USACERL Technical Report P-90/07
February 1990

US Army Corps
of Engineers
Construction Engineering
Research Laboratory

AD-A222 132

A Knowledge Engineering Approach to Analysis and Evaluation of Construction Schedules

by:
Jesus M. De La Garza
E. William East
Nie-Jia Yau

A deeper understanding is needed of how owners and contractors evaluate the quality and reasonableness of construction schedules. The research described in this report contributes to this understanding by articulating a subset of scheduling principles, thus enabling models of the construction schedule evaluation process to be created for subsequent automation.

This research recommends that construction schedule criticism be assigned a high project management task priority. Recognizing the importance of this task may induce large, medium, and small firms to perform formal schedule criticism, which is now done by only a few large corporations. This report provides guidelines on performing schedule criticism.

Another aspect of this research is the development of prototype applications systems. These prototype show that knowledge-based systems technology offers programming technique that facilitate the representation and manipulation of scheduling knowledge. The infrastructure of knowledge representation schemes provided by these prototype systems will be of value during future development of an operational system.

The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official indorsement or approval of the use of such commercial products. The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED

DO NOT RETURN IT TO THE ORIGINATOR

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1990	Final February 1990

4. TITLE AND SUBTITLE

A Knowledge Engineering Approach to Analysis and Evaluation of Construction Schedules

5. FUNDING NUMBERS

PE - 4A161101
PR - AT23
WU - E59
TA - SA

6. AUTHOR(S)

De La Garza, Jesus M.; East, E. William; and Yau, Nie-Jia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U. S. Army Construction Engineering Research Laboratory
P.O. Box 4005
Champaign, IL 61824-4005

8. PERFORMING ORGANIZATION REPORT NUMBER

TR P-90/07

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

HQUSACE
20 MASSACHUSETTS AVENUE NW
WASHINGTON, DC 20314-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Copies are available from the National Information Service, Springfield, VA 22161

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

A deeper understanding is needed of how owners and contractors evaluate the quality and reasonableness of construction schedules. The research described in this report contributes to this understanding by articulating a subset of scheduling principles, thus enabling models of the construction schedule evaluation process to be created for subsequent automation.

This research recommends that construction schedule criticism be assigned a high project management task priority. Recognizing the importance of this task may induce large, medium, and small firms to perform formal schedule criticism, which is now done by only a few large corporations. This report provides guidelines on performing schedule criticism.

Another aspect of this research is the development of prototype applications systems. These prototypes show that knowledge-based systems technology offers programming techniques that facilitate the representation and manipulation of scheduling knowledge. The infrastructure of knowledge representation schemes provided by these prototype systems will be of value during future development of an operational system.

14. SUBJECT TERMS

construction scheduling, knowledge based system

15. NUMBER OF PAGES

130

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

FOREWORD

This research was performed for the Directorate of Military Programs, Headquarters, U.S. Army Corps of Engineers (HQUSACE), under Project 4A161101AT23, "Basic Research in Military Construction"; Technical Area SA; Work Unit E59, "An Integrated, Artificial Intelligence Based Project Management System."

This research was conducted by the Facility Systems Division (FS) of the U.S. Army Construction Engineering Research Laboratory (USACERL). Dr. Michael J. O'Connor is Chief of USACERL-FS.

Jesus M. De La Garza is now an Assistant Professor of Civil Engineering at the Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Appreciation is expressed to the trio of experts who assisted in the knowledge engineering effort: Mr. Gerald Landsly from W. E. O'Neil Construction Company, Mr. Steve Pinnell from Pinnell Engineering, Inc., and CPT Scott Prosuch, Military Science Department, University of California at Davis. Appreciation is also expressed to the following individuals: Dr. C. William Ibbs, Dr. LeRoy T. Boyer, Dr. John W. Melin, and Dr. Michael J. O'Connor.

COL Carl O. Magnell is Commander and Director of USACERL, and Dr. L. R. Shaffer is Technical Director.

CONTENTS

TABLES

FIGURES

A KNOWLEDGE ENGINEERING APPROACH TO ANALYSIS AND EVALUATION OF CONSTRUCTION SCHEDULES

1 INTRODUCTION

Background

The origin of computer-based schedule analysis for project management dates back to 1959, when two separate but essentially similar procedures were developed: the Program Evaluation and Review Technique (PERT) and the Critical Path Method (CPM). PERT involves the specification of three separate duration estimates for each activity and probabilistic procedures to produce estimates of project completion. CPM uses one deterministic duration estimate. These two approaches are commonly used to represent a project in the form of a network diagram and to perform the necessary calculations on the diagram to determine the critical path.

