relieves the construction manager from having to deal with rigid reports. Different parties can communicate with one another through the system, without resorting to formal exchange of reports and forms. Also, project review can be done directly from the system, since all the rationales behind the decisions made are maintained. This representation is much clearer and more expressive than the traditional sequential documentation, as it offers significantly better means of storage, retrieval, and management of relevant information.

1.4.7. Schedule Management

When planning and monitoring a project, the construction manager must constantly rely on schedules in order to be able to evaluate the status of the project. Schedules allow him to efficiently allocate resources, and to decide between competing alternatives, so that the project is completed within the allocated time frame. A project management system should therefore include a schedule management module. Such a module would translate decisions made in the deliberation process into scheduling objects such as the tasks to be performed, the resources required, the time needed, etc. The module would capture and manage scheduling dependencies, so that if a task is no longer part of the project, any other activity relying on its outcome can be adequately updated.

1.5. Summary

In this section, we briefly describe the scope, and then the content of this thesis. Our research consists of the conceptualization and design of a project management system that provides extensive dependency management support for decision making in the construction domain.

1.5.1. Scope of Research

The construction domain is a meeting point of different professions, constraints, and tasks. It is therefore characterized by the high volume and complexity of information flow throughout the construction project. This complexity generates dependency relationships that can be very difficult to keep track of.

Our research hence focuses on the conceptualization of a project management system that permits the representation and management of dependency relationships. Such a system would facilitate—through the set of computational managers it provides—"the re-use of acquired knowledge in on-going, concurrent, and subsequent activities."

For this purpose, an existing representation scheme for decision rationales has been extended with an event-based knowledge representation of the construction task, providing a natural way of propagating changes and retrieving context-sensitive precedents from semi-formal information.

Furthermore, the framework for a project management system described in the thesis was based on a symbiotic relationship between computer and manager. While the computer system is designed to handle heavy-computation tasks, the managers will decide what are the next steps to be carried through themselves. The system therefore operates essentially at an assistant level.

1.5.2. Thesis Outline

In this chapter, we described the motivation behind studying the problems of decision rationale management in general, and project management in particular. We have presented the various types of dependencies that might arise, as well as how adequate systems can relieve problems in project management. Chapter 2 explains our choice of the construction domain through illustrative scenarios. Chapter 3 presents the technology this work is based on. Namely, we present the frameworks we used for rationale management and knowledge representation in the system, as well as a previously developed scheme that combines both techniques to manage and document the design process of mathematical artifacts. Chapter 4 describes the extensions we made to this existing technology to fit our system. Chapter 5 represents local dependencies management in the system. More specifically, we show how dependencies among the different types of objects in the system, namely scheduling objects, rationale objects, and semantics objects, are captured. Chapter 6 presents of the computational services available. We describe how dependency management participates in each of these services. Finally, chapter 7 is a discussion of the overall performance, limitations, and contributions of our work, as well as a tentative direction for future research.

CHAPTER 2

A SAMPLE SCENARIO

In this chapter, we present sample questions a project management system should help answering. First, we present the different stages of a construction project cycle, and introduce a list of such questions. Then, we illustrate how the services introduced in chapter 1 help answering these questions. Finally, we introduce a project scenario, to describe how participants of a construction project might use these computational services in order to answer those questions. We should note, however, that the scenario is not an illustration of an implemented system, as we are only at the conceptualization stage. Rather, it is meant to reflect typical concerns that arise in a construction project, along with possible ways in which construction management systems can respond to these concerns. Moreover, we will use this scenario in a subsequent chapter as a measure of the capabilities of the suggested system.

2.1. Sample Questions in Project Management

One of the advantages of construction project management systems is that they provide means to relate decisions in the project across its different stages, through dependency relationships. They therefore help the user answer dependency related questions in different stages. In this section, we will describe a model of the construction project cycle. For each stage of this cycle, we will present sample questions that decision-makers usually ask themselves and try to answer.

2.1.1. Project Cycle

The construction project can be broken down into three major steps that require the involvement of all parties. Project participants must (1) identify the work to be done, (2) analyze different alternatives and plan appropriate strategies, and finally, (3) monitor and control the actual work, to guarantee task completion within the preset constraints. The cycle produced is an iterative process (see Figure 2.1). Indeed, strategies and plans are modified throughout the project as new conditions and constraints arise. Note, however, that the cycle is not along the time dimension and that the feedback loop is not to be interpreted as one global loop for the whole project. Rather, the cycle could be applied at different levels of the project. Indeed, while it could refer to the whole project, it is also possible to apply this model to individual decisions and tasks of the project. When a decision needs to be made, decision-makers must first define it adequately, along with the related constraints and objectives. Then, alternatives are suggested and discussed. Finally, when one strategy is selected, it is carried through and monitored. The cycle repeats again as new constraints are introduced. Since decisions and tasks can be broken down into subdecisions and subtasks, the lower the level of detail of deliberation and planning, the more loops are generated.

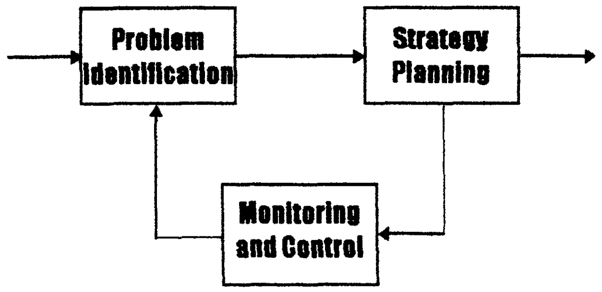


Figure 2.1. The Construction Project Cycle

2.1.2. Problem Identification

In this stage, decision-makers with different backgrounds come together to analyze and discuss the job requirements. The different people involved include the client, the architect, the consultant, the surveyor, the contractor, the subcontractor, the supplier, and the financier. The clients present their overall goals, which are then translated into project specifications. As the process goes on, descriptions of the decision problems and the goals become more and more specific and task oriented. Sample questions that may be raised at this point include:

What does the overall project consist of?

What are the general criteria used in comparing different strategies?

What is a rough cost estimate for the project?

What are the different components of the excavation task?

What is the estimated time required for project completion?

What is the allocated budget?

What are the resources available a priori?

2.1.3. Strategy Planning

Once the overall goals and constraints are all defined, the different parties must yet again come together and agree on the general procedure to follow, in order to achieve the general goals given the specified constraints. In selecting the optimal strategy, different alternatives are discussed, and different arguments are presented in the alternative evaluation process. During this stage, issues that are raised include scheduling, resource management, and cost efficiency. Again, as the project cycle repeats itself, these discussions and deliberations get more specific. In evaluating different alternatives, typical questions that come about include:

How much labor is required to do the masonry job?
What is needed to install the HVAC system?
How long does it take to complete the foundations?
When will the selected foundation sheets be available?
Which supplier offers a better product at a better cost?
What are the suppliers the company dealt with in previous projects?
How sensitive is the estimated schedule to a possible union strike?

2.1.4. Monitoring and Control

This stage corresponds to the actual construction project activity. The resource allocation and personnel coordination are performed on a closer, day-to-day basis. On-site and off-site decision-makers interact with one another. Also, the project schedule is appropriately updated as new conditions arise. This is the feedback step of the project cycle, in which corrective action is taken in response to unexpected events. Basically, at this stage, the project manager does his/her best to ensure that the project is completed within the agreed time and cost intervals. He/she must be able to both detect problems early and respond to them quickly. The longer the wait, the more effort, and consequently, the more resources are required to get the project back on track. The response time of the construction manager depends heavily on the ability of the project management system to maintain dependencies and propagate changes through the project, to provide an accurate and up-to-date documentation on the current status of the project, including all the relevant rationales, and finally to enable a clear and effective communication between the participants in the construction project. Typical questions asked at this stage include:

If the cement delivery is delayed, how will it affect the task of laying the slab?

What are the causes of the delay?

Are there other sources that will supply cement faster?

What other tasks will be affected by that delay?

Can the schedule be modified so as to minimize the overall project delay?

What are the characteristics of the foundations used?

If waterbeds are reached underground, are the chosen foundation sheets still adequate?

2.2. Computational Services

In this section, we review the desirable services a project management system should provide, as introduced in chapter 1. To illustrate how these services can be used in all three stages of the project cycle, we discuss them in the context of the questions listed in the previous section,.

2.2.1. Precedent Management

The ability to reuse previous experience is helpful in answering questions at each stage of the project cycle. For example, participants can examine previous, similar projects when breaking down a particular task. They can import general criteria that were used to rate similar strategies. Also, they can use their previous experience to estimate the total cost-time- and resource-requirements for the completion of the current project.

The construction company can also use its past experience with different suppliers and contractors when analyzing the availability and quality of equipment and materials. Previous use of certain materials and equipment also helps better predict potential problems and difficulties that may arise. Such additional information helps decision-makers make better choices when evaluating different alternatives.

Finally, precedent management plays an important role during project monitoring and control. Indeed, past experience helps identify dependency relationships, and it enables decision-makers to better predict the effects of unexpected events. It also provides more relevant information when deciding how to respond to these changes, in order to minimize delays.

2.2.2. Viewpoint Management

A system that facilitates the development and evaluation of different alternatives improves decision-making at all three stages of a construction project. During problem identification, it allows different interpretations of the project requirements to be considered. For instance, when the clients' requirements are not clear, various scenarios of the project with different assumptions can be maintained, and the goals can be defined separately for each case, until further clarification is provided. Similarly, given a set of overall requirements, different participants, like the masonry team, the HVAC team, and the architect team, can maintain their own criteria and requirements in a private "copy" of the project representation. During a discussion, the different issues are appropriately merged so as to avoid any inconsistencies.

Viewpoint management can also be used when evaluating alternatives. Indeed, each alternative can be represented separately, hence maintaining the related argumentation and the resulting deliberation independently from the other alternatives. At a later stage, the alternatives can then be compared with one another, and evaluated relative to one another, until a final strategy is selected.

Finally, viewpoint management can also be used to predict the effects of changes on the project through what-if question answering. A different viewpoint is created for every scenario—i.e., for every assumption—and changes are propagated independently in each one. The different effects can then be compared, and decision-makers are hence better prepared to respond to changes.

2.2.3. Dependency Management

Dependency management provides the ability to both respond to a changing world, and maintain overall consistency within the project. When estimating project completion time, for instance, the different tasks to be performed are related to one another according to precedence relationships, which can then be used to both efficiently allocate the resources available and accurately generate a project schedule.

Similarly, constraints and inconsistencies are relevant when examining strategies and evaluating alternatives. The availability of labor, for instance, affects the relative performance of potential strategies, depending on their reliance on labor. Hence, in the case where the strategies examined are not labor-intensive, the availability of labor may not be relevant to the current decision problem, and it will not affect it much. If the strategies are labor-intensive, their performance will greatly depend on that factor. If, however, they rely on labor to the same degree, their relative performances may remain the same. In the case of a strike, for example, two strategies will require more time, but they may still be rated the same way, relative to each other.

Finally, efficient dependency management and constraint propagation are of paramount importance for project control. The feedback loop in the project cycle relies heavily on early detection of problems, such as, for example, the delay of cement delivery due to unexpected changes. Conversely, dependency relationships can be traversed in the opposite direction, when explaining the causes of events, and justifying the overall delay of project completion.

2.2.4. Evaluation Management

The capability of a project management system to evaluate, and subsequently to choose alternatives is mostly used in the strategy planning phase of the project cycle, although it is also relevant to the other phases. Indeed, during the problem identification phase, the scope of the project is defined, along with the goals to achieve. For example, the defined goals and requirements may have different importance.

During the strategy planning phase, different alternatives may solve a problem to different extents, and the same alternative may solve different problems to different extents. Also, depending on the context of their use, different arguments may have different accuracy and/or relevance levels. The final decision-making, namely the selection of the best strategy for a particular task will therefore depend on both how the alternatives perform that task, and how accurately they perform it.

Finally, when changes occur, previous decisions and tasks are affected differently. Hence, when the clients add a new requirement, the alternatives may be rated differently. The rating of the strategies that better fulfill that requirement improves the most, whereas those that fulfill the requirement the least will have lower ratings. Also, the extent to which a change affects the project varies. Hence, if a critical task is postponed, the whole project may be delayed. On the other hand, if no other task depends on it, a task may be postponed without altering the rest of the project.

2.2.5. Communication Management

Communication management is of paramount relevance in a construction project because of the large number of participants. They all have different schedules, different perspectives, and different concerns. An efficient communication method can therefore improve the overall project tremendously. A lot of communication takes place during the project identification phase, when various parties discuss their different requirements and constraints.

The many participants must also communicate with one another to exchange their different arguments and to identify the tasks they have in common. For example, in the discussing the location of the main door of the building, the architect will be concerned with accessibility to the handicapped, whereas the structural engineer will base his/her decision on the safety of the resulting structure.

Finally, participants must remain in touch with one another throughout the project monitoring phase, so that the response time to unexpected events is minimized. If the foundation sheets that were ordered cannot be provided, the supplier must notify the construction manager, who must then propagate the information to other parties. The contractors, for example, must check if other suppliers can deliver the sheets in time, and the engineers must make sure the new types of sheets are still adequate.

2.2.6. Documentation Management

A good documentation technique is the basis for information management. It helps retrieve the right information at the right time, share it efficiently among the various project participants, reuse it in future projects, or use it to justify and review past decisions. For instance, in the first stage, all contractual agreements between the clients and contractors must be well documented. The contracts are then translated into distinct specifications for the different project participants. These new specifications must also be documented for legal purposes.

During the strategy planning phase, information concerning the different resources to be used is of paramount importance. In order to select the type and model of foundation sheets to use, structural engineers need detailed data about the strength capacity, water permeability of the material, and relevant information concerning the soil properties. Also, the different parties that evaluate alternatives must document their argumentation to reduce the risk of generating inconsistencies among the different teams.

Similarly, when changes occur during project monitoring, we must record all information concerning the reason for, the time of, and the originators of these changes, so that everyone concerned is notified accordingly. This information can be kept for future use, to predict, and hence avoid similar problems. If the foundation sheets used have some defect, e.g., if they are not as strong as expected, the participants on this project should document this fact, so that another is used in the future. Good documentation is also necessary to justify and explain delays that may occur during the project. When the contractor needs to ask for a considerable extension, the clients may require to see detailed justification of the delay, to decide whether to grant that extension, or whether to consult another contractor.

2.2.7. Scheduling Management

Construction projects evolve around scheduling tasks. Scheduling is the best measure available to the project participants to assess their progress. In the early stages of a construction project, contractors must present a tentative schedule when bidding for a project, representing both time and resource requirements. The client also uses the schedule to study the feasibility of the project, and to estimate the budget to allocate to it.

During strategy planning, scheduling issues must be taken into account. Low cost and short duration are general criteria used in evaluating different strategies. Furthermore, during the deliberation process, resource allocation parameters—such as what resource is available, how many instances of it, and for how long—must be considered too. For instance, a labor-intensive strategy might be cheaper than a more automated one. However, if most of the trained labor is allocated to another task, then the low cost advantage associated with the strategy is diminished because the task is completed later.

Finally, during project monitoring the project schedule is the major tool that reflects the effects of an unexpected event on the overall progress of the project. Also, project delays can sometimes be avoided, or at least minimized, by rearranging the schedule, and reordering the tasks yet to be performed.

2.3. Illustrative Scenario

In this section, we provide a scenario to illustrate how participants of a construction project would interact with a project management system that would provide services similar to the ones described in the previous section. We should stress, however, that the scenario is not meant to describe the syntax of a particular system, nor the format in which the information exchange should occur between user and system. Rather, it is used to illustrate what kind of information is transferred. Furthermore, the format of interaction we will use in this section is not necessarily the one we envision for the system described, as this decision has been postponed to a later stage in the system's development.

In the following, we present the rationale behind defining, organizing, and scheduling the task of *clearing the construction site*, which is one of the early tasks that take place during a construction project. Figure 2.2 represents the task at two levels of detail. Part (a) describes the precedence between the two subtasks *Topsoil Removal* and *Topsoil Disposal*, whereas part (b) shows the cyclic aspect of the topsoil disposal task.

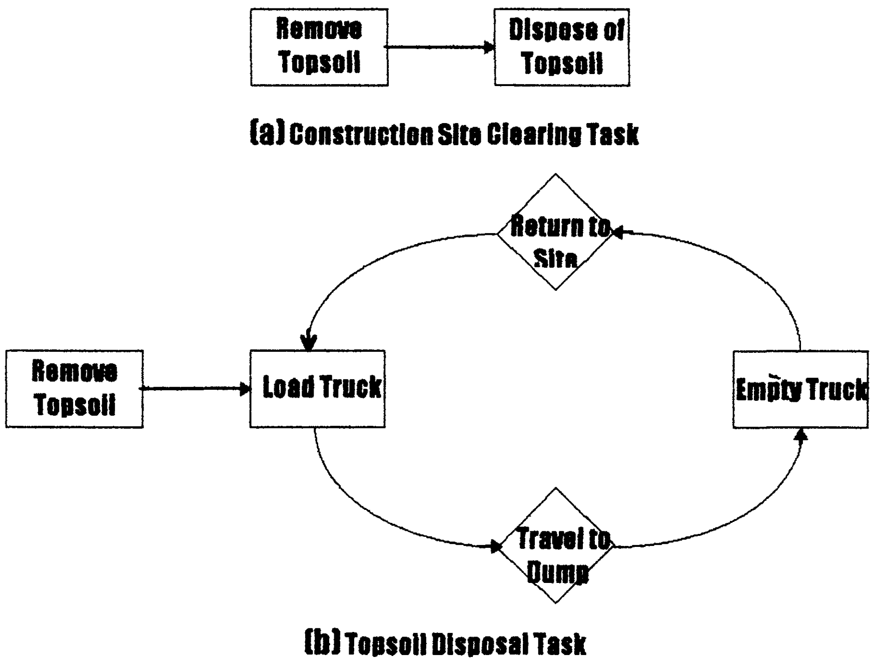


Figure 2.2. The Construction Site Clearing Task

This presentation relies on the classification of rationale management. We use <u>underline</u> format to represent a user's request, and *italic* format to represent the system's response. Comments and descriptions of the operations performed will be preceded by double semicolons.

2.3.1. Managing Rationales within a Session: Getting Started...

We first start by setting up the decision problem at hand. The user interacts with the system to clearly define the problem, based on both the user's input and the system's knowledge. The deliberation process is then initiated.

<u>We are concerned with the problem of clearing the construction site. What do you know about this task?</u>

I don't recognize this task. Would you like to define it more precisely?

;; A decision problem is initiated. The system searches for an existing representation of the problem of clearing the construction site in the knowledge base, in vain. An appropriate structure is then created to represent the newly defined problem.

<u>Clearing the construction site involves first removing the topsoil from the ground, and then disposing of it away from the site. The topsoil cannot be disposed of before it is removed from the ground.</u>

OK.

;; The problem is broken down into two subproblems. The subtasks are also related to each other in a precedence relationship, for scheduling purposes.

<u>What do you know about removing topsoil?</u>

Removing the topsoil can be thought of as a special case of moving object1 from location1 to location2 using tool1.

;; Again, the system searches through its knowledge base, looking for a definition of the problem at hand. This time, however, it comes up with a general description that can be adapted to this particular event. During this matching process, object1 of the description is bound to the topsoil object, location1 to that of the ground, and the other variables remain free. Such free variables are therefore undecided, and the users may discuss possible options for them.

<u>By disposing of the topsoil, we mean that a transportation vehicle V1 is used to carry the topsoil into location3, that is away from the site.</u>

OK.

;; The user then defines the other subproblem, and the system keeps track of this definition, at the same level of detail as the user stated it. During this interpretation process, the system also recognizes the different constraints that are implied.

<u>Tentatively, we schedule the topsoil disposal task to start at time T1. What are the available vehicles at that time?</u>

Two trucks of large capacity are available for the following three days. Three smaller ones are scheduled for other tasks starting the next day.

;; The system accesses the project schedule it maintains, and looks for the tasks that are scheduled at time T1, along with the resource allocation at that time. It then returns all the required resources that are available.

<u>Consider the two alternatives for disposing of the topsoil. Alt1 consists of using the five small trucks, Truck1, Truck2, Truck3, Truck4, and Truck5. Alt2 consists of using one large truck, Truck6. What are the problems that we might face?</u>

It may be difficult to access the site with the large truck, as it may be too wide to get across the site gate. Although the small trucks provide better maneuverability across the site, the task may take longer to complete, and the tasks scheduled to use these trucks on the next day may need to be rescheduled. Also, you must make sure that all five of them are available.

;; At this point, the problem of disposing of the topsoil is not defined with many details. The amount of the topsoil to be removed, for instance, is not known. Therefore, the system cannot make definite decisions. Considering the little knowledge it has at this point, it can only give guidelines and pointers to potential problems.

Suppose that maneuverability is not such an important concern anymore for the topsoil removal task, and the topsoil is thus stacked at the entrance of the site.

In this case, if Truck6 can get through the gate, then it may be a better solution, as it has bigger capacity, it requires less labor and fewer trips to the disposal site, and it is available for a longer time, so other tasks of the project are less likely to be affected.

;; At this point, the system goes through the suggestions it just made, and it eliminates those that are made undesirable by the new assumption.

All right. For now, we will commit to the second alternative. Please update the system accordingly, and end the session.

OK.

;; Finally, all the relevant changes are made in the system, by navigating through the different dependency and constraint relationships, as they are represented.

2.3.2. Managing rationales across Sessions: Catching Up...

In subsequent sessions, the deliberation process will carry on in a similar fashion, with constant interaction between user and system. In addition to argumentation issues that deal directly with the decision problem examined, bookkeeping issues are raised as well, as illustrated in the following examples.

What is the status of the problem of clearing the construction site?

The problem has been broken down into two subproblems that represent the tasks of removing the topsoil, and disposing of it. The first subproblem is still unresolved, while the second one has been resolved already.

;; The system searches through the rationale structures and identifies the appropriate representation of the construction site clearing problem. It also searches the knowledge base for a possible definition of the problem, which is subdivided into two parts. For each subproblem, the same process occurs, and when the rationale representation of the subproblem is accessed, the system checks whether the subproblem has been resolved.

Has anyone else brought in more arguments concerning this issue recently?

The project manager mentioned that she had been approached by a nearby truck rental company. She may be able to rent trucks for short period of times, and at lower prices.

;; When the rationale structure related to the construction site issue is identified, the system navigates through it and identifies any additions and/or changes made by other users.

Has there been any change in the evaluation of alternatives?

Not yet, but the financial team suggested that the possibility of using small trucks might become more feasible, and this alternative may need to be re-evaluated.

;; Once these changes have been identified, the system checks how these new arguments have affected previously made decisions.

What decision was made about the problem of disposing of the topsoil?

It has been decided that using one large truck was the best alternative to resolve that problem.

;; When the system recognizes a problem has been resolved, it can access the related structure to find out more about how it has been done.

On what basis was this decision made?

The following assumptions were made: The task of disposing of the topsoil was scheduled to start at time T1, and the topsoil was to be stacked near the site gate, when removed from the ground.

;; During the deliberation process, several assumptions were made by the users. These assumptions, and how they affect the overall process, were stored along with the decisions themselves, in independent copies, or viewpoints, of the problem. Therefore, the above question can be easily answered by simply identifying the viewpoint that is currently being considered.

<u>Show the argumentation related to the topsoil removal problem</u>

It has been agreed that the task of removing topsoil from the ground is a special case of moving object1 from location1 to location2, using tool1. Object1 has been bound to the topsoil object, and location1 to the ground object. Location2 has been constrained to be away from the construction site. Finally, no discussion took place concerning tool1.

;; When the system identifies a decision problem, the user may want to know more about the corresponding argumentation. All the information is stored in the rationale structure of the system. Once the structure is identified, it therefore becomes a simple task of accessing the right objects and the right fields.

<u>Make a note that the topsoil removal problem needs to be resolved immediately. There is a general meeting with the clients scheduled for the end of the week, and they requested to have a preliminary schedule of the project, along with the resource requirements and the cost estimates.</u>

OK.

;; The user can provide more information to the system, and leave messages to other users. This capability allows participants to get around, or at least to minimize, the difficult task of constantly requesting face-to-face meetings with one another.

2.3.3. Sharing Rationales across Decisions: Updating the Project...

When decisions are being made, other decisions may be affected. Also, other groups participating in the same project may feel that the arguments being brought up are relevant to their decision-making, too. As an illustration, we will assume that in previous sessions, the problem of removing the topsoil from the ground has been discussed, and that two main alternatives have been presented. The first one is a labor-intensive strategy according to which several workers are using manual shovels. The other one is equipment-intensive, such that one mechanical shovel is used. We also assume that wheelbarrows will be used to move the topsoil and stack it in both instances.

<u>How does the constraint from the topsoil disposal problem —i.e., the need to stack the removed topsoil near the site gate—affect the alternative evaluation of the topsoil removal task?</u>

Since both alternatives require the same means of removing the topsoil in order for it to be stacked, i.e., wheelbarrows, both alternatives will be affected the same way.

;; The system goes through the dependency relationships to find which aspect of the problem is affected by the new constraint, and then uses the rationale structure to analyze how the different alternatives are accessed through these relationships.

<u>What factor of this new constraint will affect the problem of removing topsoil from the ground?</u>

This constraint affects the topsoil removal problem indirectly through the location parameter. The new constraint may introduce contradictions with previously made decisions like the space allocation for the site offices, the storage areas, and so on.

;; In considering the dependency relationships, the system also goes through its knowledge base to access all available information about the new constraint. From this information, it then follows the dependency relationships to determine its effects.

Onc of thc construction unions involved in the project has mentioned the possibility of a strike around the time the site clearing task is scheduled. How sensitive to it is the topsoil removal task?

Although not all the workers belong to that union, there will nonetheless be a shortage of labor. Thus, the labor-intensive strategy will be more affected than the other one, and so the increase in task duration is more important for the first strategy. Therefore, if the probability of a strike becomes greater, it may be better to choose the second alternative, in order not to be delayed too much.

;; In a similar process, the system identifies the relevant dependency relationships connecting the various decision problems to one another. These dependencies are then translated into the scheduling domain, where tasks are represented, along with their order of precedence. The system can then identify and propagate the effect of a delay at some point in the schedule.

How can a longer time for the topsoil removal task affect the topsoil disposal task?

Initially, the starting time of the topsoil disposal task will only be delayed slightly, say by time t. Overall, however, the total delay will consist of an accumulation of the cyclic delays, t, so that the topsoil disposal task will also be slowed down.

;; Depending on the type of relationships between tasks represented in the scheduling domain, changes will be propagated differently. If task T2 is constrained to start after task T1 has started, for instance, then delaying T1, that is, causing it to take longer, will result in delaying only the starting time of T2. On the other hand, if the sequence (T1,T2) is repeated through a cycle as in this case (see 2.2), then the overall duration of T2 will also be affected (increase). Therefore, in the schedule domain, the system must be able to distinguish between the different types of task precedence relationships.

The equipment supplier has notified the contractor that he can provide fast mechanical shovels a day after the task of site clearing is scheduled to start. How can it affect the topsoil removal task?

This argument will mainly affect the evaluation of the equipment-intensive alternative. A question that remains to be answered though, deals with the one-day delay involved. If the machine is fast enough to compensate for that delay, then the performance of this alternative is improved. If, in addition to that, the risk of a strike becomes more probable, then there are even more chances that the one-day delay will indeed be compensated.

;; Here again, in order to answer this question, the system switches to and from the rationale structure, the knowledge base, and the schedule domain. In this case, the answer is undetermined because the system needs more information regarding the meaning of "early" and "fast," but even though there is no definite answer, the ability of the system to notify the user is already valuable.

How does a delay in the completion of the construction site clearing task affect the overall project?

The actual construction work cannot start before the site has been completely cleared. Also, if the task is delayed, then the resources required won't be available for other tasks. For example, if the cement is to be brought in the trucks right after the topsoil is disposed of, then the cement cannot be available as scheduled either.

;; Using the critical path method terminology, the effect of a task being delayed on the overall project depends on the magnitude of the delay, and also on whether the task is on the project's critical path or not. Furthermore, we should note, in this case, that although the delay in material availability is not so important because the construction work in itself is delayed as well, it may still involve more costs, as the supplier may charge for storage, for instance.

2.3.4. Reusing Rationales from Past Decisions: Enhancing the Project...

The construction domain is a relatively static, conventional environment. Technological discoveries are not very frequent. Construction projects are therefore very similar to one another. Also, the problems that arise during one project are typical, in that they have probably occurred in other projects. Furthermore, the fragmentation of the industry, and the tendency to subcontract most of the work, makes it very difficult for construction companies to maintain good records of their past projects, and for past experience to be transmitted from past employees to new ones. This makes the ability of project management systems to reuse past decisions both easier and more useful, as will be illustrated in what follows.

Has the company been previously involved with a construction project in a site with similar soil properties as this one?

Yes. The company has built an office complex in the same neighborhood, a decade ago.

;; The system uses its precedent management module to answer this question. Note that if the soil topology is used for indexing construction projects, then the matching process is simplified for this particular example.

How was the construction site clearing task carried out?

At that time, two small mechanical wheels were used to remove the topsoil from the ground, and seven trucks were used to dispose of part of the topsoil in a close-by landfill. The other part was used on-site. The task was completed within ten days.

;; Once a precedent decision problem is identified and accessed, questions similar to this one are answered in the way that has been described in the previous sections.

Why did it take so long?

The equipment was not as powerful and fast as the one available nowadays. Also, the site was much larger. Finally, it took time to get the authorization to dump the topsoil in the city landfill.

;; To answer, the system navigates through the dependency relations backwards, in order to explain the decisions that were made ten years ago. The causal relationships between events are found, and the appropriate rationales are accessed.

What were the main concerns of the construction company in evaluating the various strategies for this problem? Why?

The clients were cost oriented. They were going through financial difficulties, and so they insisted the project remained within budget. As a consequence, the project took a long time to complete, as the most cost-efficient solutions were not always available.

Make a note to obtain all the authorizations from the city that are required for this task to get started.

OK.

;; Both the documentation manager and the communication manager are used to store the note in the system, and then send it to the legal team of the project.

CHAPTER 3

UNDERLYING TECHNOLOGY

In this chapter, we summarize the technology used in our framework of a project management system for the construction domain. Therefore, we describe it within the project management context. Further details on the described frameworks can be found in the original works of the respective authors. We start by presenting SIBYL, the decision rationale management system that Jintae Lee developed to manage information produced during a deliberation process. We also introduce Transition Space, the knowledge representation that Gary Borchardt developed to capture and understand chains of events. Finally, we examine mSIBYL, the documentation system Lukas Ruecker developed based on both SIBYL and TS to manage and capture the design process of mechanical artifacts.

3.1. SIBYL: The Rationale Management System

As mentioned in chapter 1, a structured representation of decision rationales brings about many benefits. Such benefits include precedent retrieval, project documentation, and dependency management. Jintae Lee has developed a system that captures and manages rationales without relying on a formalized domain knowledge. Informal, thus inaccessible, descriptions of the knowledge are included with formal, thus accessible, structures of deliberation. This way, the system operates on the formal structures, rather than on the informal descriptions. This semi-formal aspect of the framework implies an intensive user interaction with the system. Indeed, the user has to interpret the deliberation role of the informal descriptions, and formalize them so that the system can adequately manage the rationale structures.

In the following, we will present the three components of the rationale management system Lee has developed. First, we introduce the decision rationale language, DRL, that was used to describe the elements of decision rationales. We then describe SIBYL, an environment in which DRL is used to capture the decision rationales. Finally, we present the most important computational services currently available in SIBYL in order to provide decision support.

3.1.1. The Decision Rationale Language

DRL provides a set of constructs and relations to represent and manipulate decision rationales. This representation takes place along five different spaces of deliberation that have been defined in DRL. In the following, we introduce the various spaces. We then describe the constructs of DRL. After that, we illustrate how these constructs map onto the five spaces.

3.1.1.1 DRL Spaces

The five spaces of DRL interact with one another to allow the system to (1) capture the rationales behind a decision, (2) compare different alternatives when making the decision, (3) evaluate these alternatives, (4) capture the criteria used in the evaluation process, and (5) associate this decision to other related ones, as shown in Figure 3.1.

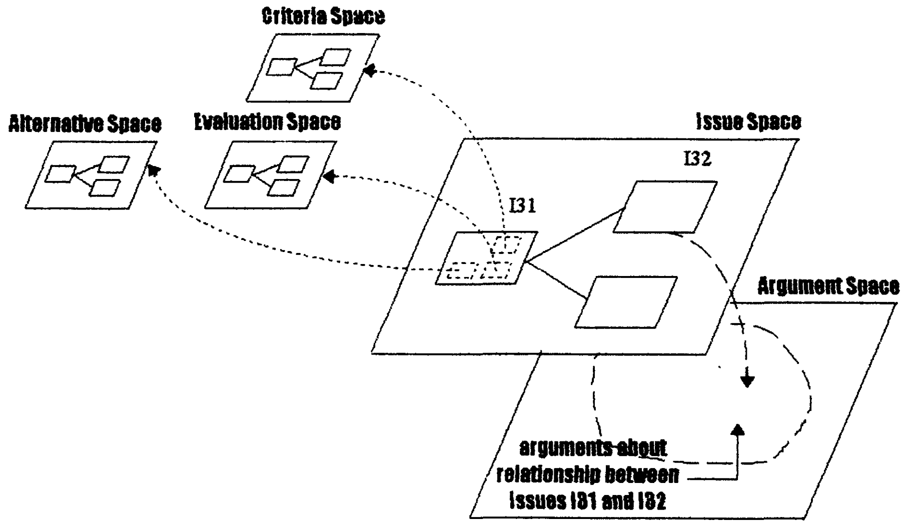


Figure 3.1. DRL Spaces

The *argument space* refers to the argumentation that takes place in a decision-making process. It is along this space that issues, alternatives, and even criteria are discussed and argued about, and it is this space that captures the rationale behind the outcome of a decision. The *alternative space* enables the user to explicitly formulate different alternatives for a given problem, describe their attributes, compare them with one another, and discuss their relevance to the problem at hand. The *evaluation space* makes the evaluation status of an alternative of the alternative space explicit, hence allowing the evaluation status to propagate along the alternative space. The *criteria space* provides an explicit representation of the criteria that need to be fulfilled by different alternatives, along with their respective importance. This space is heavily used for consistency checks across the rationale structure, to guarantee the identification of incompatible requirements, for instance. Finally, the *issue space*, relates together all the issues. It enables the system to propagate changes across decision problems, and to access issues that have been previously discussed, and sometimes resolved, as well.

3.1.1.2 DRL Constructs

Figure 3.2 shows the current type hierarchy of the DRL objects. An *Alternative* object represents a possible solution to the problem in consideration. A *Goal* object represents a state that decision makers want to achieve by making a decision, and serves as a criterion for evaluating alternatives. A D*ecision Problem* object represents the current decision to be made, i.e., it captures the current state of the process of selecting the optimal alternative. A *Claim* object can represent facts, arguments, assumptions, answers, and comments. An *IsRelatedTo* object is a special claim that links two other objects together, describing the particular relation between them. Some subtypes of IsRelatedTo are also included in the system to describe relationships more specifically. Such objects are mentioned in Figure 3.2 as well. A *Question* object is used to request information that the user needs in order to evaluate the current alternatives. A *Group* object relates objects of the same type. Objects may be grouped together by various group relationships, such as mutually exclusive, exhaustive, and so on. A *Viewpoint* object describes a particular state of the decision making process the user wants to capture. A *Procedure* object represents a procedure that has been used to answer a question. A *Status* object indicates the current state of an object, along with other relevant information, including the rationales that explain this state. A *Decided* object represents the state of a resolved decision problem, along with the alternative that was selected and the rationale behind this selection.

Figure 3.3 describes the structure for a decision graph, thus illustrating how the DRL relations are used to represent decision rationales. Chapter 5 provides a detailed listing of all the DRL constructs, including their respective meanings and attributes.

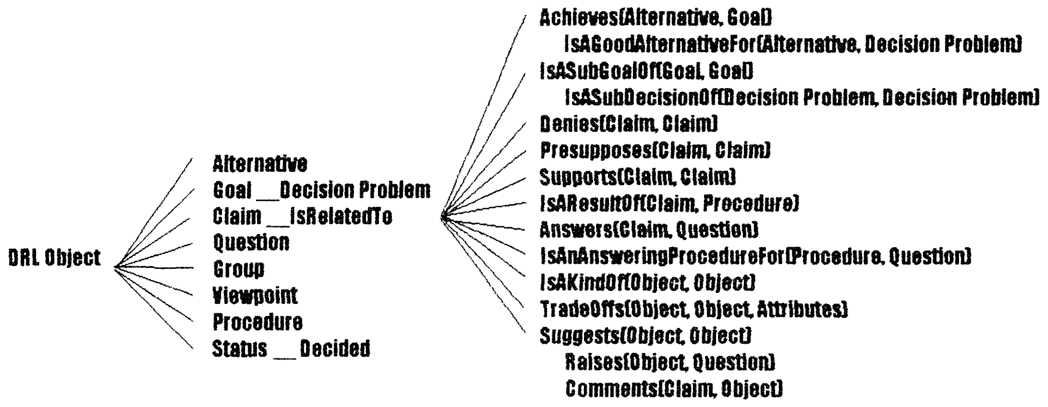

DRL Object
Alternative
Goal __ Decision Problem
Claim __ IsRelatedTo
Question
Group
Viewpoint
Procedure
Status __ Decided
Achieves(Alternative, Goal)
IsAGoodAlternativeFor(Alternative, Decision Problem)
IsASubGoalOf(Goal, Goal)
IsASubDecisionOf(Decision Problem, Decision Problem)
Denies(Claim, Claim)
Presupposes(Claim, Claim)
Supports(Claim, Claim)
IsAResultOf(Claim, Procedure)
Answers(Claim, Question)
IsAnAnsweringProcedureFor(Procedure, Question)
IsAKindOf(Object, Object)
TradeOffs(Object, Object, Attributes)
Suggests(Object, Object)
Raises(Object, Question)
Comments(Claim, Object)

Figure 3.2. DRL Type Hierarchy

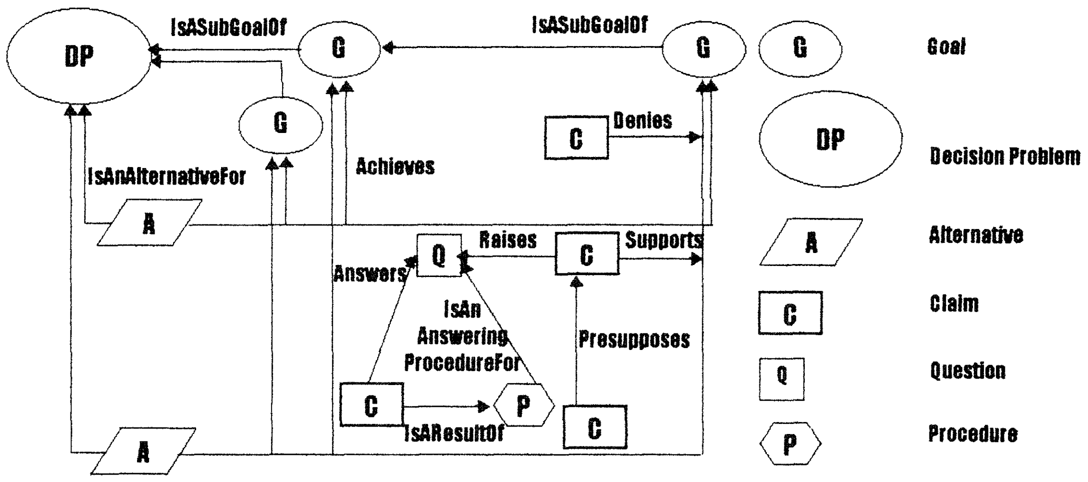


Figure 3.3. DRL Decision Graph Structure

3.1.1.3 Mapping the Constructs to the Spaces

The mapping between DRL constructs and DRL spaces is best illustrated by Figure 3.4. *The argument space* is represented in DRL by a set of related claims. Each claim has a Plausibility attribute to indicate the level of confidence the user has on the accuracy of that claim, and a Degree attribute to assess the degree to which that claim is relevant to the current problem. A claim can be supported, denied, or presupposed by another claim. Therefore, DRL relations can all be discussed and argued about, because they are all considered as special types of claims. *The alternative space* groups together alternative objects through IsAKindOf claims. We can argue about the alternative space by creating claims that argue about IsAGoodAlternativeFor, IsAKindOf, and Achieves relations. *The criteria space* deals with goals. In it, goals are related to one another through IsASubGoalOf relations. Goals may also be related through a group object that specifies the appropriate relation (mutually exclusive, independent, or partially overlapping). In either case, these relations can be argued about. *The evaluation space* is represented in DRL through the evaluation attribute of claims. For example, the Evaluation attribute of IsAGoodAlternativeFor represents the overall evaluation of an alternative as a solution to the corresponding decision problem. The semantics of this attribute is defined by the user, and it can have either absolute or relative values. *The issue space* in DRL incorporates decision problem objects, connected to one another by IsASubDecisionOf and IsAKindOf relations. IsASubDecisionOf relates a decision problem that is part of another one. For example, the problem of removing the topsoil from the construction site is a subdecision of the problem of clearing the construction site. IsAKindOf relates a problem that is a specialization of another. The problem of disposing of the removed topsoil is a specialization of the problem of moving an object from one location to another. Claims concerning these relations allow the user to argue about the issue space.

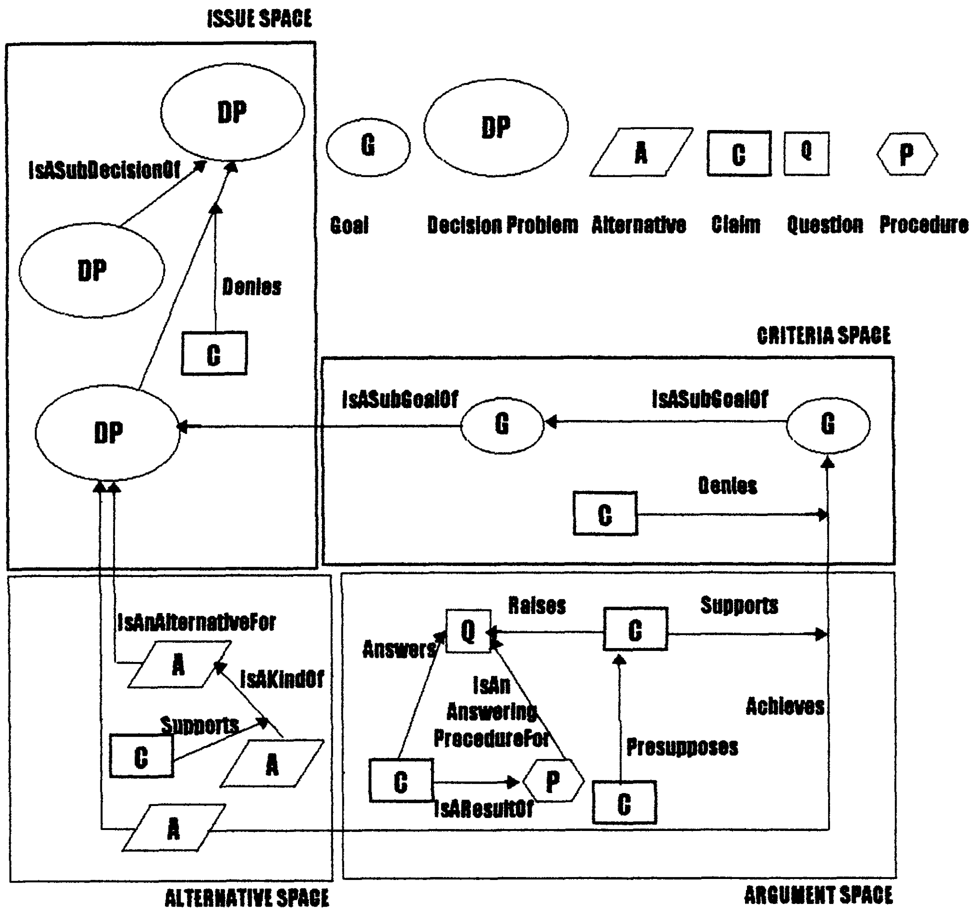


Figure 3.4. Mapping DRL Constructs to DRL Spaces

3.1.2. The Rationale Management Environment

SIBYL, the current rationale management environment developed by Lee imposes a five stage decision-making process on using DRL. For every decision problem, participants must set up an initial decision structure, release it, augment it, choose one alternative, and finally evaluate the outcome of the decision, as shown in Figure 3.5.

3.1.2.1 Setting up an Initial Structure

At this stage, the user creates a decision problem object for the decision to be discussed. The new decision problem object is then related to other objects in the system, through the issue space. At that point, the user specifies preliminary goals and alternatives in the same fashion, and the criteria space and the alternative space are updated accordingly. The user can now start to argue about the initial structure, through the argument space of the system. Details on this process will be better described at the third stage of the process.

3.1.2.2 Releasing the Initial Structure

During this stage, the user who created the decision structure makes it accessible to other decision-makers so that they can participate in the discussion. In this thesis, we will not discuss the communication protocols that were used to maintain global consistency. Rather, we relegate this topic to implementation considerations that are beyond the scope of this initial research. Possible models, however, can be found in Lee's thesis. The main requirement for the communication protocol is that it must ensure that all participants have consistent access to an updated version of the decision problem, consistent with all other versions, as quickly as possible.

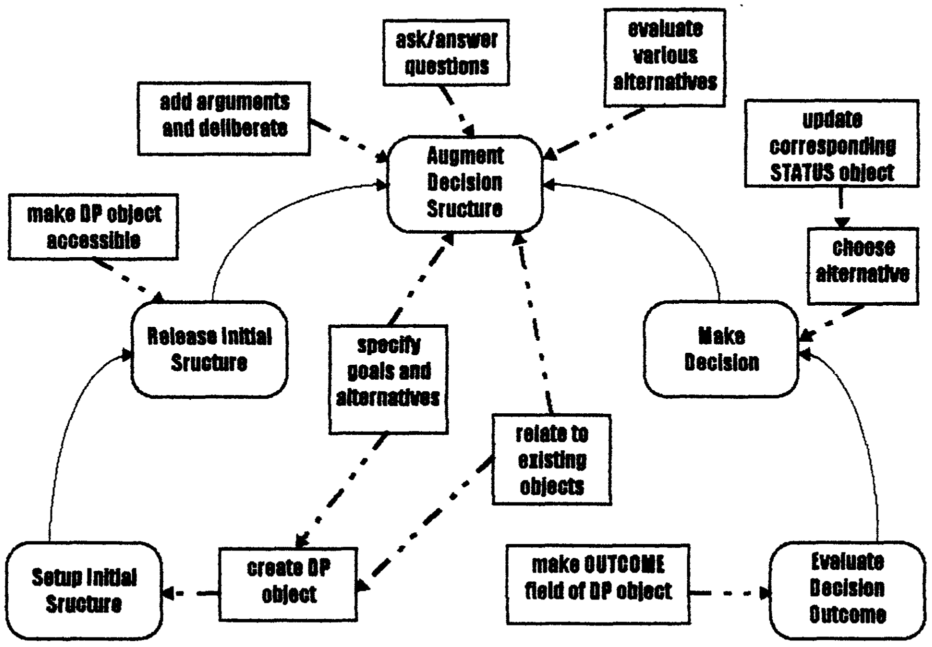

ask/answer questions
evaluate various alternatives
add arguments and deliberate
Augment Decision Sructure
update corresponding STATUS object
make DP object accessible
choose alternative
Release Initial Sructure
specify goals and alternatives
Make Decision
relate to existing objects
Setup Initial Sructure
create DP object
make OUTCOME field of DP object
Evaluate Decision Outcome

Figure 3.5. The Decision Making Process using DRL

3.1.2.3 Augmenting the Decision Structure

At this point, different users update the decision problem by relating it to an existing object, and specifying more goals and alternatives, as was described in the first stage. As the discussion gets more and more detailed, the criteria space is elaborated by creating new subgoals, and arguing about existing IsASubGoalOf relations. The alternative space can be augmented in a similar fashion. The argument space is also augmented, as users express their pro and con arguments concerning any claims that have been presented during the discussion, and ask and answer one another questions regarding previous comments that have been made. Finally, as the users assign evaluation attributes to DRL objects, the evaluation space is updated accordingly in order to reflect this new state of the decision structure.

3.1.2.4 Making the Decision

In SIBYL, the user—rather than the system—decides which alternative to choose. In case of conflicting opinions among users regarding this selection, a mediator may be chosen to settle the issue. When making his/her decision, the mediator considers the performance of the controversial alternatives with respect to the decision criteria, as well as the credibility and expertise of the various parties. This final decision-making process is usually done bottom-up. First, low-level decision problems are resolved. The deliberation participants then move up the decision structure stage by stage, along IsASubDecisionOf relations and IsAKindOf relations of the issue space and the alternative space. At each stage of this traversal, the interactions among the selected solutions are taken into account to prevent inconsistencies.

3.1.2.5 Evaluating the Outcome of The Decision

The value of a decision can be assessed when the decision is actually carried through. Through the argument space, the participants can discuss, for example, whether that particular choice was good or bad, or how it could have been done better. Previously recorded rationales describing the context in which the decision had been made are also examined. This new evaluation information is made possible through the precedent management capability of the system, as will be discussed in the following section.

3.1.3. Computational Services

The system uses captured decision rationales to provide the users with services that support decision-making. The precedent manager allows the user to access past decisions that may be relevant to the current one. The dependency manager deals with propagating the effects of changes and maintaining consistency across decisions. The viewpoint manager lets the user create multiple viewpoints of the same issues and compare them. Other bookkeeping modules are also part of the system to accurately monitor its use.

3.1.3.1 Precedent Management

The precedent manager responds to the user's need to transfer useful knowledge from past decisions to the current one. In selecting which past decisions are relevant to the current problem, SIBYL compares the decisions represented in the system according to the number of common goals with the current problem, and retrieves those decisions that rank above a predetermined threshold value of common goals. This comparison is done through a goal lattice, which consists of a network of goals in a given task domain, related through specialization (subtype) and/or example (instance) links. For each goal in the original structure, SIBYL retrieves examples of its goal type from the goal lattice. The user identifies the potential matches, and the corresponding decision problems are accessed. At this point, potentially relevant rationales from the chosen decisions are retrieved, adapted to the current decision problem, and related to the existing objects in the current decision.

3.1.3.2 Dependency Management

The dependency manager in SIBYL is based on Object lens rule system. It provides agents to monitor changes and invoke appropriate rules to carry on the desired actions. In this rule system, however, the changes to be monitored are restricted to the addition of new items, the modification of existing items, the specification of time, and the specification of events. Similarly, the actions to be executed consist of moving, copying, and deleting objects or fields of objects that satisfy the condition described in the rule. This scheme can capture the fact that the subgoal of a goal whose importance is low, has low importance. Because of these restrictions, however, many dependencies cannot be expressed. Dependencies that require comparison of two objects cannot be captured. For example, the constraint that requires the evaluation of IsAGoodAlternativeFor between alternative A and decision problem P to be low, when the importance of goal G is high and the evaluation of Achieves between A and G is low, cannot be expressed, because variable binding is not possible in this framework. This restriction can be removed by using a full-fledged rule system. This would require the user to access the underlying programming language that requires higher user expertise, hence making the user interface more difficult.

3.1.3.3 Viewpoint Management

Viewpoint management is used to represent and compare multiple decision states. This capability is useful for comparing states that describe the same objects with different attributes, states that describe sets of objects included in others, states that describe overlapping sets of objects, and states that are historically related. SIBYL represents viewpoints with a viewpoint object. This object has attribute *Elements* that point to the objects in the viewpoint, and attribute *Viewpoint Relations* that relate viewpoints to one another, through one of the relations just described. Furthermore, objects shared by multiple viewpoints are represented by one different copy for each viewpoint, but different copies have the same identifier, to allow consistent change propagation across viewpoints.

3.1.3.4 Other Services

SIBYL also offers other services that complement the ones just described. We choose not to describe them in great detail, as our framework for a project management system will not modify the way these services are managed. Although we do introduce them in this section, we refer the reader to the original work as described by Lee in his thesis for a more complete information on the respective topics. *Evaluation Management* is the capability of a rationale management system to allow users to evaluate the alternatives brought up throughout a decision-making process, and to effectively propagate the decisions they make. *Communication Management* is of particular relevance when the parties involved in related deliberation processes are numerous and scattered, and consistent and rapid propagation of changes throughout the deliberation process is primordial. Finally, *Documentation Management* is essential for capturing and storing acquired knowledge for future use.

3.2. Transition Space: The Knowledge Base

The motivation behind an event representation opposed to a state representation stemmed from the need to understand and represent causal behavior of physical systems. It was indeed felt that such informal descriptions are often used by humans as a last resort to describe their knowledge. Gary Borchardt developed such a representation to describe physical events as sequences of transitions, or combinations of changes in attributes of objects involved in the events. In this framework, partial matches between transition sequences are then identified, in order to connect events into causal chains and fill causal gaps in an informal description using precedent events. Furthermore, description discontinuities due to the use of analogy and abstraction are bridged using transformations and rules of inference that operate on representations of events.