A study of the use of network-based scheduling techniques [Davis 84] found that CPM and PERT were being employed significantly more for planning than for control, and that top management commitment is essential for their successful implementation. The study showed that the second most frequent top management concern regarding operation and use of a scheduling system was that its implementation requires an excessive amount of work. Its use is characterized by complex input/output notation, and excessive detail and paper generated by the systems.

Since this survey was conducted, availability of personal computing resources has led to a proliferation of computerized Project Management Systems (PMSs). Although these PMSs have become more substantial and complex, they are still widely acknowledged as incomplete project control aids. Specifically, their inability to collect data and interpret qualitative and subjective project information requires construction managers to interpret such information largely unassisted.

The lack of computer-aided interpretation leads to variations of construction managers' and firms' performance. This is especially so in schedule analysis and control, in which experience and subjective judgment play a major role. In addition to these problems, construction managers are dissuaded from using the CPM and PERT methods effectively by the need to perform time-consuming network searches and laborious hand calculations, which are typical of "what if" analyses.

Objective

The objective of this study was to synthesize the operational knowledge needed to perform a project management expert task.

The second objective of this research was to determine if it is feasible to codify a subset of construction scheduling principles in a critic Knowledge-Based System (KBS). Such a KBS should enhance scheduling software usage. This objective was accomplished by developing software prototypes which interface with commercially available scheduling packages.

This research advances the state of the art by (1) adding to and formalizing an amorphous mass of construction scheduling knowledge, (2) improving the recognition of construction schedule criticism as a formal project management task, (3) adapting a knowledge engineering methodology capable of transforming ill-defined construction scheduling knowledge into the specifics of an operational KBS, and (4) providing a frame-based infrastructure of knowledge representation schemes.

In short, the goal of this research was to unite some parts of the decision-making process in the field of construction schedule analysis and control with contemporary tools available from the field of Artificial Intelligence (AI).

Technological Opportunities

AI technology provides a way to capture and replicate empirical judgmental knowledge in the form of computer programs called KBSs. During a National Science Foundation conference held in May 1985 at the University of Illinois, several researchers pointed out that the nature of the construction industry makes these KBSs especially applicable to construction engineering and management [Ibbs 85, Ibbs 86]. Researchers and practitioners agree that in this industry there is a high dependence on empirical rules and procedures derived from experience, rather than from a scientific knowledge source. Concurrent with the conference, researchers were beginning to isolate construction scheduling and control as a prominent example for KBS application [Avots 85, Levitt 85, McGartland 85, Nay 85].

A subsequent workshop [O'Connor 87] confirmed and endorsed the fact that construction schedule criticism and comparison represent crucial elements of effective project management. Participants of this symposium identified KBSs as an excellent technology to (1) capture generic and idiosyncratic knowledge about construction schedule criticism, comparison, and generation, and (2) develop graphical user interfaces for operational KBSs.

Two other workshops [Kitzmiller 87, Wilson 87] outlined the criteria for and purpose of needed intelligent interfaces for current algorithmic software. [Wilson 87 p 16] summarized the consensus of the participants at his workshop, with respect to intelligent software interfaces, as follows:

> Many current software packages are large algorithms, or collections of linked algorithms, that are difficult to use or whose results demand serious, informed interpretation. . . Knowledge-Based Systems (KBSs) offer the potential of providing intelligent pre- and post-processors of such packages. . . As post-processors, the KBSs can be used to interpret program results as to their accuracy, validity and applicability. A major research issue within this approach is the assessment of how much of the knowledge required for such application is compiled and generic, and how much is empirical and idiosyncratic. . . Generic tools for such appli- cations should be able to represent the 'widely-shared) compiled knowledge, while at the same time allowing the user to augment the knowledge base with his/her own experimental knowledge.

Research of KBSs that have proven successful in other fields holds promise for providing needed assistance in construction schedule analysis.

Definition of Construction Schedule Analysis

The construction schedule analysis problem domain is characterized by the use of expert knowledge, judgment, and experience. Management of dynamic schedules requires not only knowing what has changed but also reacting to those changes before their impact occurs. Comparison and analysis of schedules are typically performed at three different stages: (1) at the time of the initial version; (2) across different schedule versions during project execution; and (3) within the last schedule version, reflecting actual and baseline expectations. This report focuses only on the construction expertise required to accomplish the first and second phases. In these stages, the issues of maintaining an on-going historic data base play a less important role than in the third stage.