In the following, we discuss how events are represented and matched in Transition Space (TS). We then introduce the transformations that manipulate these representations, in order to facilitate the matching process. Furthermore, we briefly describe the current causal reconstruction cycle in TS, as performed by pathfinder. We conclude by presenting the assumptions underlying this framework.

3.2.1. Representing Transitions and Events

Humans rely a lot on written information such as encyclopedias and reports to store and communicate causal knowledge. However, causal behavior in complex descriptions concerning intuitive concepts and metaphors is usually still expressed verbally.

In Transition Space, events are represented as a sequence of transitions. A Transition is a combination of changes occurring in given aspects of a situation within a specified time interval. More specifically, a transition is a set of assertions at and between two time points, and events are sequences of transitions or event traces. Figure 3.6 illustrates the transition space thus defined, with transitions represented by individual points and events depicted by sequences of points, or paths relating transitions to one another.

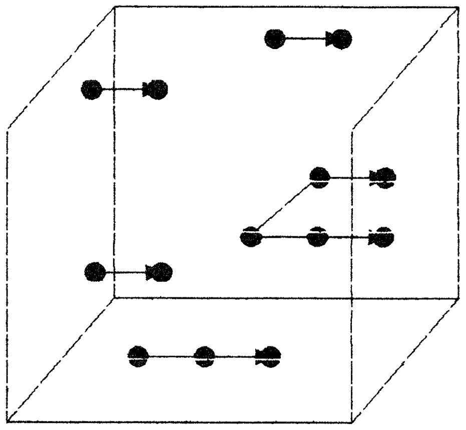


Figure 3.6. Representation of Transition Space

3.2.1.1 TS Predicates

Ten *change predicates* are used to described such changes, APPEAR, DISAPPEAR, CHANGE, INCREASE, DECREASE, together with their respective negations. They take four arguments: an attribute, an object or tuple of objects, and two time points representing the time interval of concern. Therefore, the assertion NOT-DISAPPEAR (inside, <the-water, the tank>, t9, t10) represents the statement "The Water remains inside the tank."

Four *primitive predicates* that take five arguments—EQUAL, GREATER, and their respective negations- are also added to complete the basis for the representation. They are used to define six higher-level predicates to describe single time point assertions.

Throughout this thesis, we will use A, D, C, and - to represent the first set of change predicates and precede them with / to represent their respective negations. The assertion above thus becomes /D(inside, <the-water, the-tank>, t9, t10). We will also use the graphic representation to describe event traces. Each transition is represented by a column describing assertions between vertical lines depicting time points. For example, the graphic representation of Figure 3.7 corresponds to the event space representing object O1 pushing away another object O2 in two transitions.

In the remainder of this thesis, when representing transition space event traces, each row represents the behavior of a particular object during the event, except for the first row, which describes the time frame during which the event is described. The first column represents the object whose behavior is being described, along with the point of reference used for such description. The second column states the object's attribute that is examined. The other columns describe the actual changes, or the transitions, that occur. Thus, in the first transition of the event of Figure 3.7, the objects are in contact and pressure is applied. In the second transition, there is no more contact nor pressure, and the position of O2 changes.

	t1	t2	t3
[Object O1, Object O2]	**Distance**	$\not{A}$	A
[Object O1, Object O2]	**Contact**	$\not{D}$	D
[Object O1, Object O2]	**Pressure**	A	D
[Object O1, Backgnd]	**Position**	$\not{A}$	A
[Object O1, Backgnd]	**Speed**	$\not{A}$	A
[Object O1, Backgnd]	**Heading**	$\not{A}$	A

Figure 3.7. Event Trace Depicting Event "Push Away"

3.2.1.2 TS Event Matching

Inter-event association may be detected in a causal description by identifying partial matches between the event traces. There are two classes of partial matches between event traces. In partial chaining matches, a non-initial transition in one trace partially matches the initial transition in the other trace. In partial restatement matches, traces match in other ways, as will be described in the next section. Consider the following two examples, as described by Borchardt:

The steam move sonto contact with the metal plate.
The steam condenses the metal plate.

Their respective event traces are represented in Figure 3.8(a).

The partial chaining identified in this case relates to distance disappearing and the contact appearing between the steam and the plate. The transitions in this match, however, have distinct points, and contain additional specifications. A complete match between these transitions occurs after the original traces are transformed. While Figure 3.8(b) illustrates the result of these transformations. In the transition space representation, this process thus corresponds to the transformation of the two original vectors representing the two transition events, into two new vectors that have one common point, corresponding to matching transition.

		t21	t22	t23
[TheSteam, Backgnd]	**Position**	Δ	Δ	
[TheSteam, Backgnd]	**Speed**	D̸	D	
[TheSteam, Backgnd]	**Heading**	Δ̸	D	
[TheSteam, ThePlate]	**Distance**	-	D	
[TheSteam, ThePlate]	**Contact**	A̸	A	

The steam moves into contact with the plate.

		t31	t32	t33
[TheSteam, ThePlate]	**Distance**	D	A̸	
[TheSteam, ThePlate]	**Contact**	A	D̸	
[TheSteam, Vapor]	**IsA**	D̸	D	
[TheSteam, Liquid]	**IsA**	A̸	Δ	

The steam condenses on the metal plate.

(a) Original Event Traces

		t21	t31	t32	t33
[TheSteam, ThePlate]	**Distance**	-	D	A̸	
[TheSteam, ThePlate]	**Contact**	A̸	A	D̸	

(b) Final Event Trace

Figure 3.8. Partial Matching using Exploratory Transformations

3.2.2. Exploratory Transformations

Most often than not, events are described differently by different people at different times. These discrepancies appear in the transition space framework as differences of objects in the event traces, differences of attributes that describe these objects, and differences of time points. Transformations are used in event matching to go around these differences, in search of partial matches between event traces. These transformations are therefore the means by which the system deals with analogies and abstractions that may be found in the user's informal description. There are two types of transformations. Information-preserving exploratory transformations can be inverted, i.e., if event trace T1 is transformed into event trace T2, then the same transformation maps T3 into T1. Non-information-preserving exploratory transformations do not, as the information captured in traces T1 and T2 are not the same. Equivalence and substitution are information-preserving transformations. Generalization, interval composition, attribute composition, object composition, attribute-object reification, event-attribute reification, and event-object reification, on the other hand, are non-information-preserving. Figure 3.9 is an illustration of these transformations. (a) *Equivalence* is the replacement of time points, attributes, or objects with synonym quantities to produce an equivalent description of the same event. Attributes "inside" and "is-in" are equivalent. (b) *Substitution* is the replacement of quantities with other quantities in a new context, in order to relate different events that undergo parallel changes. Thus, if the gate of the site is open, then the site is considered to be open as well. (c) *Generalization* is the replacement of reference term with a more general type. If object Truck768 is a truck and all trucks are used as conveyance vehicle to transport various materials, then Truck768 may be used for this purpose too. (d) *Interval Composition* is the merging of two adjacent time intervals into one, with changes specified as the composition of the original changes. Describing a truck slowing down and then stopping is similar to describing it stopping. (e) *Attribute Composition* is the re-expression of an activity involving different attributes with an activity involving one, encompassing attribute. Therefore, if the width of the gate decreases, with its height unchanged, then the overall size of the gate decreases. (f) *Object Composition* is the re-expression of activity involving the parts of an object with activity involving the whole object itself. If some union goes on strike, and a particular worker is a member of that union, then it is fair to assume that he/she would go on strike as well. (g) *Attribute-Object Reification* replaces activity involving an attribute with activity involving a new object representing the attribute applied to its argument. Thus, stating that the size of the gate changes is the same as stating that the amount of the gate's size is changed. (h) *Event-Attribute Reification* replaces part of an event trace with a newly introduced attribute applied to one of the participating objects. For instance, the event describing the topsoil removal activity may be substituted by the description of changes in the IsRemovedFrom attribute of TheTopsoil object. (i) finally, Event-Object Reification replaces part of an event trace with a new object representing that activity. Again, the event describing the topsoil removal activity may also be described by stating that the topsoil removal activity is completed.

t21 t22 t23

[TheTopsoil, Backgnd]	**DisposedOf**	$\cancel{A}$	A
[TheTopsoil, TheSite]	**Inside**	$\cancel{D}$	D

The topsoil is inside the site.

t21 t22 t23

[TheTopsoil, Backgnd]	**DisposedOf**	$\cancel{A}$	A
[TheTopsoil, TheSite]	**IsIn**	$\cancel{D}$	D

The topsoil is in the site.

(a) Equivalence Transformation

t21 t22

[TheGate, TheSite]	IsPartOf	$\cancel{D}$
[TheGate, Backgnd]	IsOpen	A

The gate of the site is open.

t21 t22

[TheGate, TheSite]	IsPartOf	$\cancel{D}$
[TheSite, Backgnd]	IsOpen	A

The site is open.

(b) Substitution Transformation

t21 t22

[Truck768, Truck]	**IsA**	A
[Truck, ConveyanceVehicle]	**AKindOf**	$\cancel{D}$

Truck768 is a truck, which is a conveyance vehicle.

t21 t22

[Truck768, ConveyanceVehicle]	**IsA**	A

Truck768 is a conveyance vehicle.

(c) Generalization Transformation

t21 t22 t23

[TheTruck, Backgnd]	**Position**	Δ	Δ
[TheTruck, Backgnd]	**Speed**	-	D
[TheTruck, Backgnd]	**Heading**	$\cancel{D}$	D

TheTruck slows down, then it stops.

t21 t23

[TheTruck, Backgnd]	**Position**	Δ
[TheTruck, Backgnd]	**Speed**	D
[TheTruck, Backgnd]	**Heading**	D

TheTruck comes to a stop.

(d) Interval Composition Transformation

t21 t22

[TheGate, Backgnd]	**Height**	$\cancel{\Delta}$
[TheGate, Backgnd]	**Width**	-

The width of the gate decreases.

t21 t22

[TheGate, Backgnd]	**Size**	-

The size of the gate decreases.

(e) Attribute Composition Transformation

t21 t22

[XYZUnion, Backgnd]	**OnStrike**	A
[John Doe, XYZUnion]	**MemberOf**	$\cancel{D}$

Union XYZ is on strike and John Doe is member.

t21 t22

[John Doe, Backgnd]	**OnStrike**	A

John Doe is on strike.

(f) Object Composition Transformation

t21 t22

[TheGate, Backgnd]	**Size**	Δ

The size of the gate changes.

t21 t22

[SizeOfTheGate, Backgnd]	**Amount**	Δ

The value of the size of the gate changes.

(g) Attribute-Object Reification Transformation

t21 t22

[TheTopsoil, TheGround]	**Contact**	D
[TheTopsoil, TheGround]	**Distance**	+
[TheTopsoil, Backgnd]	**Position**	Δ
[TheTopsoil, Backgnd]	**Heading**	A

The event trace describing the topsoil removal.

t21 t22

[TheTopsoil, TheGround]	**IsRemovedFrom**	A

The topsoil is removed from the ground.

(h) Event-Attribute Reification Transformation

	t21	t22
[TheTopsoil, TheGround]	**Contact**	D
[TheTopsoil, TheGround]	**Distance**	+
[TheTopsoil, Backgnd]	**Position**	Δ
[TheTopsoil, Backgnd]	**Heading**	A

	t21	t22
[TheTopsoilRemoval, Backgnd]	**IsCompleted**	A

The event trace describing the topsoil removal. The topsoil removal is complete.

(i) Event-Object Reification Transformation

Figure 3.9. Exploratory Transformations

3.2.3. Pathfinder

Pathfinder is the program that monitors causal reconstruction in the current implementation of TS. It performs this task in four stages, as represented by Figure 3.10.

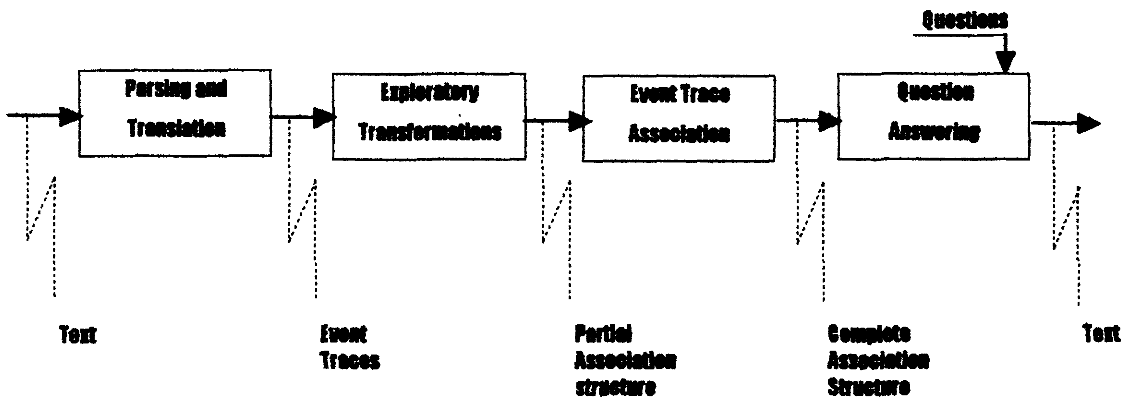


Figure 3.10. A Four Stage Causal Reconstruction Process

First, event traces are formed for events referenced in the input text descriptions. Such descriptions may include event references, background statements, meta-level statements, rules of interference, and rules of restatement. Pathfinder then extends the event traces through interference and exploratory transformations, producing partial event associations describing each event in several ways. During the next phase, partial matches between different trace associations are identified and elaborated, hence resulting in a completed association structure. Finally, Pathfinder uses this structure to retrieve information from the knowledge base, and answer questions. More details regarding this process are given in subsequent chapters, when we describe TS in the context of project management.

3.2.4. TS Assumptions

In the development of the transition space representation, a few assumptions were made, in order to facilitate the testing of the system. First, Gary simplified the task of understanding causal descriptions (the interpretation of descriptions), in order to be able to perform causal reconstruction (the explanation of events). Also, he used simplified English syntax conventions, in order to be able to concentrate on representing the semantics of the information conveyed. The knowledge necessary for causal reconstruction has been explicitly included in the system to eliminate the need of general-purpose knowledge base. Finally, although most of the processing relies on heuristics and matching, the representation itself is based on predicated logic notation, thus improving the performance of Pathfinder.

3.3. mSIBYL: A Design Documentation System

Design documentation refers to information generated during the product design process. Thus, design documentation systems are used to represent design knowledge for future reference, manage this knowledge across the different design groups participating in the development of a product, and document the rationales behind the decisions made in the design process. Design management, on the other hand, consists of planning and managing the different tasks that arise during product development. mSIBYL, the design development and management system developed by Lukas Ruecker, deals with both design documentation and design management. It is concerned with the flow of information both across products from the past to present to future, and acrossgroups within product.

In the following, we present the general guidelines underlying the framework of mSIBYL. We describe the basic element of knowledge represented in the system. Next, we describe how both SIBYL and TS are extended. We finally illustrate how they are integrated into mSIBYL.

3.3.1. Assumptions Underlying the mSIBYL Framework

The motivation behind mSIBYL stems from the need for computer-based systems to both represent and manage knowledge generated throughout a design process for on-going, concurrent, and future reuse. This framework is based on a few assumptions. First, decision making is a process that occurs at all stages and all levels of the product development. It is thus viewed as a sequence of decision making activities at different levels of abstraction and detail. Also, decision making is a group activity, in which people argue and deliberate, so that design is viewed as a group effort where people try to convince one another through rational argumentation. Furthermore, most of the design knowledge cannot be formalized, so a semi-formal representation is required, in which only part of the knowledge is accessible. Finally, engineering processes are usually described in terms of their functions, therefore objects and their behavior should be represented by their attributes and changes of those attributes over time.

3.3.2. Basic Knowledge Element for mSIBYL

In mSIBYL, the basic element of information is referred to as a "chunk" of information. It is designed as "a consistent amount of design knowledge sufficient and necessary to fulfill a singular purpose at a given point of the deliberation process surrounding a design decision." It is a two-dimensional token that represents (1) atomic (it cannot be broken down) design knowledge (2) for a particular purpose in a discussion. *The content* of a chunk corresponds to the actual design knowledge captured in the chunk, while its *context* refers to the role of the chunk in the deliberation process. Referring to the site clearing problem, using a labor-intensive solution and using an equipment-intensive solution are two different alternatives for the problem being discussed. They are addressed in the same context, i.e., they are both mentioned as alternatives to a problem, but they have different design contents, they mean different things and refer to different solutions. Hence, both chunks of information have the same rationale context field but different design contents.

In mSIBYL, a chunk of knowledge is represented as a three-field object. A *rationale context* field relates chunks used in similar deliberation processes. A *design content* field relates chunks that represent functionally similar products. A *comments* field represents all inaccessible information related to the chunk. This representation therefore allows the system to maintain a complete record of the design activity. It also enables the designer to set which information captured in a chunk needs to be formalizedmSIBYL uses DRL to capture the rationale context of a chunk, and TS to capture its design content.

3.3.3. The Rationale Context: Using SIBYL

The decision to use DRL to capture the rationale context stems from its property to efficiently represent and access informal knowledge in a structured way. It also provides multi-dimensional format of documentation that clearly captures relationships among pieces of design knowledge. Furthermore, such representation enables access of information "at any location of the dialectical network, and at any time during the design process." Finally, the computational services provided by SIBYL have direct applications in managing design knowledge during product development.

In using SIBYL, the decision rationale language is modified into mDRL to better fit the design documentation domain. Two new objects are introduced. An *Artifact* object represents the evolving design artifact, hence capturing the current status of product development. A *Plan* object is used to describe the sequence in which subdecisions of a design problem must be resolved. Furthermore, three relations are introduced. *Qualifies(C1, C2)* states that the content and/or impact of argument C2 is modified by argument C1. *IsALikelyCauseFor(C1, C2)* represents the claim that argument C2 is true whenever argument C1 is true. Also, *IsAGoodReasonFor(C, decided)* claims that argument C is one of the reasons the alternative in Decided has been selected. Finally, mDRL objects include three more fields, TS Text, TS Graphics, and TS Trace to describe the design content of the object, as described below.

3.3.4. The Design Content: Using TS

The decision to use a transition space in mSIBYL is based on the assumption that the function and behavior of objects in the design process must be represented by a set of attributes and changes of those attributes over time. Hence in TS, a sequence of transitions describes the behavior of artifacts or processes, thus representing the changes in attributes of objects modeling these artifacts or processes. Indeed, an artifact can be described by qualitative or quantitative attributes, and the change predicates provided by TS can then be used to capture the overall behavior or artifact over time. Thus, not only does TS represent the different components of the current artifact as objects, but, it can also capture the design specifications that have been fulfilled yet through attributes and constraints. Furthermore, the transformations provided by TS enable mSIBYL to deal with both lexical discrepancies, and alternative similarities. On one hand, it can compensate for the different descriptions of information across both time and users. On the other, it can relate, and thus transfer engineering solutions across domains.

In mDRL, the TS fields previously described represent the design content of a chunk and relate it to design contents in other fields. The TS Text and TS Graphics differ from the inaccessible comments field in that they are accessible, since the system is supposed to understand and use that information. Indeed, the design content of a chunk is described in either the TS Text field or the TS Graphics field. The system then interprets these descriptions and generates corresponding event traces that are stored in TS Trace field. Of course, the design content can be entered into the system directly through this field as well, in which case the system generates the appropriate description in the TS Text field.

3.3.5. Integrating SIBYL and TS into mSIBYL

The overall motivation behind the conceptualization and development of mSIBYL was to provide a computer-based system to (1) capture design knowledge generated in the product development process, and (2) provide computational services to facilitate use and reuse of that design knowledge. In the previous sections, we described how the framework of mSIBYL achieves the first goal by providing its users with means to learn from precedents, handle different alternatives, handle a changing world, make decisions, communicate with other engineers, and document the design process. Furthermore, mSIBYL provides these services along both the context and the content dimensions. In addition to the services directly inherited from SIBYL, mSIBYL also provides these services using the design knowledge captured in TS. When searching for design knowledge, for instance, the TS Trace fields of mDRL objects are accessed. TS transformations are then used to identify descriptions of related functions and behavior. The corresponding dialectical structures may then be retrieved. At that point, the precedent knowledge is examined, extracted, and adapted to the current context. The other services provided by SIBYL are similarly augmented to make full use of the design knowledge in mSIBYL.

CHAPTER 4

A PROJECT MANAGEMENT SYSTEM

This chapter introduces our framework for a project management system. We start by presenting the general criteria that we consider important in the development of a construction project management system. After that, we describe how our design integrates concepts of the three systems discussed in the previous chapter. In doing so, we compare our framework with mSIBYL, highlighting the features that we imported from that system. Then, we explain the extensions that need to be made to SIBYL in order to better fit the construction domain. Furthermore, we discuss how TS is used to represent construction-related issues. Finally, we describe how the integrated system satisfies the criteria we originally identified.

4.1. Guidelines for a Project Management System

A project management system should provide the computational services introduced in previous chapters. Namely, it must provide users with tools to (1) use past experience, (2) examine different points of view of an issue, (3) keep track of dependencies across decisions and tasks, (4) evaluate different alternatives, (5) communicate among one another, (6) document their contributions to the project, and (7) maintain an efficient and up-to-date project schedule. In order for these services to be functional, however, the system must possess certain properties. In the following, we present some of these properties, and discuss their importance.

4.1.1. User Interaction

In the construction project domain, unexpected events occur frequently throughout a project's lifetime, and the project is therefore constantly altered and updated. The different project participants will have to constantly interact with one another, in order to keep track of potential changes in the project requirements. Consequently, a project management system cannot be fully automated. Rather, provision for heavy interaction must be anticipated, so that the user can guide the system through the different decisions that need to be made. Because of this heavy relationship between user and machine, the system interface should be designed to be both natural and practical. It should provide the user with easy access to and monitoring of the system. Also, it should do so efficiently, so that the user prefers such a system over the traditional methods of project communication and documentation.

4.1.2. Flexibility

Another important property of project management systems, specially in the construction domain, is flexibility. In fact, we define various types of flexibility. First, there is *interface flexibility,* already mentioned in the above section. As the participants of the construction project have to constantly access and manage the information stored in the system, the interface should be designed so that extensive expertise in the system's operations is not required by its users. Then there is *communication flexibility.* Since the different participants of a project have different backgrounds, and thus different concerns and goals, the system should be flexible enough to enable them to communicate among themselves efficiently and accurately. It should therefore be able to capture the right meaning of the information the user provides, to interpret the information accurately, and also to retrieve the desired information, in order to bridge communication gaps occurring between different participants. Finally, *structure flexibility* refers to the ability of a system to capture and manage different amounts of information, at different levels of detail. For instance, a user would not have to incorporate a complete description of a proposed alternative. Thus, implicit information known to all participants would not be incorporated, hence allowing users to discuss only issues that are relevant to them.

4.1.3. Reusability

Another property of the construction domain is its relative stability, in the sense that the same techniques and the same tools are used from project to project. Building a three-story office block involves the same tasks, and raises the same issues as building a five-story building, for instance. Similarly, constructions in areas that have similar soil properties and/or topography will require similar types of foundations. A construction project management tool should therefore provide its users with the ability to reuse any information stored during previous projects, whether it has been stored by themselves or by other participants. Furthermore, the 'invariance' property of the domain enhances the reusability feature. Indeed, the fact that similar site topography entails similar foundation techniques suggests that topology may be a good index to access precedent. In such a case, users may retrieve previous construction projects located next to cliffs, for instance.

4.1.4. Incremental Formalization and Growth

Another desirable property is the ability to accept incomplete object descriptions, so that the users are not required, for instance, to fully define a task, nor to fully explain an argument, nor to examine all the possible alternatives at once. The system should thus be able to accept descriptions at different levels of detail and abstraction by different project participants. Furthermore, because of the occurrence of unexpected events during a project, it is very difficult to a priori encode all the knowledge a system may need to help manage the project. Therefore, the users should be able to add new information as it becomes available to them throughout the lifetime of the project. Such project management systems would, therefore, increase their knowledge over time in two ways. The knowledge description is refined as users get into more detailed levels of decision making, and the knowledge volume increases as new information is captured.

4.1.5. Three-Dimensional Information Management

Finally, we are interested in project management systems that can provide their users three different aspects of information management. First, they should capture the semantics of the information. That is, they should capture what that information consists of and what it refers to. Second, they should capture the dialectical role of a particular information, when managing the decision rationales brought up during the project. Indeed, they should reflect the context in which that information is brought up and analyzed. Third, they should maintain up-to-date project schedules that can be appropriately modified when previous decisions are changed or when new decisions are made. They should therefore be able to have a time-wise ordering of the project tasks, taking into consideration the various dependency relationships and constraints that exist between tasks.

4.2. Incorporating mSIBYL Features

Our framework for a project management system that maintains dependencies among decision rationales incorporates DRL and TS in a way similar to mSIBYL. We thus start this section by presenting the motivation behind our decision to design a system that is so close to mSIBYL. We then describe the features of mSIBYL that we have decided to use in the project management system we propose.

4.2.1. Why are mSIBYL Features Imported?

mSIBYL was developed to document and manage the information generated during the design process of mechanical artifacts from the conception phase to the disposal phases. The scope of our research, on the other hand, consists of developing a system to capture and manage rationales that arise in construction projects, along with the different dependency relationships that relate them to one another. Our motivation to develop a system close to mSIBYL is threefold. First, there is a similarity in the domain of operations of both systems. Product development and construction projects are similar processes that have similar properties and raise similar issues. Second, both systems have features that complement one another. The research behind mSIBYL focuses specifically on the capability of the system to learn from precedents, while we concentrate on the dependency management isseu. Third, perhaps the main reason for our choice is that both systems rely on the same underlying technologies, namely SIBYL and TS. SIBYL is used to capture the dialectical role of objects, while TS captures their behavior and function.

4.2.1.1 Similar Domains...

A construction project can be thought of as a special case of product development, where the product represents the structure to be designed, built, and maintained. In both cases, the overall process consists of identifying the problem from the user specifications, designing a strategy to resolve the problem, implementing the selected strategy to fulfill the specifications, and maintaining and evaluating that strategy. Because of these similarities, the same assumptions can be made regarding the design of both systems. Indeed, decision making is invariant throughout both types of processes: It is a group activity in both domains, the knowledge generated and used is semi-formal in both cases, and finally, both domains hold engineering processes that are usually described in terms of their functionality. Furthermore, objects that appear in project development as end-products may also appear in construction projects as tools. Therefore, mechanical artifacts designed and developed using mSIBYL may later be used to perform tasks of a construction project. Using similar representations of objects that appear in both contexts is therefore practical in transferring the relevant information regarding these objects and their use.

4.2.1.2 Complementary Issues...

mSIBYL was primarily conceived to "document and manage design knowledge in on-going, concurrent, and subsequent product development processes." While it does provide computational services to learn from precedents, handle different alternatives, handle a changing world, make decisions, communicate with other engineers, and document the design process, its main emphasis is precedent management. mSIBYL enables its users to access previous design knowledge according to both the dialectical role of that knowledge, and the description of functions and behaviors related to that knowledge. The system we propose still provides the services previously listed. In addition, we also include a module that helps users deal with scheduling issues as such. However, our main concern throughout this research is the representation and management of dependency relationships that arise between the various pieces of the information generated. Both systems could therefore be viewed as complementary systems, since they provide complementary services, namely precedent management and dependency management.

4.2.1.3 Same Technology...

mSIBYL relies on the representation introduced in SIBYL to capture the dialectical role of design knowledge in the deliberation process, and on TS representation to capture the behavior described by this knowledge in the system. Since the representation of decision rationales raises the same issues for both the design of mechanical artifacts and the management of construction projects, SIBYL can therefore be used for both systems. Also, since both systems are designed to manage objects across their semantics as well as their dialectical roles, and most objects represent the behavior and functions of artifacts in the corresponding domain, the transition space representation is adequate in both systems. Furthermore, since both systems focus on complementary services, while targeting complementary domains, there may be an opportunity for incorporating the features of both systems into one single framework. Therefore, making close design decisions for both systems would facilitate any such future integration. Hence, mSIBYL and TS are used in both systems similarly, as they handle similar aspects of the systems.

4.2.2. What mSIBYL Features are Imported?

In order to maintain design compatibility between mSIBYL and our system, we decided to use a similar basic knowledge element. Also like mSIBYL, the system we propose uses DRL to capture the rationale context of design knowledge and augment it with new elements. Furthermore, transition space is used—as in mSIBYL—to represent the behavior and function of objects that are used during the construction project. Finally, the computational services provided by SIBYL and updated in mSIBYL are still present in our system to better assist users in managing and monitoring a construction project.

4.2.2.1 Basic Knowledge Element

In our project management system, we still define the basic element of information, a chunk, as "a consistent amount of design knowledge sufficient and necessary to fulfill a singular purpose at a given point in the deliberation process surrounding a design decision." In mSIBYL, the basic chunk consisted of three fields to hold the context field (the role of the chunk in the deliberation process), the content field (the actual meaning of knowledge captured in the chunk), as well as the user's comments (inaccessible additional information). The problem at this point is therefore to incorporate the scheduling dimension into the knowledge element.

First, we considered redefining the basic chunk to be a three-dimensional token, by adding a task field (the scheduling dimension) to represent the task that results from the deliberation process. A task object is created only when the decision problem has been resolved, and a Decided object has been appropriately incorporated. Thus, knowledge chunks that refer to other DRL objects will not use the new field.

We then thought of defining two types of basic elements. Type I has a content field and a context field, and type II has a content field and a task field to access the scheduling dimension. Hence, while a type I chunk C1 would refer to a Decided object, a type II chunk C2 would refer to a Task object. But the content fields of both C1 and C2 now hold similar information, since the alternative described in C1 is the same as the task described in C2.

Finally, according to the scheme we decided to use, the rationale dimension and the scheduling dimension are both accessed from the context field. For both cases, the content field and the context field represent different aspects of the same piece of information, as opposed to the first scheme. Furthermore, the only structural difference between a rationale object and a scheduling object is that the former can be argued about while the latter cannot, as we will explain in the following section. The logical distinction, i.e., the distinction in the type of knowledge captured, does not affect the way they should be accessed and manipulated. Thus, while we will differentiate between them at the logical level, we do consider them to be the same at the representational level. Hence, in our system, the basic knowledge element is therefore the same as the one in mSIBYL, with the context field referring to both the rationale dimension and the scheduling dimension using DRL, and the content field accessing the semantics dimension using TS.

4.2.2.2 The Context Dimension

Like mSIBYL, the project management system we propose takes advantage of the features of DRL. Namely, (1) informal knowledge can be represented efficiently and accessed systematically. Also, (2) the two-dimensional knowledge organization allows the system to clearly capture relationships among pieces of knowledge. Furthermore, (3) the network structure of DRL enables easy access of information at any time. Finally, (4) the services offered in SIBYL have direct applications in construction project management.

In our system, the decision rationale language is augmented to better fit the construction project domain. Three new objects are introduced. A *Project* object is added to represent the evolving schedule of the project, hence capturing both its current status and the tasks that are yet to be performed. A *Task* object is included in order to represent selected solutions and bridge together the rationale dimension and the scheduling dimension. A *Resource* object is used to capture scheduling attributes describing resource availability, resource allocation, and resource incompatibility. Three new relations are also introduced. *IsATaskFor(T, DEC)* is the claim that task T is the result of selecting the alternative solution described in object DEC. *IsAMotivationFor(C, D)* states that argument C is one of the reasons why decision problem D is considered at all. *IsAnAvailableResource(R, DP)* is the claim that resource R is available to decision problem DP.

4.2.2.3 The Content Dimension

Knowledge chunks access the semantics dimension through the TS Text and TS Trace fields, as defined in mSIBYL. TS is therefore used to describe in a sequence of transitions the behavior of processes mentioned in the project. Like in mSIBYL, these transitions reflect the changes in attributes of objects representing these processes. The user decides which information in the chunk is to be encoded, and at which level of detail and abstraction it should be done. Such information is then described in the TS Text field. The system then interprets these descriptions and generates corresponding event traces that are stored in the TS Trace field. Transformations may then be applied to help the system perform various computational services. More specifically, users will be able to learn from precedents, handle different alternatives, handle a changing world, make decisions, communicate with other engineers, and document the construction project. SIBYL provides these services along the rationale dimension. Like in mSIBYL, we will use TS to provide these services along the semantics dimension as well, as we describe later.

4.3. Incorporating the Rationale Dimension

In our system, DRL is augmented to better fit the construction project domain. As mentioned earlier, It incorporates three new objects—Project, Task, and Resource—and three new relations—IsAMotivationFor, IsATaskFor, and IsAnAvailableResource. The objects and relations introduced in mSIBYL, namely the Plan and Artifact objects, and the IsALikelyCauseFor and IsAGoodReasonFor relations, will not be discussed in this section, even though we do plan to incorporate them into the project management system. Rather, we will present each new object and each new relation separately, show the motivation behind their addition, and compare them to existing DRL and mDRL objects and relations.

4.3.1. New Objects

4.3.1.1 The Project Object

This object captures the current status of the schedule of the project. It is different from the Plan object of mSIBYL in that it describes a different type of object. Indeed, it captures a sequence of *tasks* that need to be performed during the project. The plan object, on the other hand, describes a sequence of *decisions* that need to be made during the project. Both sequences need not be the same. That is, decisions and tasks are not ordered the same way. For example, the decision problem of clearing the construction site can be resolved well after the building's plans have been finalized. On the other hand, the task of clearing the construction site must be completed before any construction of the building can start. The difference of sequences for decisions and tasks is our major motivation in distinguishing between the two types of objects. Note, however, that the decision making process can still be incorporated into the overall project's schedule, by creating task objects to represent the decision making activity and incorporating them into the project object. In that framework, such a task object would represent the task "Deciding on how to clear the site," for instance. One could argue that since the same information—i.e., the sequence of decisions to be made—is kept in two different objects, why not keep only one of them. The answer to that stems from the other distinction between both objects. The plan object is a first class DRL object, in the sense that it can be supported, denied, and argued about by other claims. This property allows the system to view recursive decision problems within a similar framework. For example, deciding on how to solve the problem, and deciding on a sequence to follow in solving problems, can both be represented by decision problem objects, even though the decision making process occurs at different levels. Figure 4.1 illustrates how these problems may be related in the overall structure. In that case, D37 is a result of choosing A1 to solve D0. Therefore, the argumentation capability is desirable. On the other hand, the Project object cannot be argued about. It is indirectly updated. When new decisions are made, or current ones are modified, tasks in the Project object are also created or modified, thus altering the overall project. The reason behind this choice is that information captured in the project object has to be formalized, as it will be used by external scheduling algorithms to provide the optimal project schedule. In that aspect, it is similar to the Artifact object defined in mSIBYL. The project object can therefore be thought of as both a result object that describes the current status of the project, and an interface object that may be attached to external modules to optimize the project's schedule.

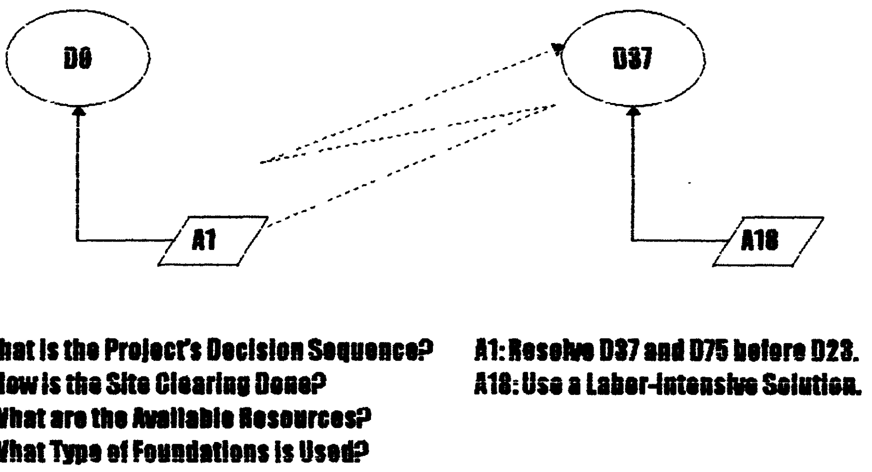

D0
D37
A1
A18
D0: What is the Project's Decision Sequence?
D23: How is the Site Clearing Done?
D37: What are the Available Resources?
D75: What Type of Foundations is Used?
A1: Resolve D37 and D75 before D23.
A18: Use a Labor-Intensive Solution.

Figure 4.1. Representing Recursive Decision Problems in DRL.

4.3.1.2 The Task Object

This object captures the scheduling attributes of the alternative that has been selected to resolve a particular problem. By scheduling attributes, we refer to the resources required to carry out the tasks described in that alternative, the constraints introduced by this alternative into the overall project, etc. The distinction between a decision and a task was mentioned previously, and it relates to the distinction between the Plan object and the Project object. For example, the decision problem "Clearing the Construction Site" refers to the action of planning for the site to be cleared, and deciding when and how it will take place. On the other hand, the task "Clearing the Construction Site" describes the actual action of clearing the site. A decision problem from the Plan object is discussed, and several alternatives are compared and evaluated, before the best solution is selected to carry out the decision. At that point, a task object corresponding to the selected alternative is created, capturing the requirements to be satisfied in order for it to be performed. The Task object is then incorporated into the Project object and related to other tasks. At that point, the schedule can be updated, depending on the availability of the resources. Like the Project object, the Task object cannot be part of the deliberation process. It can only be a consequence of it. Thus, the Task object is the basic transfer object between the system's rationale dimension and the scheduling dimension.

4.3.1.3 The Resource Object

This object is used to capture information regarding the resources that are available to a particular decision problem. Although the users may actually create a decision problem "What are the resources available to solve the site clearing problem?" for instance, these resources can be presented within any kind of rationale object in a deliberation process concerning the site clearing decision problem. That is, it may be mentioned in a claim, a goal, a procedure, a viewpoint, or even a decision problem. We thus decided not to constraint the users while making decisions by incorporating a resource object in the scheduling dimension, hence maintaining the system's flexibility, while providing it with a necessary object for project scheduling. The resource object can be created by the users at any point in the deliberation process. When that happens, the Resource object is incorporated into the Project object and related to other resources through the incompatibility field. At that point, the schedule can be updated. Like the Project object and the Task object, the Resource object is not part of the deliberation process. Rather, it is only a result of it. In fact, the Resource object is as an interface object between the system's rationale dimension and the external schedulers.

4.3.2. New Relations

4.3.2.1 The IsAMotivationFor(C, D) Relation

IsAMotivationFor(C, D) states that argument C is one of the reasons why decision problem D is considered at all. This relation is different from both the IsALikelyCauseFor relation and the IsAGoodReasonFor relation of mSIBYL. IsALikelyCauseFor involves the accuracy of the claims involved, stating that C2 is true whenever C1 is true. IsAMotivationFor does not deal with the accuracy of the objects it relates. It merely captures the rationale behind the decision to discuss a particular problem. Furthermore, IsAGoodReasonFor explains why an alternative has been selected to solve a decision problem, while IsAMotivationFor explains why that decision problem needs to be solved in the first place. Finally, IsAMotivationFor differs from IsASubDecisionOf in that it refers to decision problems motivated by claims, whereas the other one refers to decision problems motivated by other decision problems. In other words, IsAMotivationFor is a relation from the argument space to the issue space, whereas IsASubDecisionOf is a relation within the issue space. In the case of the Site Clearing problem, the claim that the equipment-intensive alternative achieves the problem's goals may result in the need to examine several suppliers. Thus, a new decision problem may be generated to choose suppliers, and that claim is linked to it through an IsAMotivationFor relation.

4.3.2.2 The IsATaskFor(T, DEC) Relation

The IsATaskFor(T, DEC) relation is the claim that task object T represents the selected alternative described in decided object DEC. Consider the topsoil removal problem, for instance. Suppose that there are two alternatives, a labor-intensive one and an equipment-intensive one. After extensive deliberation, the first alternative is selected. Following the original SIBYL protocol, a decided object is then created for that decision problem, referring to the labor-intensive alternative. At that point, the users create a "Remove topsoil" Task object, and transfer all the requirements necessary to accomplish this task (e.g., how much labor & equipment needed, what is the expected duration) into the Task object. An IsATaskFor relation is created to link this task object to the decided object. These requirements are not formalized during the deliberation process. There is no specific representation for the dialectical role of "requirements" in the rationale dimension. The reason for that is that requirements can be mentioned in different dialectical contexts. They could be part of a claim, or they could be included in a goal, for instance. Because they are not formalized, these properties and requirements cannot be transferred to the scheduling dimension systematically, hence the need for the IsATaskFor relation. Furthermore, unlike the Task object, this relation is first class, in that it can be rationalized and argued about. Since it inherently depends on human interpretation of the issues raised in the deliberation process, different people may translate an alternative and its requirements into different tasks and resources. This relation allows the users to question not only the degree to which the task object is a good 'translation' of the selected alternative, but also the content of the task object itself, or the types of resources that the task needs in order for it to be performed.

4.3.2.3 The IsAnAvailableResource(R, DP) Relation

The IsAnAvailableResource(R, DP) relation is the claim that resource object R represents a resource that is available to the users in solving decision problem DP. Consider again the topsoil removal problem, and the two previous alternatives, a labor-intensive one and an equipment-intensive one. At some point in the deliberation process, a participant may claim that these alternatives cannot be properly evaluated without knowing how much labor is available for the topsoil removal. Another user who knows the appropriate numbers may then give the answer in a claim object. After discussing the new information, the participants may agree on the exact amount of labor available during a given period of time. At that point, a Resource object R is created, capturing all the relevant information. An IsAnAvailableResource is also created, relating object R to the topsoil removal decision problem. Unlike the Resource object, this relation is first class, and it can be argued about. Similarly to the IsATaskFor relation, this relation allows the users to question not only the extent to which the resource object is available to the decision problem, but also the content of the resource object itself, or the availability and incompatibility constraints mentioned in the deliberation process.

4.4. Incorporating the Semantics Dimension

Like mSIBYL, the project management system we propose uses TS representation to capture the semantics of chunks of knowledge that participate in the deliberation process that takes place throughout the construction project. On the other hand, unlike mSIBYL, the system uses TS to capture sequences of changes that occur along both the rationale and the scheduling dimensions. More on the distinction between those two kinds of knowledge is presented in subsequent chapters. Indeed, we discuss how these dependencies are represented in chapter 5, and how they are managed in chapter 6. From the perspective of the TS framework, however, both kinds of knowledge are represented and manipulated similarly. In the following, we therefore review the four stages of operation of Pathfinder from the project management perspective, irrespective of this distinction. Mainly, we describe how semantics knowledge is captured and later retrieved throughout the deliberation process.

4.4.1. Parsing and Translation

During this stage, Pathfinder interprets text descriptions such as event references, background statements, meta-level statements, rules of interference, and rules of restatement, and transforms them into event traces. In the rationale management context, this corresponds to defining the semantics of the rationale objects. The user decides which part of the knowledge in a chunk needs to be formalized, and stores the corresponding descriptions in TS Text field of rationale objects. This is the description that is interpreted by pathfinder to generate appropriate event traces, hence capturing the "semantics" of the chunk. However, event traces that represent system objects and changes thereof in both the rationale and the scheduling dimensions are encoded within TS prior to the system's use. Indeed, the vocabulary of DRL is determined prior to the system's use, and no such predicate/ttribute semantics can thus be altered throughout the use of the system.

4.4.2. Exploratory Transformations

TS offers a set of transformations to describe events differently, therefore enabling the system to deal with analogies and abstractions encountered in the informal descriptions of the TS Text field. These transformations are equivalence, substitution, generalization, interval composition, attribute-object reification, event-attribute reification, and finally event-object reification. They are heavily used by several modules of the system to offer services across the semantic dimension, as described in mSIBYL. For example, the precedent manager module is able to bridge analogy and abstraction from one description to the other, and can therefore recognize precedent events that may be relevant to the current problem. The performance of the documentation manager is similarly improved, as the information retrieval is faster and more accurate. These transformations are also used by the system to propagate changes within the system knowledge, as will be discussed in chapters 5 and 6.

4.4.3. Association

During this stage, inference rules are heavily used to form partial matches between event traces in different clusters, and sequences of events are built. Indeed, partial matches describing causal links between events are ordered according to a set of heuristics and consistency constraints, and the best match is selected. A set of event traces linked by transformations and complete chaining or restatement matches is referred to as an *association structure.* It is this process that maps into the task of dependency management in the dependency management context. Indeed, if dependencies are viewed as causal effects, then partial matches recognize the different dependency paths, and the association structures generated capture the dependency relations between different objects, while maintaining consistency between both the decisions and the tasks.

4.4.4. Question Answering

At this point the user can access the association structure resulting from the previous phase for the information retrieval. The effect of potential changes in the structure can be followed through the several links in the association structure. Conversely, the causal links may be traversed in the opposite direction to explain events and access their causes. In the context of project management, the dependency manager uses the association structure to propagate the effects of change along any of the three dimensions. Similarly, the viewpoint manager is able to better answer what-if questions and to describe hypothetical and alternative scenarios. Also, the evaluation manager can better retrieve and explain the rationales behind decisions that have been previously made, hence providing additional relevant information for the alternative evaluation process. Finally, the schedule manager accesses the association structure to translate the effects of a new or modified decision into scheduling terms and update the project schedule appropriately.

4.5. Introducing the Scheduling dimension

In construction projects, time is often perceived as the most expensive and the scarcest resource. Time management is therefore of paramount importance. Thus, schedule generation, update, and control are essential tasks in the dependency management context. Indeed, if dependencies are viewed as causal effects, then these partial matches recognize the different dependency paths, and the association structures generated capture the dependency relations between different objects, maintaining consistency between both the decisions and the tasks in project management. By scheduling dimension, we refer to those capabilities of the system that manage project schedules. However, it was not our intention to produce new scheduling concepts, nor to substitute scheduling in project management by other activities. Rather, we believe that the system we propose can provide an *interface* to an external schedule algorithm, relating the tasks to be performed throughout a project with the rationales behind them, as they appear in the rationale dimension. In other words, when decisions are discussed and finally made, they are translated into tasks that have to be incorporated in the project schedule. Similarly, when external events affect a previously made decision, the schedule should be updated accordingly. Such interactions allow the users to better manage the project. In this section, we present the objects that capture scheduling issues in the system, and we then illustrate how this dimension is related to the rationale dimension. The interaction of the scheduling dimension with the semantics dimension is discussed in detail in chapter 5.

4.5.1. Objects in the Scheduling Dimension

The main objects that were introduced along the scheduling dimension are the *Project* object, the *Task* object, and the *Resource* object, as described in previous sections. In this section, we review these objects from the scheduling perspective, as shown in Figure 4.2.

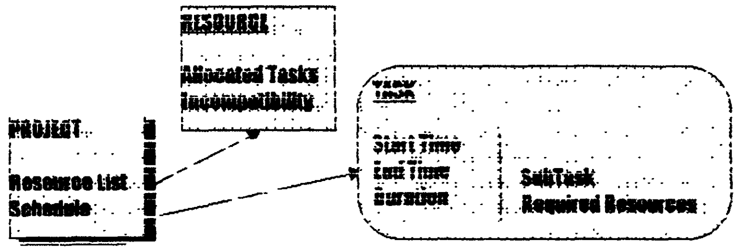


Figure 4.2. Objects of the Scheduling Dimension

4.5.1.1 The Project Object

The Project Object represents the sequence in which the tasks need to be performed in the project, and the resources available to the project, hence providing a good description of the current status of the project. It has a resource field RL, which is a list of Resource objects representing the resources available to the project. The other field of the Project object is the schedule field, consisting of a network of Task objects that are connected to one another through precedence relationships. The three basic precedence relations among Task object are end to start relations ES, start to start relations SS, and end to end relations EE. T1 ES T2 means that T2 cannot start before T1 ends. T1 SS T2 means that T2 cannot start before T1 starts. Finally, T1 EE T2 means that T2 cannot end before T1 ends. Other relationships can be generated from the basic ones by incorporating lead and/or lag elements. For instance, an end to start relationship between T1 and T2 with lag L states that T2 cannot start before a time L after T1 ends. Precedence relationships are incorporated into the system by the users, as a result of the deliberation process.

4.5.1.2 The Task Object

The task object captures those attributes of a task that are used by a scheduler. It is created when a decision problem has been resolved and an alternative has been selected. The task object has a start time field S, and end time field E, and a duration field D. These fields represent the time allocated to the task represented by the task project. However, since different schedulers may use different combinations of these fields, we chose to include all of them for portability reasons. Values of the fields are all determined by the external scheduler. The Task object also has a subtask field ST, consisting of a list of task objects that represent subtasks of the current one. Finally, the Task object includes a resource requirement field RR, which consists of a list of pairs. Each pair is made of a Resource object R representing a resource required by the task and a time duration RD, representing the duration for which that resource is needed. The values of both the subtasks and the resource requirements field are determined by the user as a result of the deliberation process.

4.5.1.3 The Resource Object

This object represents the resource characteristics that are used for allocation purposes. The Resource Object has an allocation field AT, which is a list of pairs. Each pair is made of a Task object that represents the task this resource is allocated to, and a time interval [ts, tf] during which the resource is allocated to the task. Both the availability field and the allocation field are determined by the scheduler. Finally, the resource object has an incompatibility field I, that consists of a list of pairs, each of which consisting of a Resource object that it is incompatible to this particular resource, and an attribute that may specify the type of incompatibility. Unlike the previous fields, both kinds of information captured by the incompatibility field of the Resource object are decided by the user as a result of deliberation processes.

4.5.2. Interacting with the Rationale Dimension

The scheduling dimension interacts explicitly with the rationale dimension through *IsATaskFor* relations and *IsAnAvailableResource* relations. While users can deliberate over both relations, their task-dimension arguments are not accessible by the rationale dimension, as they cannot be supported nor denied by other claims.

4.5.2.1 The IsATaskFor Relation

When a decision is resolved in the rationale dimension, a decided object is created to point to the alternative selected, and to capture the rationale behind this selection. At that point, the user creates a Task object referring to that alternative, and captures all the resources that have been mentioned to be required by the task. The Task object is then incorporated to the schedule field of the Project object. Similarly to IsAnAvailableResource, IsATaskFor will be used to transfer changes in the tasks from the rationale dimension into the project plan, as illustrated in Figure 4.3.

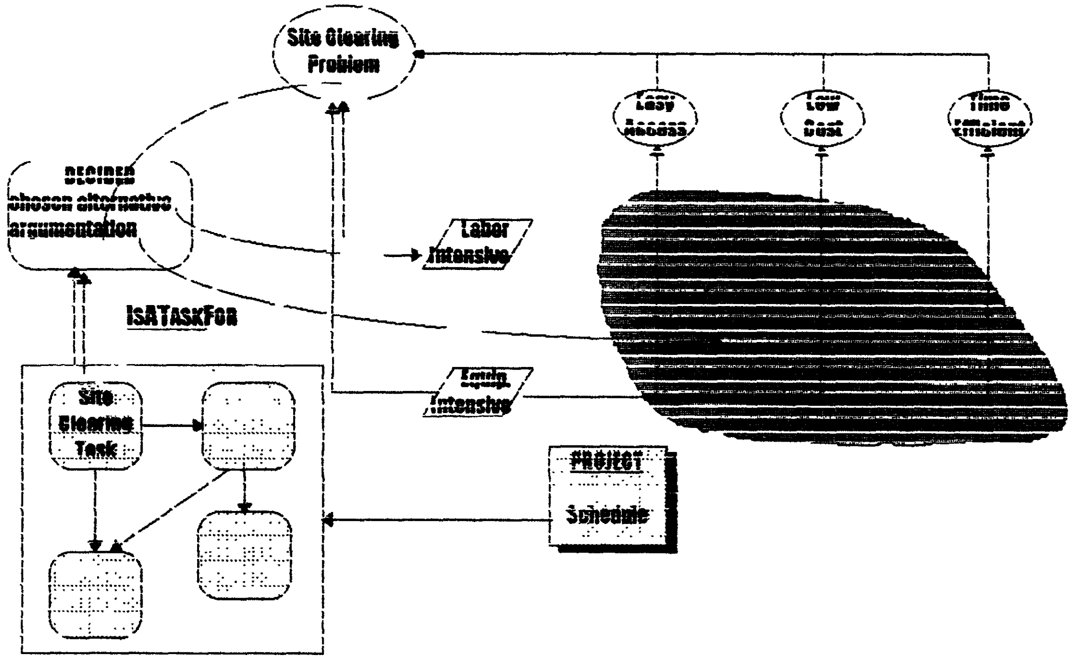


Figure 4.3. The IsATaskFor Relation in the Site Clearing Example.

4.5.2.2 The IsAnAvailableResource Relation

During a deliberation process, several participants are discussing and comparing alternative ways to solve a problem. Eventually, a resource that is available to the project will be mentioned as such in the discussion, probably within Claim objects, although possibly within other objects. Alternatively, the participants may explicitly list all the resources available to the project using Alternative objects related to Decision Problem object "What are the resources available to us," for instance. In either case, IsAnAvailableResource relates the rationale object containing the resource description to a Resource object in the task domain, including the incompatibility information. At that point, the resource list field of the project is updated accordingly. This relation will be used to transfer any change that may occur to the resources along the rationale dimension into the Project plan, so that the scheduler is appropriately updated. Figure 4.4 describes how the rationale dimension and the scheduling dimension are connected through the IsAnAvailableResource relation.

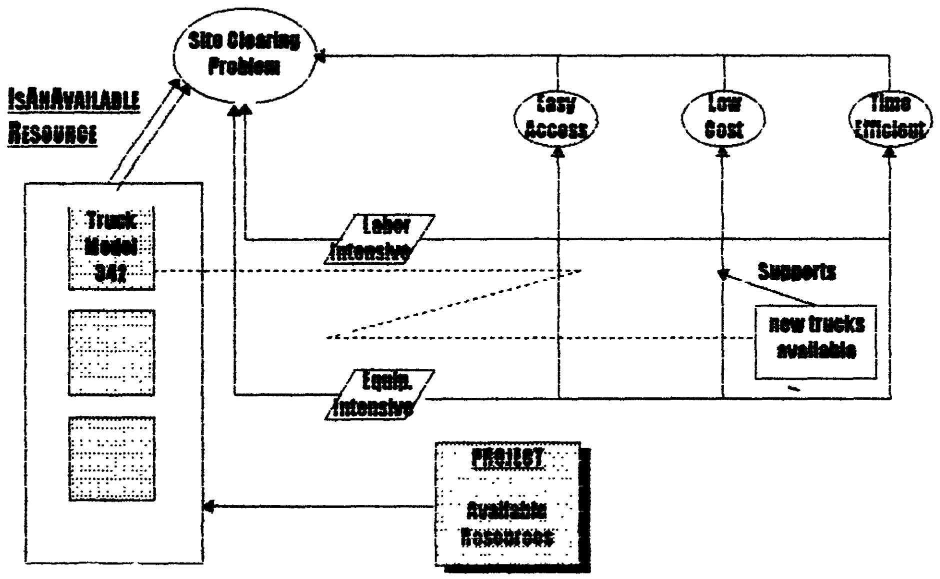


Figure 4.4. The IsAnAvailableResource Relation in the Site Clearing Example.

4.6. General Guidelines, Revisited

The project management system we propose provides its users with means to use past experience, examine different points of view of a same issue, keep track of dependencies across decisions and tasks, evaluate different alternatives, communicate among one another, document their contribution to the project, and maintain an efficient and up-to-date project schedule. It does so by following the guidelines listed in the first section of this chapter. Namely, it provides user interaction, system flexibility, knowledge reusability, incremental system formalization and growth, and three-dimensional information management. In the following, we review these guidelines, and describe how the project management system we propose in this thesis satisfies them.

4.6.1. User Interaction

The project management we propose is designed to operate in symbiosis with users. It therefore relies on heavy user interaction. This provides opportunities for the users to closely monitor and control the project, and to efficiently react to unexpected events. The communication management module of the system ensures that the contributions of several participants are quickly and accurately incorporated into the system. When an event is captured by the system, changes are propagated accordingly, using the rationale network, TS association structures, and the overall project schedule. Chapter 5 provides a more detailed and more complete description of the propagation of changes in the system.

4.6.2. Flexibility

Flexibility of the system is provided by the underlying technology used in the system. First, *interface flexibility* is allowed by the ability of TS to understand simple English descriptions of objects and events, and by the informal property of DRL. No extensive prior expertise is required from the users. Second, the ability of TS to manage abstractions and analogies through its set of transformations provides the system with *communication flexibility*. Terminology gaps between the participants are thus reduced. Finally, *structure* *flexibility* is guaranteed by both SIBYL and TS, as they are both able to capture and manage various amounts of information at different levels of detail. More specifically, the chunk data structure provides this flexibility, as users decide what information to encode.

4.6.3. Reusability

Reusability is a major motivation for the development of our system. Indeed the same situations occur and the same decisions are made over and over again from project to project. Like mSIBYL, our project management system provides a precedent management service to enable users to reuse information from previous experience. Precedents are retrieved from knowledge captured by both TS and SIBYL. The users are therefore able to access objects that have similar dialectical roles in the system, as well as objects that have the same functions and behavior. Furthermore, TS descriptions and transformations available to the system allow for efficient indexing of objects, given a set of relevant attributes of the event descriptions.

4.6.4. Incremental Formalization and Growth

Another property of our system is embodied in its incremental character. Incremental formalization happens along all three dimensions. In the semantics dimension, event traces are refined through exploratory transformation as finer descriptions of the events are provided. In the rationale dimension, relations such as IsASubGoalOf, IsASubDecisionOf, and IsAKindOf, all provide ways to refine the dialectical role of objects. Finally, incremental formalization takes place in the scheduling dimension through the subtask of Task objects. For instance, during one session, the project manager may include the *site clearing task* in the project schedule. At a later stage, this task can be broken down into two subtasks, the *topsoil removal task* and the *topsoil disposal task*. Furthermore, the system can also accommodate incremental growth of knowledge along the three dimensions. Indeed, users can incorporate new information in the decision problem (rationale dimension) as it becomes available. They may also decide to encode part of it in the system's knowledge base (semantics dimension). Finally, when a decision problem is resolved, the selected alternative is incorporated in the project's schedule through the Task object (scheduling dimension). Going back to the topsoil removal task, however, more efficient mechanical shovels may be available. Appropriate and relevant descriptions will then be added to the system during the deliberation process taking place between the participants. At this point, if a new alternative is chosen, then new tasks will be incorporated in the schedule.

4.6.5. Three-Dimensional Information Management

The system we propose enables users in the construction domain to discuss events and raise issues in three different dimensions. Indeed, it can capture and manage the semantics of objects mentioned in the discussion, the structure of the discussion and the deliberation process itself, and the up-to-date current project schedule. Figure 4.5 illustrates how the three dimensions are related to one another.

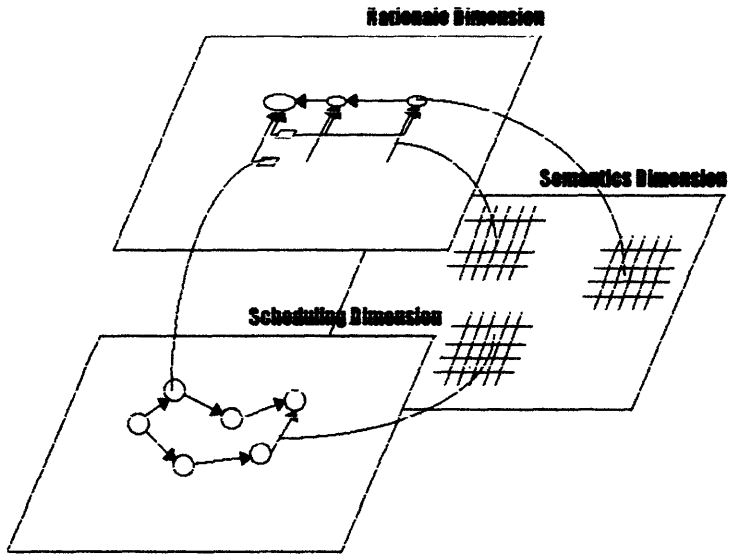


Figure 4.5. Three Dimensional Argumentation

The semantics dimension consists of descriptions of information objects cited in the discussion. In this context, the term *object* is used in its most general meaning, and refers to "anything that can be mentioned in the system." Trucks, goals, and tasks are all objects in this sense. Along the rationale dimension, the system views an entity according to its role in the deliberation process, regardless of the meaning of the object it contains. "How to clear the construction site" and "How to set up the foundations of the building" are represented by decision problem objects related to alternative objects representing several ways to solve them, and goal objects representing the criteria and requirements to be satisfied when solving the problem. They may have a similar dialectical structure even though they refer to totally different problems. Finally, the scheduling dimension represents scheduling issues. Tasks are ordered according to a set of precedence relationships, and resources are allocated to different tasks according to their availability. Along this dimension, the major concern is to produce a schedule that is both time- and cost-efficient.

CHAPTER 5

DEPENDENCIES IN PROJECT MANAGEMENT

In chapters 3 and 4, we have developed a representation model for capturing and managing information generated and used during a construction project. First, we introduced the different schemes of representation that we used. Then, we described how we modified these schemes to better fit our system. In this chapter, we explain how our representation model captures the dependency relationships that arise among the various pieces of knowledge generated in a project's lifetime. To that effect, we distinguish between two kinds of knowledge. Domain knowledge describes the objects that represent the information actually produced during construction projects, whereas system knowledge consists of the information used by the system for bookkeeping operations. In this chapter, we start by presenting the various distinctions between domain knowledge and system knowledge. Next, we describe the object types provided by the system, as introduced in the previous chapters. We mainly review the attributes of the various objects, as a basis for the following sections. We then illustrate how system dependencies—i.e., those dependencies that occur within the system knowledge—are captured and propagated using the transition space representation. In conclusion, we describe how the two dimensions interact with one another. Namely, we illustrate how TS is used to transfer dependencies between the rationale dimension and the scheduling dimension.

5.1. Domain Knowledge Vs. System Knowledge

Domain knowledge and system knowledge vary according to several parameters. More specifically, they represent (1) different kinds of information, generated by (2) different sources, at (3) different times, and for (4) different purposes.

5.1.1. Different Types of Information

Domain knowledge refers to the actual information generated and used by participants during the construction project. When deliberating on the best way to clear the construction site, for instance, one might be concerned with the load capacity of a particular truck (to evaluate the potential duration of the task), and its size (to deal with accessibility issues). System knowledge, on the other hand, is used by the system to capture the semantics of the objects used and the relations between them. The semantics of IsASubGoalOf relation is an example of system knowledge. It is independent of the issues raised by users in a construction project.

5.1.2. Different Sources of Information

By source of information, we refer to the originator of the said information. Thus, domain knowledge such as descriptions of the decision problem "Construction Site Clearing," is generated by the system's users as they deliberate on a way to perform the task. Similarly, the fact that accessibility is a goal to the same decision problem is also generated and evaluated by the users. On the other hand, the meaning of an IsASubGoalOf relation between a decision problem object and a goal object is part of the a priori knowledge of the system. For instance, the fact that increasing the plausibility of that relation may increase the importance of the goal to that decision problem is part of the system knowledge.

5.1.3. Different Times of Generation

Domain knowledge is not a closed set of descriptions. As more issues are discussed in more details by the users, more refined descriptions are incorporated into the system. Decision knowledge hence increases throughout the deliberation process. The more the system is used, the more the domain knowledge is enriched. On the other hand, system knowledge does not expand. The reason for it is that the language of the system does not expand. The object types, as well as the type hierarchy and the object attribute are pre-defined. Therefore, the semantics of the various relationships between different objects need be defined only once--during the implementation phase of the system—and they do not have to be altered during the system's use.

5.1.4. Different Purposes of Use

Domain knowledge is used mainly to provide an additional dimension of indexing to the rationale management system. Indeed it allows the system to provide the computational services of SIBYL along the semantics dimension. For instance, previously defined objects may be retrieved according to their functionality, and the effect of an event on others may be propagated based on their respective descriptions. The main purpose for system knowledge, on the other hand, is to provide basis for change propagation and constraint checking along both the rationale dimension and the task dimension, hence improving dependency management in both contexts.

5.2. The System Vocabulary, Revisited

In this section, we describe the attributes of the objects incorporated into the project management system. Namely, the meaning of the various object fields is fully explained. This presentation will be referred to in subsequent sections in order to describe how changes of attribute values are propagated throughout the system. We proceed with the object description following the type hierarchy structure of the system, as illustrated in Figure 5.1. Newly incorporated objects are depicted in **CAPS**.

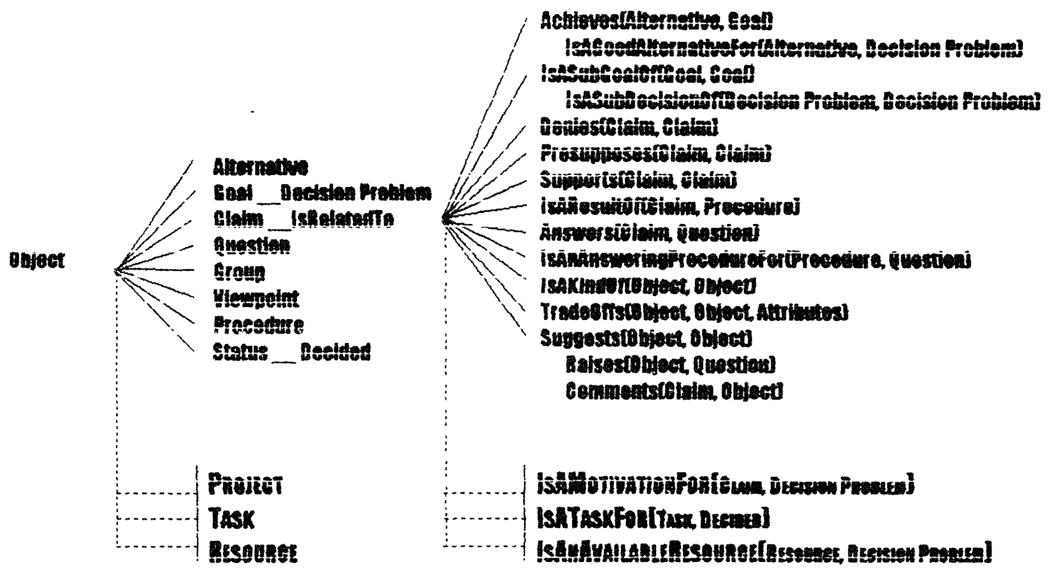

Object
Alternative
Goal __ Decision Problem
Claim __ IsRelatedTo
Question
Group
Viewpoint
Procedure
Status __ Decided
Project
Task
Resource
Achieves(Alternative, Goal)
IsAGoodAlternativeFor(Alternative, Decision Problem)
IsASubGoalOf(Goal, Goal)
IsASubDecisionOf(Decision Problem, Decision Problem)
Denies(Claim, Claim)
Presupposes(Claim, Claim)
Supports(Claim, Claim)
IsAResultOf(Claim, Procedure)
Answers(Claim, Question)
IsAnAnsweringProcedureFor(Procedure, Question)
IsAKindOf(Object, Object)
TradeOffs(Object, Object, Attributes)
Suggests(Object, Object)
Raises(Object, Question)
Comments(Claim, Object)
IsAMotivationFor(Claim, Decision Problem)
IsATaskFor(Task, Decision)
IsAnAvailableResource(Resource, Decision Problem)

Figure 5.1. The System's Type Hierarchy.

All the objects of Figure 5.1 have generic bookkeeping attributes that are used for operational purposes. Such attributes include the date and time of creation, the creator's identity, as well as the identity of potential receivers. We do not expand on this kind of attributes because the corresponding values are usually directly extracted from the environment of operation of the system. Rather, we concentrate on those attributes that differentiate one object from the others.

An **Alternative** object represents an option to consider in trying to resolve a decision problem. The STATUS attribute indicates whether the represented alternative is chosen to be selected option, rejected in favor of other alternatives, or whether no decision has been made yet.

A **Goal** object describes a desired state to be achieved by all the selected alternatives. It is therefore used as a criterion to evaluate various alternatives. The IMPORTANCE attribute of a Goal object represents the relevance of this goal to the overall problem, and therefore affects the extent to which an alternative is considered adequate.

A **Decision Problem** object is a subtype object of the Goal object. It represents the special goal of selecting the optimal alternative for the overall problem. The STATUS attribute reflects the current status of the problem, i.e., whether it has been resolved or not, and if not, for what reasons. The EVALUATION attribute is used to represent the performance of the selected alternative. It is a measure of the success of the action taken to satisfy all the criteria described by the Goal Object.

Claim objects are used to represent any statements. Such statements may consist of actual facts, arguments, assumptions, answers, or just comments. Distinctions between these kinds of statements are made dynamically, within the context of use of the claim objects. The PLAUSIBILITY attribute measures the probability, or likelihood, of a claim being true. Facts will therefore have relatively high plausibility measures. The DEGREE attribute measures the extent to which that claim is actually fulfilled. Thus, while the claim's plausibility measures its truthfulness, the degree tends to vary with the claim's evaluation, depending on the current status of the decision problem. The EVALUATION attribute provides a mean for the user to combine the plausibility and the degree of a claim into a single measure according to which claims may be compared.

An **IsRelatedTo** object represents the claim that two objects are related to each other. It is a supertype of other more specified relations, such as the Supports relation and the IsAMotivationFor relation, as shown in Figure 5.1. Since all relation objects are subtypes of the Claim object, they all have a DEGREE attribute that reflects the extent to which the objects are related, and a PLAUSIBILITY attribute that measures the uncertainty about that extent. The role of both the degree attribute and the plausibility attribute are the same for all subclasses of IsRelatedTo, i.e., for the pre-defined relations.

A **Question** object represents a request for additional information. It is used for deliberation purposes during the decision-making process. It does not directly participate in the alternative deliberation process. However, the answers provided to a question (either through claims or through procedures) will affect the alternative evaluation, as users may now use the new information to argue and deliberate. The DEADLINE attribute indicates the date before which the information is needed, and it may be used to generate periodic reminders, until a response is obtained.

A **Group** object is used to gather together objects of the same type that have something in common, or that are somehow related to one another. The MEMBERS attribute identifies the object to be grouped. The MEMBER RELATIONSHIPS attribute, on the other hand, describes how they are to be grouped, e.g., whether they are conjunctive, disjunctive, or mutually exclusive.

Viewpoint objects are used to capture a state of the deliberation process. For example, a viewpoint can represent states that have the same set of objects with different attributes, states that have subset/superset objects of objects in other viewpoints, or states that are historically related. The ELEMENTS attribute point to all objects that are part of that viewpoint. The VIEW RELATIONS attribute relates a Viewpoint object to other Viewpoint objects.

A **Procedure** object represents a procedure that is used to answer a question. The STEPS attribute describes the procedure in more details, and it may consist of either other procedures or informal descriptions. It can be useful when an answer needs to be repeatedly evaluated, or when a similar question needs to be answered in a slightly different context.

A **Status** object provides information about the state of an object. Most objects have status fields, but sometimes complex information describing that status needs to be recorded as well. For instance, when a decision problem is decided, information regarding the person who made the decision, the time of the decision, and the rationale behind the decision needs to be captured as well.

The **Decided** object is a special template of the status object. When an alternative is chosen to resolve a decision problem, the status field of the Alternative object takes on the *chosen* value, the status field of the Decision Problem takes on the *decided* value, and a Decided object is created. The CHOSEN ALTERNATIVE attribute of this object thus points to the selected Alternative object, while the RATIONALE attribute represents the reason for such a selection.

A **Project** object represents the current status of the project's schedule, including the current states of the various tasks to be performed during the project. The RESOURCE attribute is a list of Resource objects describing the resources available to the project. The SCHEDULE attribute consists of a network of Task objects, connected to one another through precedence relationships.

A **Resource** object is used mainly for allocation purposes. It has an AVAILABILITY attribute that states whether the resource is currently available or not. An ALLOCATION attribute consisting of a list of pairs is also used to represent the tasks to which this resource is allocated and the time interval during which it is. Finally, an INCOMPATIBILITY attribute is included into the resource object to represent the list of incompatible resources, along with the type of incompatibility.

A **Task** object captures the attributes of a task used by a scheduler. It has a START TIME attribute, an END TIME attribute, and a DURATION attribute. The task object also has a SUBTASK attribute, in addition to a RESOURCE REQUIREMENT attribute to represent the list of resources required by the task and the respective time duration for which those resources are needed.

5.3. Assumptions

One of the main motivations behind the design of the project management system is its generic aspect. Users are able to use the system in different domains—such as the construction domain or the software development domain—and different contexts—such as the design of the building's layout or the scheduling of tasks. Thus, in representing the *semantics* for the various system's object types, we do not want to specify a particular *meaning* and *use* of these objects. We rather concentrate on a more generic definition of the various dependencies that arise among objects, independently of the system's domain and context of application. Hence, while there is one system specification of dependencies related to the IsAMotivationFor relation, for example, its use does indeed vary from the construction domain to the software development domain, and from architectural tasks to scheduling tasks.

With that in mind, we have designed our system to capture the dependencies among the system's objects without affecting the user's understanding and interpretation of their use. To do so, we have chosen to leave the semantics of certain attributes unspecified, so that the users can define them themselves, according to the current domain and context of use of the system. Therefore, dependency relations involving unspecified quantities cannot be completely defined. The change propagation is thus affected by the semi-formal aspect of the system, where some values are intentionally left unspecified. In this section, we present the assumptions we made in order to capture dependency relations among objects. We distinguish between formalization, technology, and representation assumptions. Formalization assumptions refer to design decisions made to have a generic and flexible project management system. Technology assumptions relate to technological limitations of the system. Representation assumptions are assumptions made to represent generic knowledge such as the system's type hierarchy, and object creation and deletion.

5.3.1. Formalization Assumptions

The generic property of the project management system we propose imposes some constraints on the representation of dependency relations in the system. We would like to specify the semantics of the various relations as long as we do not restrict their context of use. Therefore, there will be cases where system quantities are not completely defined, and where users will have the possibility to assign a meaning to these quantities according to their context of use. This situation may occur in various situations of the system. The semantics of certain object attributes, the semantics of the values of certain object, and the semantics of certain relationships between objects are all left unspecified.

5.3.1.1 Unspecified Semantics of Attributes Types

In some cases, the semantics of attributes are left unspecified. Although they are defined in our system, the kind of information they represent is not completely identified. For instance, the EVALUATION attribute of the achieves relation is used to evaluate how well the alternative achieves the goal. However, we have not defined the type of this attribute. Rather, it is up to the users to choose a quantity that is meaningful to them. They may decide to have either quantitative (e.g., Interval [0, 1]) or qualitative (e.g., High, Medium, Low) attributes. Therefore, dependencies involving this attribute cannot be specific. Although we can still argue about the value of EVALUATION attribute increasing or decreasing, we cannot describe it any further.

5.3.1.2 Unspecified Semantics of Attribute Values

In other cases, the semantics of the *value* of an attribute may not be determined, even though its type is well defined. For example, the DEGREE attribute of a relation object states the extent to which that relation connects the other two objects. Although we have defined this attribute to be a measure within the unit interval [0, 1], we didn't assign a definite meaning to the various values. In some context, the users may be satisfied with a 0.5 value, while in other cases, they may require a higher threshold value. Furthermore, they may decide to give different meaning to degree attributes of different relation objects, based on the context of their use. Once again, for such cases, dependency constraints can only refer to the general trend of the changes occurring to such attributes.

5.3.1.3 Unspecified Semantics of Attribute Relationships

Finally, we may leave the semantics of certain attributes relative to other attributes unspecified. In this case, even though both the types and the values of the attributes are fully determined, the semantics of how the attributes are related to each other is not specified. Assuming that both the type and the value of the IMPORTANCE attribute of the goal object are well defined, the relationship between the IMPORTANCE attributes of two goal objects may still be ambiguous. For example, if G1 is a subgoal of G2, and both are assigned values I1 and I2 for their respective IMPORTANCE attribute, then it is still not clear whether the measures of I1 and I2 are absolute measures, or whether I1 is relative to I2. Therefore, we cannot know a priori what happens to I1 when I2 changes. Hence, this dependency cannot be incorporated in the system.

5.3.2. Technology Assumptions

The technology underlying our system also calls for constraints on the representation of dependency relations. Indeed, the project management system inherits the limitations that are inherent to the technology it relies on. In the previous section, we presented assumptions inherited from SIBYL through the rationale dimension. In this section, we examine assumptions that are transmitted to the system through the semantics dimension. As seen in chapter 3, TS provides a qualitative representation of causal effects. However, it does not incorporate quantitative descriptions. In the following, we describe how this affects the representation of dependency relationships in the project management system.

5.3.2.1 Representation of Quantity

Since the transition space representation allows for qualitative descriptions only, quantities cannot be accessed by the TS mechanisms. Therefore, TS cannot capture dependencies among numerical attributes, and changes of these attributes cannot be propagated in the system. For instance, if more labor is made available to perform a particular task, the system may guess that the duration of that task will thus be reduced, hence affecting the evaluation of the various alternative currently under consideration. However, the system is incapable of knowing the exact amount of such improvement. Although the mathematical relationship may have been explicitly mentioned by the participants, the current system cannot use it to propagate a change. It can only guess the direction of the resulting change.

5.3.2.2 Representation of Proportionality

For the same reason as the one described above, TS does not understand the concept of proportionality. However, we can get around it to some extent. Indeed, we propose to define TS predicates PROPORTIONAL+ and PROPORTIONAL- to represent positive proportionality and inverse proportionality, respectively. These concepts are defined by use of the TS equivalence transformation, that allows TS to substitute one description with another one that has equivalent meaning. Thus, PROPORTIONAL+(att1, att2) is true when the ratio between att1 and att2 does not change after both attributes att1 and att2 do change, as shown in Figure 5.2. PROPORTIONAL-(att1, att2) is defined similarly.

	t1	t2	t3
[Object1, Backgnd]	**Att1**	Δ	$\cancel{D}$
[Object2, Backgnd]	**Att2**	Δ	$\cancel{D}$
[Object1, Object2]	**RatioAtt1Att2**	$\cancel{\Delta}$	$\cancel{\Delta}$
[RatioAtt1Att2, Backgnd]	**Sign**	$\cancel{\Delta}$	$\cancel{\Delta}$

Attributes Att1 and Att2 are proportional.

	t21	t22	t23
[Object1, Backgnd]	**Att1**	Δ	$\cancel{D}$
[Object2, Backgnd]	**Att2**	Δ	$\cancel{D}$
[Object1, Object2]	**RatioAtt1Att2**	$\cancel{\Delta}$	$\cancel{\Delta}$
[RatioAtt1Att2, Backgnd]	**Sign**	$\cancel{\Delta}$	Δ

Attributes Att1 and Att2 are inversely proportional.

Figure 5.2. Proportionality Predicates.

5.3.3. Representation Assumptions

In this section, we concentrate on presenting assumptions we made concerning the representation and management of dependency relations in the system. These assumptions are not due to external decisions as in the previous two cases. Rather, they deal with representation issues inherent to the project management system itself. Namely, we illustrate how the system's type hierarchy is captured by TS. Then, we describe how object creation is translated in the transition space framework. Finally, we explain the meaning of deleting an object, and the context in which this operation is performed. Note that at this point, we do not discuss the dependency relationships associated with this knowledge. Rather, this information should be taken as a basis for the following sections, where dependency relations are described in detail.

5.3.3.1 Representing the Type Hierarchy

The transition space framework provides a set of exploratory transformations to manipulate the information captured in the knowledge base. We use the GENERALIZATION transformation described in chapter 3 to capture the system's type hierarchy in TS. As we will see in subsequent sections, some dependency relations are common to several types of objects. It is therefore more efficient to represent these dependencies for a general case, and let the TS mechanisms adapt them to each specific situation. We therefore use TS to relate the system's object types to one another, as described in Figure 5.1. This is done using the IS-A TS predicate. For instance, Figure 5.3 states that any ACHIEVES relation connecting ALTERNATIVE object A to GOAL object G can be generalized into an ISRELATEDTO object relating A to G.

		t1 t2
[Alt22, Alternative]	**IsA**	$\not{D}$
[Goal123, Goal]	**IsA**	$\not{D}$
[Alt22, Goal123]	**Achieves**	A

Alt22 achieves Goal123.

		t21 t22
[Alt22, Alternative]	**IsA**	$\not{D}$
[Goal23, Goal]	**IsA**	$\not{D}$
[Alt22, Goal123]	**IsRelatedTo**	A

Alt22 is related to Goal123.

Figure 5.3. Type Hierarchy Representation.

5.3.3.2 Representing Object Creation

When the user creates an object along the rationale dimension or the scheduling dimension, a corresponding event trace is generated in TS to capture the existence of that object in the system's knowledge base. Note that this is different from representing the object's semantics dimension. Rather, we are here concerned with representing the context field value—or the rationale role—into TS. For example, when the user creates a GOAL object, a chunk is created in the system's knowledge base. It is given a unique Id O1, and its context field becomes GOAL. At that point, statement IS-A(O1, GOAL) is incorporated into TS. This operation is performed before the content field is even initialized.

5.3.3.3 Representing Object Deletion

In the following section, we refer to object deletion in the rationale context only. That is, when an object is deleted, it is merely removed from the current decision problem structure it was initially part of. We do not mean that it is completely removed by the system. Its representation in TS still exists. More specifically, the chunk that refers to it still exists, and the IS-A relation generated when it was created still exists. That object could still be accessed and used by the system's various computation managers such as the precedent manager, from instance. More on this topic will be presented in chapter 6. The deletion takes place at a higher level. In fact, we are dealing with the removal of the object from the rationale structure, and not with the disposal of the object from the system's knowledge base. At this point, we are not concerned with this latter type of deletion. In fact, we choose to push it to the implementation stage of the system.

5.4. Representing Rationale Dependencies

In this section, we discuss the dependencies that arise along the rationale dimension, and describe how they are represented in the transition space framework. The various mechanisms of Transition Space are used by the system to propagate changes along these dependency relations both within a decision problem and across decision problems. In this section, we examine dependencies that arise in a statement of the form *IsRelatedTo(O1, O2)*. We assume that an object refers to any object of the rationale dimension, and IsRelatedTo is either the IsRelatedTo object itself, or a subtype of it. First, we discuss the effect of changes of attributes of O1 and/or O2, and those of attributes of IsRelatedTo. We then present the effect of creation and deletion of O1, O2, and IsRelatedTo in a given decision problem. We refer to this type of change as object changes. Note that when an object is created, it must be related to other objects in at least one decision problem. However, for flexibility purposes, we do not want to impose an order on the creation of objects (first the object then the relation or vice versa). Thus, the system must account for both possibilities. As it turns out, object changes do rely extensively on attribute changes.

Rather than describing every case for every object of the system, we present an example scenario for each one of the four situations just mentioned. Figure 5.4 represents the general schema of a decision problem related to an alternative and a goal through IsAnAlternativeFor, IsASubGoalOf, and Achieves relations. We use it to illustrate various cases of rationale dependencies, presenting the appropriate TS descriptions for each one.

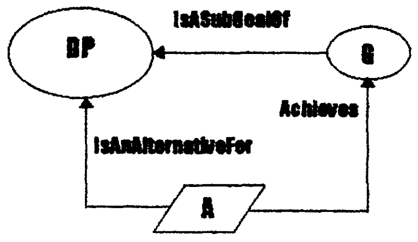


Figure 5.4. Illustrating Rationale Dependencies.

5.4.1. Effects of Attribute Changes of O1 and/or O2

Given the rationale structure of figure 5.4, if goal G becomes more (or less) important in resolving decision problem DP, then alternative A may need to be evaluated differently. Note that in this case, the evaluation attribute—not the degree attribute—is affected. Indeed, the degree attribute is not affected by the importance of the goal. The same alternative still achieves the goal to the same extent. However, the final evaluation measure of the alternative may vary, because a different evaluation procedure may now be required. Figure 5.5 illustrates the transition space representation of this fact. Note that in this event trace, IsASubGoalOf is represented as an attribute, while Achieves is represented as an object. In fact, they both represent DRL relations. An attribute-event reification was performed on the event trace representing the Achieves relation as an attribute.

t66 t67 t68

[G, DP]	**IsASubGoalOf**	D	*
[G, DP]	**Importance**	Δ	*
[AchievesAG, DP]	**ParticipatesIn**	D	*
[AchievesAG, DP]	**Evaluation**	*	Δ

If the importance of a goal changes, then so does the evaluation of the achieves relation.

Figure 5.5. Attribute Change of Goal Object.

5.4.2. Effects of Attribute Changes of IsRelatedTo(O1, O2)

Consider the same decision problem as the one described above. If alternative A achieves goal G to a lower extent, then alternative A will measure differently, with respect both to goal G and decision problem DP. In the rationale dimension terminology, this would translate to a lower evaluation measure of the Achieves relation—since the evaluation attribute of a rationale relation depends on the degree attribute—as well as a lower degree measure of the IsAnAlternativeFor relation, as represented in Figure 5.6.

		t66	t67	t68
[G, DP]	**IsASubGoalOf**	D	*	
[AchievesAG, DP]	**ParticipatesIn**	D	*	
[AchievesAG, DP]	**Degree**	–	*	
[AchievesAG, DP]	**Evaluation**	*	Δ	
[IsAGdAltA, DP]	**ParticipatesIn**	D	*	
[IsAGdAltA, DP]	**Degree**	*	–	

If the degree of an achieves relation decreases, then so does the evaluation of the achieves relation and the degree of the IsAGoodAlternativeFor relation.

Figure 5.6. Attribute Change of Achieves(A, G) Object.

5.4.3. Effects of Object Creation

Once again, we refer to the rationale structure of Figure 5.2. In this section, we are concerned with the meaning of creating objects O1, O2, and/or IsRelatedTo. If the user creates a claim object C, for example, he/she must connect it to at least one object participating in a decision problem through some IsRelatedTo relation, such as the IsAMotivationFor relation, for instance. Similarly, a binary rationale relation such as any IsRelatedTo relation cannot exist without either one of the objects it is supposed to relate. However, even when both the new object and the new relation are created the system is still *unaware* of their existence *in the context current of the decision problem*. Although they are now part of the decision problem, they have not actually participated in the structure yet. They will only do so when the user initializes their attributes. At that point, change propagation will take place in the rationale structure as described in the previous section. That is, it becomes a situation of attribute change propagation. No transition space rule is therefore needed to handle object creation. Figure 5.7 represents the rationale structure of Figure 5.4 after claim C is related to decision problem DP through the IsAMotivationFor relation.

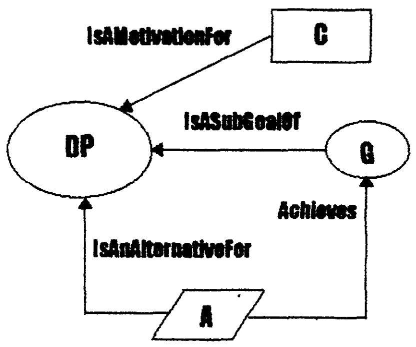


Figure 5.7. Creation of claim object C and Relation Object IsAMotivationFor.

5.4.4. Effects of Object Deletion

Object deletion is also based on the TS mechanisms of attribute changes of objects. The only additional required rules are those needed to ensure that no dangling objects remain after the deletion operation. That is, when a rationale object—a relation or otherwise—is deleted, the system must propagate this change locally, to the objects that are immediately related to it, and check whether they remain accessible in the decision problem structure after the deletion operation is performed. If so, then the operation is completed. If not, then the object that is now inaccessible as a result of the deletion is also removed from the structure. For instance, if claim C of Figure 5.5 is deleted from the structure, then the adjoining—binary—IsAMotivationFor relation is undefined. Therefore, it must also be deleted from the structure. At this point, the system checks if decision problem DP is accessible through another relation, and recognizes the IsASubDecisionOf relation connecting it to decision problem DP1. The deletion operation is therefore not propagated any further, resulting in the rationale structure of Figure 5.2. If, on the other hand, relation IsAnAlternativeFor is deleted, then alternative A is no longer relevant to decision problem DP, and must therefore be deleted, together with the Achieves relation connecting it to goal G. Figure 5.8 illustrates a general case of the local deletion propagation process.

	t66	t67	t68
[Object1, DP1]	**ParticipatesIn**	D	$\not{A}$
[Object2, DP2]	**ParticipatesIn**	$\not{D}$	$\not{D}$
[Object3, DP3]	**ParticipatesIn**	$\not{D}$	$\not{D}$
[Object1, Object2]	**IsRelatedTo**	$\not{D}$	D
[Object3, Object1]	**IsRelatedTo**	$\not{D}$	D

If object1 is removed from the decision problem DP, then all the IsRelatedTo relation linked to it are deleted.

Figure 5.8. Local Object Deletion Propagation.

The issue that remains to be resolved now refers to what happens at the end of the local propagation. When the Achieves relation is deleted from the rationale structure, it reflects the fact that alternative A no longer participates in satisfying goal G. In terms of the rationale dimension, this is equivalent to reducing the degree measure of the Achieves relation to zero, and the Transition Space rule of Figure 5.4 is therefore in effect. Figure 5.9 illustrates the rule that describes this equivalence relationship.

	t63	t64	t65
[IsRelatedToO1O2, DP]	**ParticipatesIn**	D	$\not{A}$
[IsRelatedToO1O2, DP]	**Degree**	*	–

An IsRelatedTo relation deleted from decision problem DP is equivalent to the degree of IsRelatedTo decreasing.

Figure 5.9. Equivalence Rule for Object Deletion.

5.5. Representing Scheduling Dependencies

In this section, we present the dependencies that occur in the scheduling dimension, and describe how they are represented in the transition space framework. Just like for rationale dependencies, the various mechanisms of Transition Space propagate changes along the scheduling dependency relations both within an object and across objects. In this section, we first describe the various precedence relationships that relate task objects together. Then, we examine dependencies that arise between different attributes within a single task object. Next, we illustrate change propagation from task object T1 to task object T2. Finally, we describe dependencies relating task objects to resource objects.

Like in the previous section, we present an example scenario for each one of the situations just described. The scheduling structure of Figure 5.10 represents the structure of a precedence graph relating task objects T1, T2, and T3 through an end-to-start precedence relationship and through the subtask field, along with resource object R. In the following sections, we illustrate scheduling dependencies based on that structure.

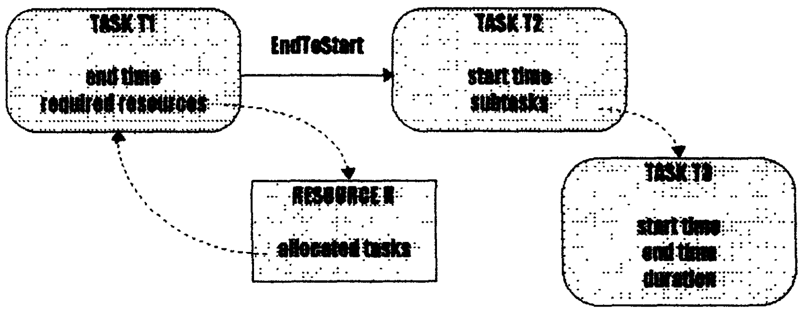


Figure 5.10. Illustrating Scheduling Dependencies.

5.5.1. Precedence Relationships

In the scheduling dimension, task objects are related to one another via precedence relationships. The resulting precedence graph is then used by a scheduler algorithm to generate a project schedule based on the constraints represented by these precedence relations. The selection of an optimal scheduler algorithm is beyond the scope of our research. Rather, we intend for the scheduling dimension to act as an interface between our project management system and any external scheduler.

The three basic such relations are *end-to-start* relations ES, *start-to-start* relations SS, and *end-to-end* relations EE. T1 ES T2 means that T2 cannot start before T1 ends. T1 SS T2 means that T2 cannot start before T1 starts. Finally, T1 EE T2 cannot end before T1 ends. *Lead* and/or *lag* elements can be incorporated to generate other precedence relations. For instance, an end-to-start relationship between T1 and T2 with lag L states that T2 cannot start before a time L after T1 ends.

The precedence relations are incorporated into the scheduling dimension by the users. They reflect the time constraints between the various tasks generated from the argumentation process captured by the rationale dimension. On the other hand, constraints due to resource allocation conflicts are not explicitly represented in the system. Rather, when the scheduler is activated, a consistency check is performed at the resource object level, identifying any potential resource allocation inconsistency, such as too many tasks using the same resource simultaneously.

5.5.2. Dependencies within Same Task

As mentioned in previous chapters, a task object has attributes *start time* S, *end time* E, and *duration* D, representing the time allocated for the task to complete. It also has attribute *subtask* ST, a list of tasks that are subtasks of the current one, and attribute *resource requirement* RR, a list of pairs of resource object R and time duration D, representing the resource needed by the task and the time for which it is needed.

These task object attributes are related to one another. Therefore, any change that occurs to one attribute may affect the others. For example, the first three attributes are all constrained by the relation S + D = E. Therefore, if the value of D increases, then either the value of S must decrease by the same amount, or the value of E must increase by the same amount, or the net effect of the change of E - S corresponds to this amount. Although TS cannot understand quantitative values, it represents this constraint only partially, capturing the direction of the changes, without the amounts. In this example, it may represent the fact that increasing S increases E and/or decreases D, like in Figure 5.11.

t16 t17 t18

[T1, Task]	**IsA**	$\cancel{D}$	$\cancel{D}$
[T1, Backgnd]	**StartTime**	+	*
[T1, Backgnd]	**EndTime**	*	+
[T1, Backgnd]	**Duration**	*	-

If the start time increases, then so does the duration.

Figure 5.11. Change Propagation within a Task Object.

5.5.3. Dependencies across Tasks

Attribute changes are also propagated from one task object to another through both the precedence relations and the subtask attributes. For example, an end-to-start relationship between tasks T1 and T2 implies the constraint T1.E <= T2.S. Therefore, if the value of T1.E increases, then the value of T2.S may need to be increased as well. Conversely, if the value of T2.S decreases, then the value of T1.E may need to be decreased. Again, since TS does not represent quantitative values, a conservative approach is chosen. The system assumes no gap between the two values, i.e., T1.E = T2.S, so that every time one value is changed it gets propagated to the other value, as shown in Figure 5.12.

		t16	t17	t18
[T1, Task]	**IsA**		D̸	D̸
[T2, Task]	**IsA**		D̸	D̸
[T1, T2]	**EndToStart**		D̸	D̸
[T1, Backgnd]	**EndTime**		+	*
[T2, Backgnd]	**StartTime**		*	!

Increasing end time of T1 increases start time of T2.

Figure 5.12. Change Propagation along Precedence Relations.

Furthermore, the subtask attribute ST of a task object T is a list of task objects T' that are subtasks of task T. In other words, T consists of its subtasks T'. It is only completed after all its subtasks in ST are completed. This attribute therefore implies that for every task T' from attribute ST of task T, we have T.E >= T'.E. Thus increasing the value of T'.E for one of the tasks in attribute ST of task T will cause the value of T.E to increase as well. Again, the inequality is interpreted by the system conservatively, as shown in Figure 5.13.

		t16	t17	t18
[T2, Task]	**IsA**		D̸	D̸
[T3, Task]	**IsA**		D̸	D̸
[T3, T2]	**IsASubTaskOf**		D̸	D̸
[T3, Backgnd]	**EndTime**		+	*
[T2, Backgnd]	**EndTime**		*	!

Increasing end time of T3 increases end time of T2.

Figure 5.13. Change Propagation along Subtask Attributes.

5.5.4. Dependencies between Task and Resource

Dependencies can also be propagated between task objects and resource objects, describing the allocation time of resources to tasks. Figure 5.8 illustrates a resource R that is allocated to a task T. The time constraint is represented by equation T.RR.(R, D).D <= length{R.AT.(T, [ts, te]).[ts, te]}. The left hand side (LHS) represents the duration time for which task T needs resource R. The right hand side (RHS) represents the time interval [ts, te] for which resource R is actually allocated to task T. Therefore, taking the conservative viewpoint again, increasing the value of D requires increasing te and/or decreasing ts. Figure 5.14 reflects the transition space representation of the constraint.

t16 t17 t18

[T1, Task]	**IsA**	$\not D$	$\not D$
[R, Resource]	**IsA**	$\not D$	$\not D$
[T1, R]	**IsAllocatedTo**	$\not D$	$\not D$
[R, T1]	**IsNeededBy**	$\not D$	$\not D$
[R, T1]	**RequiredDuration**	!	*
[T1, R]	**IntervalOfUse**	+	+

Increasing a resource's required time increases the resource's allocated time interval.

Figure 5.14 Change Propagation from Task to Resource

Similarly, attribute changes can be propagated in the other direction. Changes in the resource object also affect the task objects connected to it through the allocation relation. Considering the same example of Figure 5.10, if ts increases, stating that the resource is available at a later time, then the starting time S of the corresponding task T may also need to increase, hence delaying the task, as illustrated in Figure 5.15.

t16 t17 t18

[T1, Task]	**IsA**	$\not D$	$\not D$
[R, Resource]	**IsA**	$\not D$	$\not D$
[AllocatedToRT1, Backgnd]	**StartTime**	+	*
[T1, Backgnd]	**StartTime**	*	+

Increasing a resource's allocation start time increases the task's start time.

Figure 5.15. Change Propagation from Resource to Task.

5.5.5. Creation/Deletion of Scheduling Objects

Scheduling changes occur when a task object or a resource object is created into or deleted from the project object. The effects of these changes are propagated in the SCHEDULE attribute and the RESOURCE attribute of the project object. This propagation is performed the same way as in the rationale dimension. For example, creating a task object does not affect the project object until it is properly incorporated into its SCHEDULE attribute and related to other task objects through precedence relations. Similarly, when a resource object is deleted from the scheduling dimension, its attributes are emptied, and all the tasks to which the resource had been allocated are updated accordingly. The RESOURCE attribute of the project object is also updated by removing the resource from the list of available resources.

CHAPTER 6

COMPUTATIONAL TOOLS IN PROJECT MANAGEMENT

In the previous chapter, we saw how dependencies are represented in our framework for a construction project management system. In doing so, we chose to mainly focus on system dependencies. Namely, we showed how rationale predicates and schedule attributes are described in the transition space representation. Domain dependencies were studied by Gary Borchardt in his work on the development of an impact language based on TS as well. In this chapter, we discuss how system dependencies are used to better support the various computational modules incorporated in the system to deal with dependency management, precedent management, viewpoint management, evaluation management, and scheduling management in the system. The structure of these computational tools is inherited from both SIBYL and mSIBYL. In this chapter, we examine the computational tools from those two perspectives first. We then explain how system dependency representation improves their performance. Throughout our presentation, we use cases from the construction site clearing problem to better illustrate how users interact with the project management system, and how the various modules assist them in solving problems.

6.1. Introduction

A major motivation behind the development of information management systems in general is their potential to provide decision makers with tools that would allow them to better (1) react to changes of external events, (2) learn from past experience of other decision makers, (3) analyze issues from different perspectives, (4) examine and evaluate various solutions to a problem according to various criteria, (5) maintain an up-to-date project schedule, (6) document their contributions to the deliberation process and access others', and (7) communicate and interact with other—interested—decision makers. We have therefore incorporated computational tools in our project management system for that purpose. Namely, we have included (1) a dependency management module to keep track of changing events and check for constraint inconsistencies, (2) a precedent management module to access and retrieve relevant information from other decision problems, (3) a viewpoint management module to explore a given issue from various independent perspectives, (4) an evaluation management module to capture, manage, and display the status of the various objects participating in the deliberation process, and (5) a scheduling management module to closely monitor the project's schedule and thus better control its progress. Although we do intend to incorporate a documentation management module to efficiently capture new information and interact with existing information and a communication management module to relate the right people together, reducing unnecessary communication activity, we will not discuss them in this chapter. We postpone the communication issues to the implementation stage. On the other hand, documentation issues were thoroughly examined by Lukas Ruecker. We refer the reader to his work on a design documentation and management system for mechanical artifacts.

In our system, just like in SIBYL, the distinctions between the various modules is purely logical. The modules are classified according to the type of questions they help users answer. We are not concerned about the actual implementation of the modules.

Therefore, we are by no means implying a clear-cut distinction in the structure of the various modules. In fact, most often than not, one module will actually make use of the services of another one, as we describe in subsequent sections.

The various computational managers present in mSIBYL are also based on those in SIBYL. They are, however, augmented through the content dimension, hence making the content of objects, as well as their context, accessible to the users. For example, the precedent manager is able to access previous decision problems with similar dialectical context, as well as those with similar.

In the construction project management system we present, we also use system knowledge and system dependency management to better support the computational modules. In fact, our presentation focuses on the participation of system knowledge in the operation of the various tools available in the system.

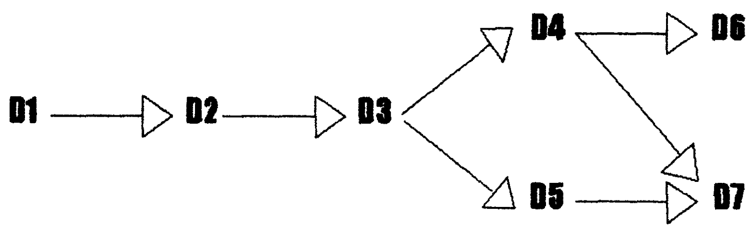
Therefore, we decided to describe the various computational modules of the project management system from these perspectives. In doing so, we use the construction site clearing example. But instead of following the five-stage decision-making process of SIBYL, i.e., setting up an initial structure, releasing it to the other participants, augmenting it, choosing an alternative, and finally evaluating the decision's outcome, we decided to describe the modules independently.

6.2. The Dependency Module

The dependency module refers to the mechanism of the system that deal with propagating external changes, and maintaining global consistency in the system. While the previous chapter described the representation of local dependency relationships, in this section we are concerned with a more global problem. We are concerned with the effect of change propagation beyond single dependency relationship links.

There are various ways of classifying dependencies. For example, Lukas Ruecker classified them into three categories. A STRICT dependency is a relationship that always holds. A STRONG dependency is a relationship that usually holds. A WEAK dependency is a relationship that exists for special cases only. More is discussed on the topic in Ruecker's thesis. However, it is difficult to incorporate any kind of dependency classification into our project management system. The reason is that the distinction between various types of dependencies is usually fuzzy, and thus hard to formalize in a way the system can understand. It is usually the user's task to identify the type of dependency, if not explicitly, then at least implicitly. Let's consider, for instance, the claim that an alternative that does not achieve an important goal should have low ratings. A priori, one may think of this rule as a strict dependency. However, if none of the other alternatives achieve that same goal, and if alternatives are rated relative to one another, then that rule may not hold anymore. Others may thus view this rule as a strong dependency.

Since these classifications tend to be arbitrary, we chose to provide the system's users different modes of dependency propagation, instead of trying to classify the dependencies up-front. In LINK mode, the system receives directions from the user at each step of the propagation. In BRANCH mode, it propagates changes until it reaches a branching point into the dependency path. Finally, in MERGE mode, the system requires user participation when a dependency propagation is caused by more than one. In the remainder of this section, we examine each mode of propagation in more detail. We illustrate our presentation using the dependency structure of Figure 6.1. Dependency D1 pointing to dependency D2 means that propagating D1 results in propagating D2.



$$\begin{tikzcd}
 & & & D4 \arrow[r] \arrow[dr] & D6 \\
D1 \arrow[r] & D2 \arrow[r] & D3 \arrow[ur] \arrow[dr] & & \\
 & & & D5 \arrow[r] & D7
\end{tikzcd}$$

Figure 6.1. A Dependency Relationship Structure.

6.2.1. Link Mode Propagation

In this mode, the system receives directions from the user at every step of the propagation. It is the most conservative mode of dependency propagation, in that the system awaits for confirmation at each step. Namely, it propagates dependency D1 of Figure 6.1 returns the result to the user. After the receiving user's approval or correction, it propagates the next dependency. Note that if the user corrects the system's result at this point, the rest of the dependency structure will probably be modified. The only advantage resulting from this mode is that the system performs the task of identifying the next dependencies that may result from propagating the current one, based on the current system's knowledge. For simple deliberation structures, this advantage becomes even less important, since the user is able to identify these dependencies without too much difficulty. In fact, this help becomes more appreciated when the dependencies are less obvious, as is the case for dependencies between different decision problems. This mode is also useful to trace the system's previous activity, helping the users track mistakes that may have been made previously. In practice, users would probably tend to use the link mode in conjunction with other faster modes for a step by step analysis of the system's performance.

6.2.2. Branch Mode Propagation

Branch mode refers to the case when the system is left to propagate changes along various dependency links, as long as no decision branch is reached. It only requests user participation at the end of a simple dependency path. In this context, a simple path is a sequence of dependencies in the dependency structure such that every dependency leads to at most one other dependency, and every dependency is reached at most once. In Figure 6.1, the sequence D1, D2, D3 is a simple path. Therefore, if the system starts propagating dependency D1, it will follow on until D3. At that point, it recognizes two different propagation paths, along dependencies D4 and D5. It then stops the propagation process, presents them to the user, and waits for the user's decision. As in the previous mode, users may choose one of these two paths, they may modify them slightly, or they may decide to use a completely new dependency rule. The advantage of the branch mode over the link mode is that it accelerates the propagation process by reducing the level of user interaction. Furthermore, if the result of a change propagation through a simple path is wrong, the user can easily backtrack along that simple path. Also, if the result is indeed wrong, then it means that a dependency rule in the path needs fine-tuning. At any rate, the user can correct the system while the path structure is still simple, so the mistake is not propagated too far.

6.2.3. Merge Mode Propagation

When operating in this mode, the system propagates changes until a join is created in the dependency structure. When a dependency that is already in the propagation path is reached again, the system stops and returns to the user for new directions. In other words, it is up to the user to combine the effect of the two incoming changes on the current node of the structure. In the case of Figure 6.1, for instance, the system propagates D5 into D7, first, and when D4 propagates into D7 as well, it returns to the user for guidance. If the effect of D4 and D5 are contradictory, then the user notices the potential for inconsistency, and acts accordingly. If the propagation is correct, then the user recognizes the effect of the initial change as a cause for inconsistency. If, on the other hand, they disagree with the propagation result, then he/she can backtrack, like in the two previous modes, in an attempt to identify the exact source of the mistake. This backtracking can be done incrementally, by setting checkpoints using either the branch mode or the link mode. This mode becomes more accurate with time. In the early stages of the system, the users still need to fine-tune the various dependency relationships. But with time, these rules get better defined, and therefore the system will have less propagation mistakes. Thus, inconsistencies resulting from merge mode propagation will be mostly due to wrong hypotheses.

6.3. The Precedent Module

The precedent module is concerned with efficient access and fast retrieval of the right information at the right time and at the right level of detail. SIBYL dealt with retrieving precedents along the dialectical dimension of knowledge. Given a particular rationale structure, the system could locate and recognize similar structures previously captured. mSIBYL provided precedent management along the semantics dimension of knowledge. Given a decision problem, the system could access its semantics dimension, and then search the transition space knowledge space for similar representations. In our system, we reinforce the precedent module by representing dependency relationships, and propagating them using various modes.

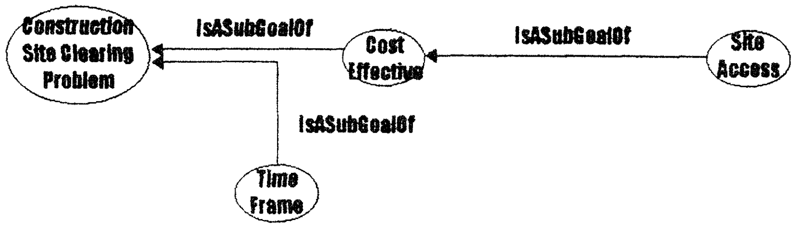
In both SIBYL and mSIBYL, precedent management is a three-stage process, consisting of query specification, information retrieval, and precedent integration. We also chose to maintain this three-stage aspect in our project management system. In the first step, the user identifies the current information against which the precedent will be compared, and states the context in which precedents are to be found. Next, the system uses various tools to locate and retrieve potential precedent candidates. Finally, the user identifies relevant information from the candidates and integrates it in the current context.

In the following, we discuss each stage in more detail. We examine the rationale dimension, the semantics dimension, and then the dependency dimension for each one of them. That is, we present the contributions from SIBYL, mSIBYL, and then we illustrate how dependency management tools improve precedent use in our system.

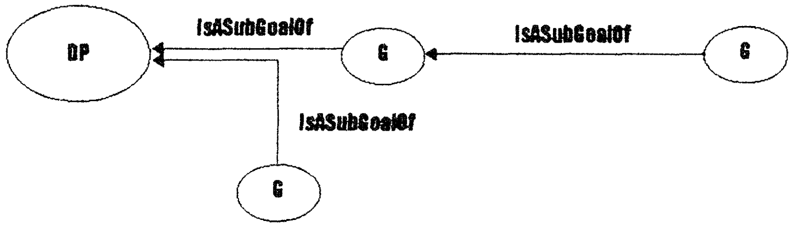
6.3.1. Query Specification

In the first stage, the user specifies what kind of precedent he/she is looking for. As we mentioned earlier, the search may be along the rationale dimension (e.g., show all decision problems that have been resolved and satisfy the low-cost requirement), or along the semantics dimension (e.g., previous examples of object removal). Most queries, however, usually involve precedent retrieval along both dimensions.

Along the rationale dimension, for example, let us consider the decision problem of the construction site clearing problem of chapter 2. Figure 6.2.a represents the current status of the CONSTRUCTION SITE CLEARING problem. Decision-makers have already identified three high-level goals, COST EFFECTIVENESS, SITE ACCESSIBILITY, and TIME FRAME. That is the solution must be relatively cheap, the site location must be taken into account when deciding on the various decisions, and the task must be performed early in the project. At that point, a deliberation participant may want to access previous cases that share the same rationale structure as this problem. He/she then decides to examine all previous problems that have already been resolved and had three goals. Figure 6.2.b illustrates the corresponding template that the user specifies. It represents the rationale structure the system will use in subsequent stages to retrieve the information requested. Indeed, after the precedent module is completed, it should give the users all **DECISION PROBLEM** objects that are related to **GOAL** object COST-EFFECTIVENESS through **IsASubGoalOf** relation objects, such that the ***status*** attribute of the decision problems is set to *resolved.*



(a) Initial State of the Construction Site Clearing Problem.



(b) Precedent Retreival Template.

Figure 6.2. Query along the Rationale Dimension.

Along the semantics dimension, on the other hand, the user specifies a transition space template in order to retrieve precedent cases. Let us suppose, for instance, that the users further refine the construction site clearing problem by recognizing that consists of two subproblems, namely the TOPSOIL REMOVAL problem and the TOPSOIL DISPOSAL problem. At that point, decision-makers have already defined what they mean by the TOPSOIL REMOVAL problem. Figure 6.3.a illustrates a high-level representation of that definition in TS representation. More specifically, the ***contact*** between the *ground* and the *topsoil* disappears, the ***distance*** between the *ground* and the *topsoil* appears, and the ***position*** of the *topsoil* changes. One participant notices this description is similar to a special case of the event of moving an object *object1* from a location *location1* to a location *location2*, as illustrated in Figure 6.3.b. He/she thus decides to examine previous cases when other such events have been discussed, and uses that description as a template for semantics precedent retrieval.

[Topsoil, Ground]	**Contact**	D
[Topsoil, Ground]	**Distance**	A
[Topsoil, Backgnd]	**Position**	Δ
[Topsoil, Backgnd]	**Heading**	A

[a] Event of Removing Topsoil from the Ground.

[Object, Location1]	**Contact**	D	~~A~~
[Object, Location1]	**Distance**	A	~~D~~
[Object, Location1]	**Position**	Δ	~~Δ~~
[Object, Location1]	**Heading**	A	D
[Object, Location1]	**IsAt**	D	~~A~~
[Object, Location2]	**Contact**	~~A~~	A
[Object, Location2]	**Distance**	~~D~~	D
[Object, Location2]	**Position**	Δ	~~Δ~~
[Object, Location2]	**Heading**	A	D
[Object, Location2]	**IsAt**	~~A~~	A

[a] Event of Moving Object1 from Location1 to Location2.

Figure 6.3. Query along the Semantics Dimension.

In the construction management system we propose, users can specify a new type of queries involving dependency relationships. For example, they can request the list of all previously decided decision problems that may be affected by some event. Although mSIBYL was already designed to manage queries that involve both dimensions, this example illustrates two new features. Since the rationale domain is represented in TS as part of the system knowledge, one query to the transition space knowledge base is enough to identify the appropriate template. Furthermore, because system dependencies are also encoded in TS, the notion of an object being affect by another object is implicitly deduced from these relationships. Figure 6.4. shows the TS event trace for the query of Figure 6.2.

		t16 t17	t18
[DP11, DP]	**IsA**	A	~~D~~
[Goal1, Goal]	**IsA**	A	~~D~~
[Goal2, Goal]	**IsA**	A	~~D~~
[Goal3, Goal]	**IsA**	A	~~D~~
[DP1, Goal1]	**IsASubGoalOf**	A	~~D~~
[DP1, Goal2]	**IsASubGoalOf**	A	~~D~~
[DP1, Goal3]	**IsASubGoalOf**	A	~~D~~

A Decision Problem with Three Subgoals.

Figure 6.4. Query in the Project Management System.

6.3.2. Information Retrieval

In this stage of precedent management, the system uses the precedent templates specified by the user, and tries to locate matching information in its knowledge base. In SIBYL, additional tools were used to access closely related objects. In mSIBYL, the TS knowledge base was used to access precedents along the semantics dimension of objects. Precedent identification in our system is based on mechanisms developed in these two works. The challenge lies in the semi-formal and the generic properties on the three systems. Indeed, the system should be able to locate precedents at the right level of detail. That is, the query should be specific enough so that it is of some interest to the user, and general enough so that a maximum number of precedents is retrieved. In the remainder of the section, we describe how each one of the three systems searches for precedents in its knowledge space.

SIBYL maintains a goal lattice to relate goals through both specialization links and instance links. The precedent manager uses this lattice to compare goals to one another. Rather than maintaining different lattices for the various object types of SIBYL, the system generates comparison measures for other typed objects through the goal lattice. For example, two goals are comparable if they are closely related in the hierarchical structure of the lattice, two decision problems are similar if they have similar goals, and alternatives are equivalent if they achieve similar goals. In the case of Figure 6.2, the system will search the goal lattice for the COST-EFFECTIVENESS goal, and extract closely related goals such as the LOW-COST goal, for instance. After that, the system examines the retained candidates for the other conditions mentioned in the query template. In our example, it accesses the decision problems that have the selected goals in their rationale structure, and checks the status fields to keep only the ones that have been resolved.

In mSIBYL, on the other hand, precedent templates are compared to other traces in the transition space representation of the system's knowledge. It uses transformation mechanisms to reformulate the template description so that more potentially relevant information may be accessed. Getting back to the example of Figure 6.3, the TS trace representing the event of moving an object from one location to another may go through reification transformation to produce a single object that would represent. Throughout these transformations, the system relies on the various heuristics incorporated into the TS framework to guide its progress towards relevant cases. Again, once the potential candidates are identified, the system accesses back the corresponding knowledge chunk, and then reaches the rationale dimension to identify the deliberation role of the chunk. At that point, the relevant rationale structure is retrieved.

Our system uses the approach introduced in mSIBYL to locate precedents in its knowledge base. However, there are two major differences with both systems. The first one is in the matching process itself. In the construction project management system we propose, both rationale queries and semantics queries are represented in TS, as we explained in the previous section. Therefore they both have the same matching process. More particularly, because the rationale taxonomy is encoded in TS, there is no need for a goal lattice anymore. The specialization/instantiation relationships between the various goals are already encoded in TS either as part of the system knowledge, or as part of the specific descriptions of the various goals. The second difference results from the representation of system dependencies in TS, as discussed earlier in the dependency management section of this chapter. Indeed, when the dependency management module is used, and dependency paths are identified, the users have the option to confirm or correct them. Also, users may choose to encode a portion of the dependency path, so that if the system comes across this path at a later stage, it can propagate the changes without requesting user supervision for that part of the path. Of course, this option would be efficient, but it is not recommended when dealing with weak dependencies. This stage of the precedent module is thus used by the dependency module when propagating changes across the knowledge base.

6.3.3. Knowledge Integration

When relevant candidates are selected, the users must examine them and decide which part of the precedent's knowledge to integrate into the current decision problem. At this point, it is possible to retrieve information from the five spaces of the rationale dimension, and from the semantics dimension itself. Users usually check the lists of potential goals, decision problems, and alternatives, and decide which ones correspond to objects in the current decision problem, which ones should be incorporated into the current context, and which ones are not relevant at all. When such objects are incorporated into the current decision problem, their corresponding argumentation is also checked and incorporated. Evaluations may also be relevant when a block of rationale structure is transferred.

When users have selected all the information they wish to incorporate, they fine-tune the integration process by editing the transferred information. A new situation may invalidate some of the transferred arguments. Evaluation criteria may be different in the new context. Also, a better understanding of a task may have resulted in a more refined definition that may partially correct—and thus contradict—previous definitions. At this stage, users may make use of the dependency management module to propagate the effects due to the newly transferred information, and to ensure that no inconsistencies are introduced.

6.4. The Viewpoint Module

Viewpoint management deals with the creation, storage, and manipulation of viewpoint objects. It provides a context in which objects that share common assumptions are grouped together for further analysis. Such common assumptions may come from the rationale dimension (e.g., describing a decision problem by incorporating additional goals), or from the semantics dimension (e.g., giving a different definition to an object). The major issue of viewpoint management is to be able to manipulate objects that hold both information that is invariant across viewpoints and information that is proper to their viewpoint of use.

In the remainder of this section, we first present a detailed description of the viewpoint object and describe how information within a viewpoint is generated and captured. Then, we explain how viewpoints are related to one another and how information is transferred from viewpoint to viewpoint. We conclude the section by illustrating viewpoint management in both SIBYL and mSIBYL, as well as in our system.

6.4.1. Representing Information within a Viewpoint

A viewpoint object is made of three fields. The <u>Name</u> field holds the name of the viewpoint object that users create to distinguish it from other viewpoints. The <u>Elements</u> field is a list of pointers that refer to the various objects that are part of the viewpoint. The <u>Viewpoint Relation</u> field relates the viewpoint to other viewpoints of the same decision problem. Because the assumptions shared by the objects belonging to a viewpoint can relate to any dimension of the system, it is therefore not convenient to use a more detailed and formal structure than the list of the <u>Element</u> field. Furthermore, like other rationale objects, viewpoints can be argued about just as they can be used to discuss other objects.

A decision problem (e.g., the CONSTRUCTION SITE CLEARING problem) is created within an initial viewpoint V_0 that is used as a reference viewpoint against which to compare other subsequent viewpoints. After some deliberation, the decision-makers are curious as to the effect of scheduling the TOPSOIL DISPOSAL task earlier in order to satisfy resource availability constraints. The users then create a new viewpoint V and link it to V_0 through an ISRELATEDTO rationale relation. The objects of V_0 are copied into V, and the new time constraint is also incorporated into V. We should note that when V is created, V_0 still represents the current status of the decision problem, whereas V represents potential alternatives for the state of the decision problem. When (and if) the decision to actually move the TOPSOIL DISPOSAL task starting time ahead is made, then all relevant information is transferred from V into V_0, as we explain next.

6.4.2. Transferring Information across Viewpoints

The Viewpoint Relations field of the viewpoint object is used to relate various viewpoints to one another. There are new relations that specify how two viewpoints V_1 and V_2 are related. They therefore represent the specialization of the ISRELATEDTO relation described in the previous example. For instance, V_1 and V_2 may have the same set of objects but the attribute values are different. The set of objects in V_1 may be a subset/superset of the set of object in V_2. Or, V_1 may precede V_2 historically, representing a state of the decision problem prior to the state represented in V_2.

As we mentioned at the beginning of this discussion, some of the information captured by an object is proper to the object itself, while the rest varies with the viewpoint in which the object is. For example, we would like objects in viewpoints that represent two different alternatives of a particular decision problem to be the same so that decision-makers are able to effectively compare the alternatives. On the other hand, we do not want for changes that occur in one viewpoint to affect object in the other viewpoint. This problem was resolved in SIBYL by assigning two Ids to every object that is created. A unique **object ID** (OID) is assigned to an object upon creation, and a **version ID** (VID) is assigned to it every time its attribute values are changed. Therefore, while object Ids are persistent across viewpoints, version Ids are not. Thus, in the previous example, the two alternatives achieve the goals of the decision problem in both viewpoints, i.e., objects ALTERNATIVE, GOAL, and DECISION PROBLEM all have the same OIDs across the viewpoints. On the other hand, the ***evaluation*** attributes of the ACHIEVE relations between alternatives and goals have different *values* in both viewpoints, because they all have different VIDs across viewpoints.

6.4.3. Illustrating Viewpoint Management

Along the rationale dimension, viewpoints can be created to represent different states of the project in the various decision rationale spaces. Thus, a viewpoint may be generated from the criteria space to examine the effect of adding a new goal into the discussion, from the alternative space to examine an alternative in more detail, from the evaluation space to evaluate the importance of a particular goal, from the argument space to examine the problem from a given perspective, and from the issue space to resolve independently issues that have been motivated by the current discussion.

Along the semantics dimension, viewpoints may also be used to answer what-if questions concerning the definitions of objects. This provides the system and its users with the ability to develop new ways to solve decision problems. Heavy use of the transition space transformations helps identify the various scenarios that are relevant to the current problem. Such scenarios are then examined separately so they can be better defined without affecting the current state of the system. After this process is completed, the users may then decide to include the appropriate information into the project representation.

Dependency management of the system knowledge provides with more efficient ways to manage viewpoints in the system. First, within a particular viewpoint V, it improves the propagation of local changes originating from the assumptions proper to it. Thus, if a goal is made more important relative to the other goals, for instance, the system updates the evaluations of the affected alternatives in V, without affecting objects outside viewpoint V. And second, system dependency management also improves the integration of information generated in the various viewpoints V into the active viewpoint V_0. Hence, after various viewpoints $V_1,...,V_4$ representing various solutions to a decision problem have been examined, users may choose to integrate some information of V_2, for instance, into V_0. The integration is done as described in section 6.3.3 describing knowledge integration in precedent management. If, on the other hand, the decision-makers want to incorporate information from several viewpoints V_1, V_2, and V_3, for example, then, they may perform the previous steps for each one of V_1, V_2, or V_3. Or they may create another viewpoint V_5 in which they include all the relevant information from V_1, V_2, and V_3. Local dependencies and inconsistencies are thus taken care for within V_5, without affecting the other viewpoints. Then knowledge from V_5 may be integrated into V_0 as before.

6.5. The Evaluation Module

The evaluation management module provides users with means to combine, access, and maintain various objects relevant to the current deliberation process. Indeed, it is this module that is concerned with relating similar objects to one another, allowing for appropriate evaluation comparisons to be made. It also provide the users with specific display tools, enabling them to access all and only the information that is relevant to them when evaluating the various alternatives. Finally, it interacts with the dependency management module to propagate evaluation changes from object to object, using dependency rules similar to those described in chapter 5. To avoid confusion, however, we should note that the evaluation values of the various objects of a decision problem are not generated by the system. Rather, the users are the original providers of these values. The evaluation manager is only an assistant to the users in performing this task.

We begin this section by presenting the basic element of evaluation—the decision matrix—provided by SIBYL to enable the users to interact with the evaluation manager. We then show in more detail how the alternative evaluation takes place, along with the criteria used to guide the process, as described in mSIBYL. Finally, we describe how system dependencies are used by this module, and how the dependency management module collaborates with the evaluation management module to better assist decision-makers.

6.5.1. The Decision Matrix

The ability of the system to handle complex situations is one of its major benefits. In the construction domain, for example, a typical project consists of various issues to be resolved, various tasks to be performed, and various criteria to consider. Furthermore, such information may be described from various perspectives and at various levels of detail. Very soon in its lifetime the project becomes so complex that decision-makers have trouble keeping track of all of its aspects. Therefore, even though the system is able to capture and manage such complexity, it must also be able to present it to its users efficiently. More specifically, users may be able to access the information that is relevant to them without having to deal with excess information.

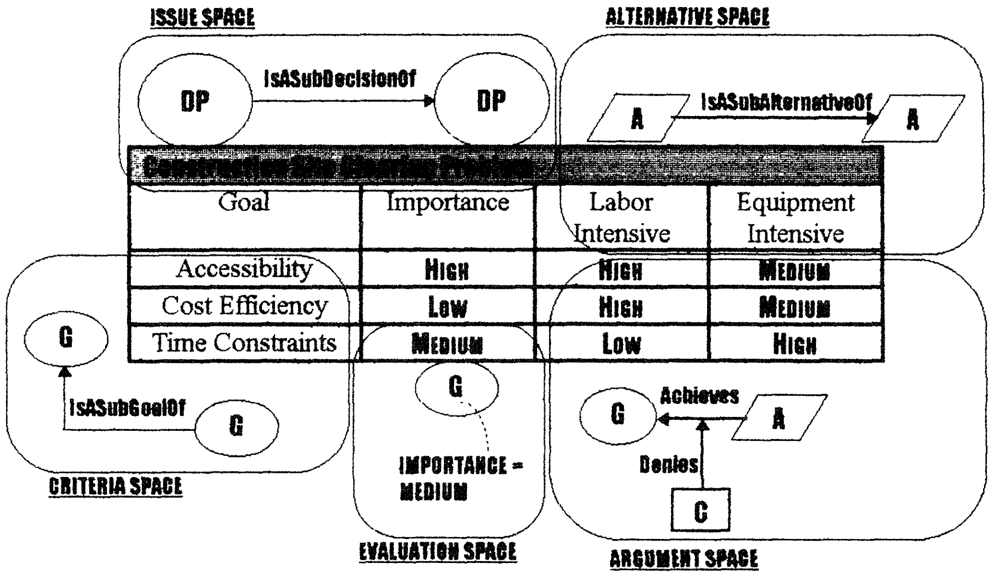
To that effect, the evaluation manager provides the users with a decision matrix structure that displays the current status of the various alternative solutions for a given problem, relative to the appropriate criteria. Figure 6.5 shows such a decision matrix for the site clearing problem. It maps goals and alternatives against each other, and describes the evaluation parameters describing the goal's importance and the extent to which each alternative achieves each goal.

Construction Site Clearing Problem			
Goal	Importance	Labor Intensive	Equipment Intensive
Accessibility	**HIGH**	**HIGH**	**MEDIUM**
Cost Efficiency	**LOW**	**HIGH**	**MEDIUM**
Time Constraints	**MEDIUM**	**LOW**	**HIGH**

Figure 6.5. Decision Matrix for the Construction Site Clearing Problem.

Furthermore, the decision matrix is also an access point to elements of the other rationale spaces that participate in the deliberation process. Thus, the issue space is accessed through cell Construction Site Clearing Problem, the alternative space through cells Labor-Intensive and Equipment-Intensive, the criteria space through cells Accessibility, Cost-Efficiency, and Time-Constraint, and the argument space through each cell of the matrix.

The evaluation space is already displayed by the matrix. Figure 6.6 illustrates how the various spaces are accessed from the decision matrix of the Site Clearing problem.



Goal	Importance	Labor Intensive	Equipment Intensive
Accessibility	HIGH	HIGH	MEDIUM
Cost Efficiency	LOW	HIGH	MEDIUM
Time Constraints	MEDIUM	LOW	HIGH

Figure 6.6. Accessing the Rationale Spaces through the Decision Matrix.

Finally, the decision matrix helps the participants set individual evaluation values for the various alternatives. But it also assists them in combining the individual values and generates an overall measure of the various alternatives relative to one another. More on the subject is explained in the remainder of the section.

6.5.2 The Evaluation Process

When all alternatives have been evaluated with respect to the various goals in the decision problem, users need to combine the evaluations into one measure in order to decide which alternative to translate into a project's task. As in mSIBYL, the default decision method in our project management system is based on Pugh's concept selection process. It is an iterative process whose main advantage is that it helps users decide on the next action to perform after each iteration. This method consists of four steps for each iteration.

First, the various alternatives are described at similar levels of detail. Indeed, decision-makers may deliberate on the various alternatives using the techniques described in previous sections. Arguments are brought in to support and/or deny other arguments. Viewpoints are used to describe alternatives separately. Knowledge from different viewpoints and different decision problems are incorporated into the active viewpoint of the current decision process. But before the various alternatives are compared with one another, they must all be expanded to comparable levels.

Second, a reference alternative A_0 for the current decision problem is selected. The use of a reference point is of particular importance in the construction project environment, where different participants have different priorities and thus different criteria evaluations. Furthermore, an evaluation measure that seems appropriate to one decision-makers may be completely irrelevant to others. Thus, the explicit use of a reference point is an attempt to standardize the evaluation process, by reducing the 'errors' due to the decision-makers' different backgrounds.

Third, the performance of the alternatives relative to the various goals is compared to that of A_0. The evaluation of alternative A_1 relative to a goal is set to '+' if A_1 performs better than reference alternative A_0, '-' if A_0 performs better, and '=' if both alternatives are comparable. So, if A_0 achieves a goal to a high extent, while A_1 does it to a lower extent, then the evaluation attribute of the Achieves relation relating A_1 to that goal is set to '-'.

Fourth, the individual measures of performance are appraised. Distinguishing between the individual measures relative to the various goals enables the users to identify the strengths and weaknesses of each alternative, relative to each goal. In some cases, decision-makers may even be able to produce a new alternative that maximizes the strengths of the initial alternatives and minimizes their weaknesses. Also, if all alternatives perform equally relative to a particular goal, then it may be that goal is not discriminating enough. Furthermore, the ability to measure alternatives using a reference point makes it easier for the users to group the individual measures into an overall evaluation parameter. Finally, two alternatives having similar overall measures may be a hint that the alternatives are not specified enough.

Figure 6.7 illustrate the decision matrix of the construction site clearing problem when this decision method is used. The reference alternative we use is a solution that uses labor and equipment in similar proportions.

Construction Site Clearing Problem				
Goal	Importance	Labor Intensive	Equipment Intensive	Reference Alternative
Accessibility	**HIGH**	+	=	**I**
Cost Efficiency	**LOW**	+	=	**I**
Time Constraints	**MEDIUM**	-	+	**I**
Overall Evaluation	**X**	**+H, +L, -M**	**=H, =L, +M**	**X**

Figure 6.7. Decision Matrix Including a Reference Alternative.

6.5.3 The Evaluation Propagation

Dependency management in our system maintains the accuracy and the validity of the evaluation module's decision matrix. For instance, if the importance of a goal is modified, then the various mechanisms of the dependency modules are propagated efficiently, and the users are notified when the overall evaluations of the alternatives need to be updated. Depending on the mode in which the dependency module is set to operate, the user is able to follow the propagation steps by accessing the various issue spaces as described previously. Similarly, if a change occurs in the reference alternative, the users are notified and decision matrices that use that alternative as a reference need to be adjusted. Subsequently, when alternative evaluation measures are altered, the overall status of the decision problem, as well as that of related decision problems, may change.

Furthermore, the evaluation process we described is meant to be a default method for alternative selection that we incorporate in the evaluation module. However, users are also able to define their own decision algorithms with their own evaluation measures. In such a case, the dependency module may be used to ensure that the user-defined measures are always well mapped from one algorithm to another. For instance, if the decision-makers decide to evaluate the status of a decision problem based on various decision algorithms, they may thus choose to create various viewpoints to represent the various evaluation procedures. Then, if the importance of a goal is changed in one viewpoint, the dependency module, along with the viewpoint module, interact with the evaluation module to appropriately update the decision matrices in the various evaluation viewpoints.

6.6. The Scheduling Module

The scheduling management module is a newly introduced module. It provides the users with means to capture, access, and manage scheduling information. Tasks are related to one another through precedence relationships. They are also related to resources through allocation relationships. Chapter 4 describes all the scheduling objects in our system. These objects are also used to propagate scheduling constraints to and from the system's rationale dimension, as illustrated in chapter 5. In this section, we describe how these scheduling dependencies are used to optimize the project's schedule.

As we stated in previous chapters, in incorporating the scheduling dimension into the project management system, we did not intend for it to be a substitute to existing scheduling techniques. In fact, we chose a broad precedence relationship classification that allows a number of classical scheduling algorithms to be used in conjunction with this system. We thus begin by describing the classical critical path method (CPM) for task scheduling. We then present the dependency matrix method (DMM) that relates tasks to one another in an attempt to better arrange the tasks so as to reduce dependencies. Finally, we illustrate how the deliberation knowledge captured in the project management system may be used to improve the results of the two methods above.

6.6.1 Implementing CPM

The critical path method establishes an optimal project schedule by computing a critical path through a logic diagram. A critical path passes through the project's critical tasks, i.e., the tasks that must be completed on schedule. A logic diagram is a network whose arrows represent the various tasks, and whose nodes represent events. Thus, the initial node of the diagram represents the event "project start," while the terminal node represents the event "project completion."

Each task T has an early start time ES (the earliest time at which the task can be started), an early finish time EF (the earliest time at which the task can be completed if it started at time ES and completed within the estimated duration), a late start time LS (the latest time the task can start and still completes within the estimated duration, i.e., by time EF), and a late finish time LF (the latest time at which a task may finish without delaying the project).

We also define task float measures to keep track of the difference between the task's early and late start and finish times. These measures capture the degree to which a task is "non-critical." Thus, the total float, TF is the amount of time a task may be delayed without affecting the project (TF = LF - (ES + Duration)). The free float, FF is the amount of time a task may be delayed without delaying the start of another task beyond its ES time. The interfering float IF the amount of time by which the task may be delayed without affecting a critical task's starting time (IF = TF - FF). Note that if a task is on the critical path, then by definition TF is zero.

Finally, we define the early event time EET, and the late event time LET. For every event, EET is the earliest time at which the event may occur, considering the duration of preceding tasks. LET is the latest time at which the event can occur, if the project is to be completed on schedule.

The CPM algorithm is a four-stage algorithm. First, the logic diagram is set up, given the task dependencies and their respective expected duration times. Second, the event times are then computed in two steps. A forward pass into the diagram computes the early event

1) draw the logic diagram, incorporating:
  the logical relationships between tasks
  the duration of these tasks.

2) compute the event times assuming $I \xrightarrow{T} J$:
  $EET(J) = \max_I \{ EET(I) + Duration(T) \}$
  $LET(I) = \min_J \{ LET(J) - Duration(T) \}$

3) compute the task times assuming $I \xrightarrow{T} J$:
  $ES(T) = EET(I)$
  $EF(T) = EET(I) + Duration(T) = ES(T) + Duration(T)$
  $LF(T) = LET(J)$
  $LS(T) = LET(J) - Duration(T) = LF(T) - Duration(T)$

4) compute float times:
  $TF(T) = LET(J) - EET(I) - Duration(T)$
  $= LF(T) - EF(T)$
  $= LS(T) - ES(T)$
  $FF(T) = EET(J) - EET(I) - Duration(T)$
  $IF(T) = TF(T) - FF(T) = LET(J) - EET(J)$

times, while a backward pass does the same for the late event times. Third, the task times are computed as well, using both their respective duration times and the event times of the adjacent nodes. Finally, the float times are computed for every task, hence representing the leeway in which a particular task may be delayed or extended. We should note that for critical tasks, i.e., tasks that are on the critical path, there is no such leeway, IF = 0. Thus, delaying a critical task will automatically delay the whole project's completion. In the following, we describe every step of the algorithm in more detail.

The mapping between these various time variables and the ones we defined in our project management system is discussed in section 6.6.3, along with the use of our system to monitor the project's schedule.

6.6.2 Incorporating DMM

The technique we describe in this section was developed by Steven Eppinger in an attempt to better represent the information requirements in project development, and to better understand the structure of the information flow in this context. Information is generated, used, and transferred at every stage of the project development. All along, decisions are subject to constraints imposed by other decisions, and they propagate new constraints to other decisions. As we have seen in previous chapters, both the volume and the complexity of the information flow may be difficult to keep track of since very early in the project's lifetime.

Dependency matrices use the decomposition of the information flow itself to break down the project's tasks into smaller subtasks. Thus, an activity of n tasks is represented by an nxn sparse matrix. Element a_{IJ} of the matrix is non-zero if task I provides information to node J. The algorithm presented is a two-stage process. The *partitioning* stage consists of reordering tasks to maximize the availability of the required information. The *tearing* stage consists of ordering within blocks of coupled tasks to find an initial ordering to start the iteration.

The first step of the partitioning algorithm is to identify the independent tasks. Hence, tasks that have all the required information available (empty rows in the dependency matrix) are scheduled in the early stages, while tasks that do not provide any information for subsequent tasks (empty columns in the dependency matrix) are scheduled for later stages. The second step consists of identifying and grouping dependent tasks. When all independent tasks are removed from the iteration, then the remaining tasks must be arranged in at least one information dependency loop. At this point, the algorithm identifies these dependency loops, and collapses its members into a single composite task. The partitioning process then continues, resulting in a mostly lower triangular matrix, with composite tasks on the diagonal. Figure 6.8 represents the partitioning process for a six-task activity. Task E is brought forward because it does not need any information. Dependency loop between tasks A and C is then identified, and the two tasks are collapsed into a single task AC. Task AC does not provide any information for subsequent tasks. It is therefore put at the end of the matrix. The dependency loop BFD is then identified. The last matrix represents the final outcome of the partitioning algorithm.

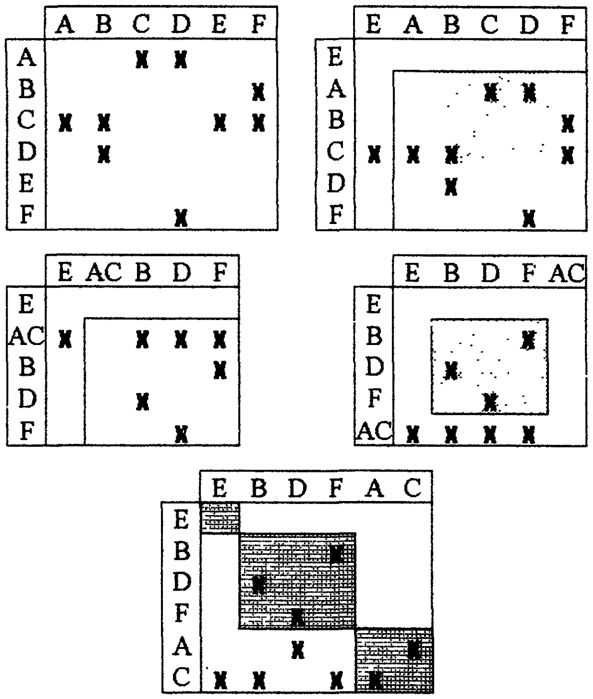


Figure 6.8. The Partitioning Algorithm.

The tearing analysis consists of ordering the tasks within a dependency block after the partitioning has taken place. More knowledge on the dependencies between the various tasks is needed to make the appropriate judgment. For instance, in some cases, breaking up a task into subtasks may remove the dependency loop. In the example of Figure 6.8, task A may be broken down into two tasks A_1 and A_2, such that A_1 depends on C and C depends on A_2. Furthermore, instead on just identifying tasks that depend on one another, the dependency matrix may include numerical values that measure the amount of information dependency between tasks. This brings an extra dimension to the tearing stage, in that an ordering based on the weights of the various dependencies may be used.

6.6.3 Using Rationales in Project Monitoring

The project management system we described throughout this thesis was to interact with external scheduling algorithms such as CPM and dependency ordering algorithms such as the dependency matrix algorithm described above. As we stated earlier, we presented these two schemes to illustrate how our system may interact with other systems. At this point of the design of the project management system, we have not committed to any particular external scheduling system. Instead, we have tried to provide a generic base to capture scheduling entities. In the remainder of this section, we illustrate how the two systems just described interact with our system, and how the entities of the scheduling module may be mapped into the domain of the new systems.

6.6.4 Interacting with CPM

When using the critical path method method, the interaction with our system's scheduling module takes place at the first step, when the logic diagram is formed. Indeed, the algorithm requires both the task duration and the dependencies between tasks to be specified. In our system, every task object has a duration field. Furthermore, since CPM does not differentiate between the types of precedence relationships, all such relationships will be mapped to a single type. In our system, we represent how tasks affect one another, while CPM represents what tasks affect one another. All the other time attributes are generated by the CPM algorithm, and therefore do not require any additional information from the scheduling module.

Another interaction between both systems occurs during the project monitoring stage. Indeed, when a decision is changed, a new task is introduced, an old task is canceled, or a constraint is modified, the various mechanisms of the system's dependency module propagate these changes up to the scheduling module. If at this point, a dependency relationship or a task duration is affected, and/or a task is added or deleted, then the CPM scheduler needs to be notified. Conversely, the project's schedule may be used during the deliberation process. This may be done through a Procedure object or a Claim object introduced to answer a question, for instance.

6.6.5 Interacting with DMM

The dependency matrix method was developed to capture the various dependencies representing the information flow constraints. In our project management system, the method could be used to represent the dependencies between the various tasks that are captured by the scheduling module as a result of the deliberation process. As with CPM, all the necessary information for the partitioning algorithm is, in fact, captured by the objects of the scheduling module.

But the most important advantage of using DMM in conjunction with the project management system is the three-dimensional feature of the latter. Indeed, the users' ability to access the project's context through the rationale dimension and its structure through the semantics dimension provides a straightforward application for efficient tearing algorithms. For example, the scheduling module's task object has a subtask field that may used to break down dependency loops. Also, decision-makers have more information concerning the type of dependencies represented in the dependency matrix. They may choose to use this information directly, or they may prefer to slightly modify DMM so that it can classify the dependencies. Different measures may be used to distinguish between the strength of the dependency relationships. Alternatively, the users may define different types of dependencies, and assign a priority weight to each type of dependency.

CHAPTER 7

CONCLUSION

In this chapter, we wrap up the thesis with concluding remarks on our endeavors. First, we present a general overview. We include a summary of the driving motivation behind this study, and a description of the construction project management system we propose. Then, we present the main contributions of our work to the field of project management. Next., we discuss the system's limitations that we anticipate, and try to present ways to get around them. After that, we describe research that has been—or is currently being—done in related areas. We conclude our presentation with an survey of potential future research that may provide natural extensions to the current work. Throughout this chapter, we illustrate the scope, contribution, role, and extension of our work from three perspectives. Indeed, the construction domain, the dependency management problem, and the information representation technology are the three aspects making up our research. Thus, the various sections of this chapter are organized around these three points.

7.1. Research Overview

7.1.1. Benefits of Decision Rationale Management Systems

Decision rationales need to be managed within a single deliberation session, maintained across deliberation sessions, shared across decisions, and reused in future decisions. Decision rationale management systems allow their users to access the right amount of information at the right level of detail and at the right time. They provide a structured framework for precedent use, a good forum for transfer of knowledge from experts to decision-makers, and a relevant environment to incorporate various computational tools, such as dependency management.

7.1.2. Properties of the Construction Project Domain

The inherent complexity of today's construction industry suggests a strong need for appropriate construction project management systems, and it offers a good forum for the study of dependency management. Indeed, the projects are short, various professions interact extensively and constantly, and new project techniques—e.g. subcontracting, concurrent engineering, fast-tracking—emerge. Thus, dependency management , in such an environment is of paramount importance, as a high volume of information that is highly interconnected needs to be maintained. Furthermore, although dependencies may be classified across participant professions, across project stages, and across information scope, we examined dependency management across the scope of knowledge captured by the system, using the construction site clearing problem as an example.

7.1.3. Advantages of Combining SIBYL, TS, and mSIBYL

Our framework of a construction project management system was based on the rationale management system developed by Jintae Lee, the event-based knowledge representation developed by Gary Borchardt, and on the documentation system developed by Lukas Ruecker. The decision rationale language consists of five spaces of deliberation that are used to capture the rationales behind a decision, compare different alternatives when making the decision, evaluate these alternatives, capture the criteria used in the evaluation process, and associate this decision to other related ones. The transition space representation provides a natural way to describe objects' functionality and behavior throughout a project, and to capture and propagate changes along event-based dependency relationships, hence maintaining the system's overall consistency. The research behind ,mSIBYL had a paramount impact in the development of our system, in that similar domains—product development and project management—were examined, complementary issues—precedent management and dependency management—were addressed, and the same technology—SIBYL augmented with TS—was used similarly.

7.1.4. Guidelines for a Project Management System

In conceptualizing our construction project management system, we have opted for a framework that relies heavily on user interaction, system flexibility, information reusability, incremental formalization and growth, and three-dimensional information management. User interaction needs to be efficient and smooth because the system operates as an assistant to its users who actually make the final decisions. System flexibility consists of interface flexibility, communication flexibility, and structure. Reusability is of paramount importance in construction projects because the industry is relatively stable, in that the same techniques are used from project to project. Incremental formalization and growth is a necessary property to our system because the domain of application is too wide to be fully formalized, and information is constantly generated during a project. Finally, a three-dimensional approach is desirable because it allows users to keep track of the decisions being made, the argumentation behind them, and the tasks resulting from them.

7.1.5. Representation of System Dependencies

Dependency management was a major part of our research. In the context of project management systems, we have distinguished between domain knowledge and system knowledge, and we have concentrated on system dependency representation only. System knowledge and domain knowledge differ in that they both capture different kinds of information that is generated by different sources at different times and for different purposes. In representing system dependencies, we use the transition space framework to capture dependencies along both the rationale dimension—effects of attribute changes, object creation and deletion—and the scheduling dimension—dependencies within a same task, across tasks, between tasks and resources, and due to object creation and deletion.

7.1.6. Features of the Project Management System

Finally, we designed our project management system to provide its users with tools to react to changes of external events, learn from past experience, analyze issues from different perspectives, and maintain an up-to-date project schedule. The dependency module offers three modes of execution of change propagation, the link mode, the branch mode, and the merge mode. The precedent module goes through a three-stage process according to which the precedent template is specified, the matching information is retrieved, and the relevant knowledge is integrated into the current context. The evaluation module provides a decision matrix through which the various spaces of deliberation may be accessed, so that objects in these spaces are evaluated appropriately, and the evaluation values are propagated accordingly. Finally, the scheduling module allows the system's users to interface with scheduling algorithms that may be used to generate and maintain project schedules. In our thesis, we included a description of the critical path method that generates project schedules, a presentation of a decision matrix method that helps rearrange tasks so as to minimize the dependencies between them, and an illustration of how both CPM and DMM interact with our project management system.

7.2. Research Contributions

We have undertaken this work with the goal of developing a conceptual framework for a project management system. In fact, the contribution of such system is threefold. Our first motivation was to investigate the problem of dependency management. The second purpose was to study the construction industry domain. Finally, our other goal was to explore the capabilities of a new event-based knowledge representation.

7.2.1. The Problem: Dependency Management

The problem of dependency management is a universal problem that concerns various parties at various hierarchical levels of an organization. At the low level, individuals worry about the effects of one's actions on their own situations. At a higher level, project managers are concerned with the effects of one team's decisions on other teams' actions. At an even higher level, top management wants to know how one organizational change may affect future operation of the overall organization.

Furthermore, dependency relationships occur in various environments. For instance, dependency mismanagement may have social consequences—the decision of a taking a particular job depends highly on the benefits offered. In other situations, there may be economic consequences—choosing one project organization over another affects the project's duration and ultimately its cost. Finally, there may even be political consequences—the stand of an influential group on an important issue depends on the issues it supports.

Hence, dependency management is of paramount importance for humans, as it may affect them at various levels and in various circumstances. The formulation and study of the various type of dependencies—irrespective of the domain of application—provides a better understanding of how objects are interrelated. Understanding dependencies is a first step towards managing them. It is a nontrivial problem because there are many possible contexts of analysis. For example, some dependency relationship are proper to a particular domain, while others are more generic, and it is often difficult to distinguish them.

7.2.2. The Domain: Construction Project

Our next contribution relates to our choice of the domain of application. We decided to examine the construction industry environment for various reasons. First, the complexity of the industry makes room for various types of dependencies that may be classified according to various criteria. Dependency relationships in a construction project may occur at different stages of a construction project. They may affect and be affected by objects of different types. Or they may even occur between different professional groups. The fragmented characteristic of the industry provides room for even more dependencies.

Furthermore, various professions interact within a construction project to deliver one common product. This melting pot of different backgrounds—and thus different criteria and different interests—provides a broader scope of application of a project management system. It thus allows us to test dependency management concepts in a broader environment, hence forcing an even better understanding of dependency relationships.

The construction domain also introduces new concerns that need to be incorporated into a decision-making management system. In order to be able to manage a project properly, participants must maintain an accurate schedule of the tasks that need to be done, those that have already been done, the time at which they are scheduled to be done, and their effects on the overall project. A decision management system that is used in this environment must therefore provide tools to manage the scheduling issues of the project.

7.2.3. The Technology: Information Representation

Finally, our work is also a contribution from a technological standpoint. We have indeed examined a new way of semi-formal information representation. First, we have tried to capture and manage system dependencies, i.e., dependencies that affect objects describing the system's state, and not the domain of application. Such dependency relationships may be identified—and thus captured—during the implementation stage of the project management system, and they don't need to be altered during the system's use, unless the semantics of some system objects are modified.

Also, we have presented ways to represent scheduling issues within a decision rationale management system. We have incorporated a new scheduling dimension against which scheduling issues may be argued about, as well as be used to argue about other statements. We have also developed this dimension as an interface between the decision management system and other scheduling algorithms. We have done so by "translating" scheduling entities such as tasks and resources from the deliberation environment into the scheduling environment.

Finally, we have augmented the rationale management system by encoding objects of SIBYL into the transition space knowledge base. This has enabled the system to capture and manage the various relationships between rationale objects, providing an initial support for system dependency representation and management. Indeed, dependencies more complex than the ones represented by Object Lens in SIBYL may now be captured by the project management system we propose.

7.3. Related Research

Extensive efforts have been done in the domain of decision management system. In this section, we list examples of other related research that complement our threefold—Problem-Domain-Technology—work. Lukas Ruecker is concentrating on the problem of precedent management. Bhavya Lal examined the role of rationale management system in the domain of Nuclear Engineering. Finally, Gary Borchardt is studying ways to represent domain dependencies in decision rationale management system. Readers should refer to the original works for more detailed descriptions.

7.3.1. The Problem of Precedent Management

In his work on a design documentation and management system, Lukas Ruecker augmented SIBYL by encoding the objects' contents into the TS knowledge base. In doing so, he concentrated most of his efforts on issues related to precedent management, i.e. the ability to use knowledge and experience that have been previously acquired by the system's users. His system is therefore able to capture, index, and retrieve precedent cases of design knowledge along the semantics dimension as well as the dialectical dimension. Issues of concern include the selection criteria of information indexing, the trade-off between system selection and user selection, and the effect of information indexing and retrieval on the overall system performance.

7.3.2. The Domain of Nuclear Engineering

Bhavya Lal worked on "a framework for incorporating best practices at nuclear power plants." Best practices are principles and improvement measures that are believed to have helped improve the performance of certain nuclear power plants. They are extracted from their original setting and transferred to other plants to achieve good operational levels.

However, this transfer of information is not always accurate. The descriptions of the practices are usually incomplete because the rationales behind them are not captured. Also, it is often difficult to identify which practice is the most beneficial to a particular situation, precisely because of the lack of its context of generation and use. Finally, adapting the practices into a new environment may cause problems, because managers don't have enough information about them to tailor them to their specific needs.

7.3.3. The Technology behind the Impact Language

Finally, the current efforts of Gary Borchardt complement our study of dependency management representation. While we have concentrated on system dependencies, he, on the other hand, is investigating ways to capture domain dependencies in decision rationale management system, using the transition space representation he developed. This problem is more complex because domain knowledge is semi-formal and described at various levels of abstraction. Therefore, there cannot be an a priori classification of the dependencies, since the content of the knowledge base is unknown. The challenge is thus to identify dependency relationships between objects that may be formalized to different extents or described at different levels.

7.4. Future Research

Our research was not intended to be a final product. Rather, it is only a basis to future efforts. In this section, we describe the various extensions that we envision. We start by describing what needs to be done to improve dependency management within the project management system we propose. We then present our approach to better understand the construction industry domain. Finally, we discuss technology-related improvements that may be incorporated into the system.

7.4.1. Dependency Management

As we mentioned in earlier chapters of this thesis, the distinction between the various types of dependencies—e.g., strict dependencies, strong dependencies, and weak dependencies—is very blur. One possible direction of research is thus to investigate new and improved dependency classifications. Such classifications should be more clear-cut than the existing ones, in order to obtain a higher level of formalization of dependencies within the system. Another possibility is to incorporate a user-development kit into the system. This way, users are able to create new object types, specify dependency relationships, and fine-tune system knowledge semantics. This alternative reduces the need for a dependency classification methodology. The problem, however, is that the burden is in fact pushed on to the users, since they are the ones that need to make these classification decisions now. A thorough study of the trade-offs between these two possibility therefore need to be done.

7.4.2. Construction Industry

There are many practical limitations that ma inhibit decision-makers from using a project management system such as the one we described. For example, users may be reluctant to go through the "extra" effort to thoroughly think through their actions, in spite of the obvious advantage of long-term accuracy. Also, employees may hesitate to document all their decision-making into an accessible environment, as this information may be used to review and evaluate their performance. Managers also may vacillate in using such system for security reasons. While these problems are not proper to the construction industry only, they are, however, enhanced because of the fragmentation property. Any effort to solve—or at least reduce—these problems will need to be multiplied in order to affect the various participants of construction projects. We attempt to solve the first problem by working on more efficient user-interfaces, including new communication devices such as pen-based devices, in order to reduce the burden on users. The other two problems may be resolved through appropriate techniques that would convince and educate potential users to re-examine their decision-making processes.

7.4.3. Event-Based Representation

The first extension we envision from the technological perspective is to implement a small-scale prototype in order to have a better feel of the performance of the system we propose. This will provide us with a more accurate description of the current capabilities and limitations of our system. We also intend to refine the dialectical dimension to improve system dependency representation and propagation. The challenge is to reach a level of formalization where the objects are general enough that users are not burdened in deciding the dialectical role of a statement ,and specific enough that attributes are well defined and dependencies accurately propagated. Furthermore, we would like to investigate ways to incorporate quantitative descriptions into the transition space representation. This would enable the system to capture quantitative relations such as proportionality. It would also provide a better framework for time references in TS.

BIBLIOGRAPHY

Bahler, D., and Bowen, J., "Supporting Multiple Perspectives: A Constraint-Based Approach to Concurrent Engineering," *Artificial Intelligence in Design*, Kluwer Academic Publishers, The Netherlands, 1992.

Bailey, Stephen, *Offices*, Butterworth Architecture, London, 1989.

Borchardt, Gary, *Transition Space*, Artificial Intelligence Laboratory Memo No. 1238, MIT, Cambridge, Ma., 1990.

Borchardt, Gary, "Understanding Causal Descriptions of Physical Systems," *The National Conference on Artificial Intelligence*, San Jose, California, 1992.

Borchardt, Gary, *Causal Reconstruction: Understanding Causal Descriptions of Physical Systems*, PhD Thesis, MIT, Cambridge, Ma., 1992.

Ching, Francis D.K., and Adams, Cassandra, *Building Construction Illustrated*, Van Nostrand Reinhold,New York, 1991, second edition.

Conklin, J., and Begeman, M., "gIbis: a Hypertext Tool for Exploratory Policy Discussion," *ACM Transaction on Office Information Systems*, 1988.

Eppinger, Steven D., Whitney, Daniel, E., Smith, Robert P., and Gebala, David, A., "Organizing the Tasks in Complex Design Projects," *ASME International Conference on Design Theory and Methodology*, Chicago, Illinois, 1990.

Eppinger, Steven D., and Gebala, David, A., "Methods for Analyzing Design Procedures," *ASME International Conference on Design Theory and Methodology*, 1991.

Fischer, Gerhard, and McCall, Raymond, "JANUS: Integrating Hypertext with a Knowledge-Based Design Environment," *Hypertext '89 Proceedings*, 1989.

Lai, K.Y., "Object Lens: A Spreadsheet for Cooperative Work," *ACM Transactions on Office Information Systems*, Volume 6, No. 4, 1988.

Lal, Bhavya, A Framework for Incorporating Best Practices at Nuclear Power Plants, Department of Nuclear Engineering, MIT, Cambridge, Ma.

Lee, Jintae, *Knowledge Base Integration: What can we Learn from Database Integration Research?*, Artificial Intelligence Laboratory Memo No. 1011, MIT, Cambridge, Ma., 1988.

Lee, Jintae, *Decision Representation Language (DRL) and its Support Environment*, Artificial Intelligence Laboratory Working Paper No. 325, MIT, Cambridge, Ma., 1989.

Lee, Jintae, *A Decision Rationale Management System: Capturing, Reusing, and Managing the Reasons for Decisions*, PhD Thesis, MIT, Cambridge, Ma., 1992.

Lenat, Douglas B., and Guha, R.V., *Building Large Knowledge Based Systems: Representation and Inference in the CYC Project*, Addison-Wesley Publishing, 1990.

MacCall, R., PhIbis: "Procedurally Hierarchical Issue Based Information Systems," *Proceedings of the Conference on Planning and Design in Architecture*, Boston, Ma., 1987,

MacLean, A., Young, R., and Moran, T., "Design Rationale: the Argument behind the Artifact," *Proceedings of CHI '89*, Austin, Texas, 1989.

Pena, Feniosky, A., *Design Process Information and Goal Achievement*, Master Thesis, MIT, Cambridge, Ma., 1991.

Pugh, S., "Further Development of the Hypothesis of Static/Dynamic Concepts in Product Design," *The International Symposium on Design and Synthesis Proceedings*, Tokyo, Japan, 1984.

Ruecker, Lukas F.S., and Seering, W.P., *User-Oriented Intelligent Indexing and Retrieval in a Design Documentation System*, Artificial Intelligence Laboratory, MIT, Cambridge, Ma.

Ruecker, Lukas F.S., and Seering, W.P., "A Dialectical Reasoning System for Design Documentation," *ASME International Conference on Design Theory and Methodology*, 1991.

Ruecker, Lukas F.S., *mSIBYL - a Design Documentation and Management System*, Master Thesis, MIT, Cambridge, Ma., 1992.

Serrano, D., *Constraint Management in Conceptual Design*, PhD Thesis, MIT, Cambridge, Ma., 1987.

Toulmin, S., *TheUses of Arguments*, Cambridge University Press, Cambridge, UK, 1958.

Ulrich, Karl, and Seering, W.P., "Function Sharing in Mechanical Design," *Proceedings of the 7th National Conference on Artificial Intelligence,* Saint Paul, 1988.

Willis, Edward, *Scheduling Construction Projects*, Wiley, New York, 1986.

Winston, P.H., Binford, T.O., Katz, B., and Lowry, M., *Learning Physical Descriptions from Functional Definitions, Examples, and Precedents*, Artificail Intelligence Laboratory Memo No. 679, MIT, Cambridge, Ma., 1983.

Winston, Patrick H., *Artificial Intelligence*, Addison-Wesley Publishing, Reading, Ma., 1984, second edition.

Zepezauer, Steven V., "The Information Age," *Engineering Horizons*, Fall 1993 Edition.