Initial schedule analysis, including cash flow projections, is defined as the verifications conducted by owners or contractors on the initial construction planning schedule. Project managers, whose task at this stage is to assess whether the proposed schedule is reasonable, need answers to such questions as the following:

- o Does the schedule meet the contract requirements?
- o Is the critical path reasonable?
- o Are owner-controlled activities included?
- o Have major subcontractors participated in the formulation of the plan?
- o What is the overall degree of schedule criticality?
- o Do procurement activities precede special installation tasks?
- o Does the cost estimate comply with the contract documents?

In-progress schedule analysis is defined as the type of evaluations conducted by owners or contractors across different schedule versions during project execution. In this phase, project managers face some of the same questions related to initial analysis and other issues such as, Are we on schedule? How much should we pay? What is different about these schedules?

In-progress schedule evaluation provides project managers with answers to such questions as the following:

- o Are winter-sensitive activities being scheduled during winter?
- o Is the progress payment request reasonable?
- o Should the duration of future activities be modified based on past experience?
- o How can I tell if activities are in trouble?

Both initial and in-progress schedule analyses are considered with four major categories of issues: (1) general requirements, (2) logic, (3) cost, and (4) time.

Report Organization

This report is divided into eight chapters. Chapter 2 presents system-level and computational KBS concepts, KBS architectures, the knowledge engineering process, and a brief description of some KBSs being developed in the construction scheduling domain. Chapter 3 defines the research scope and provides an overview of the knowledge transformation process. Chapter 4 is a detailed description of the knowledge elicitation methods utilized and a profile of all sources of expertise upon which this research draws. Moreover, it provides an overview of the structure imposed on the knowledge, and the specific conceptual knowledge incorporated in the "paper" knowledge base. Chapter 5 defines a generic object-oriented knowledge acquisition grammar used to transform the scheduling knowledge from English form to English-like notation. It also presents some illustrative examples of such knowledge metamorphosis. Chapter 6 profiles the surveyed software tools, and identifies the functional requirements for the software. This chapter also introduces the necessary mappings to translate the knowledge from an object-oriented,

English-like notation to a computer language syntax, and describes the "electronic" representation of the operational knowledge by means of examples. Chapter 7 describes a personal computer prototype, the flow of information between a PC-based PMS and a KBS, the conceptual operation of an operational KBS, and an implementation in the Automated Reasoning Tool (ARTuTmd) programming environment. Chapter 8 provides a retrospective analysis of the project goals, and outlines where future research is needed.

2 KNOWLEDGE-BASED SYSTEMS CONCEPTS

Overview

KBSs provide researchers with a different way of thinking about programming. Applications difficult to program--because the domain knowledge is highly heuristic or because applications in which explicit algorithms provide only a partial solution--become approachable with this technique. On the other hand, KBSs give no real advantage over traditional methods for some applications (e.g., a stand-alone CPM program), and other applications that still resist computerization (e.g., actual measurement of work in place).

System-Level Concepts

This section describes the definitions provided by [Cugini 87] regarding: (1) KBS techniques, (2) KBS languages, and (3) KBS applications to which this report adheres.

KBS Techniques

KBS techniques consist of software concepts which describe the behavior of a program at an abstract level and how to understand the execution of a program. Some common examples are rules, frames, facts, pattern matching, forward chaining, and backward chaining.

KBS Languages

A KBS language provides support for KBS techniques. This support may be more or less direct, depending on the specific purpose of the language. For example, FORTRAN can implement KBS techniques [Lopez 84], but more effort is required than with LISP. Examples of general purpose representation languages with features especially designed to support KBS techniques are, among others, LISP, Prolog, OPS5, and SRL. There also exist KBS building platforms which greatly facilitate the development of a KBS. Many representation issues and control strategy decisions are already incorporated in these high-level tools. Inference Corporation's ART and IntelliCorp's Knowledge Engineering Environment (KEEuTmd) are among the more comprehensive high-end AI environments.

KBS Applications

A KBS application is any system whose software implementation employs KBS techniques as the major structuring mechanism. An important class of KBS applications is Expert Systems (ESs). The major function of ESs is to simulate and augment the problem-solving behavior of experts in a narrow domain. Knowledge about the domain is the key factor in the performance of such ESs [Feigenbaum 83]. Expert-level performance means that the computer program is expert by virtue of excellent performance, despite the fact that in current applications a system cannot improve its behavior based on experience [Buchanan 86]; that is, it is static.

Thus, the term ES refers to the external behavior of computer systems, whereas the term KBS refers to their internal structure. Similar definitions of these terms are also found in [Ibbs 85].

The spectrum of KBS applications consists of the following functions [Hayes-Roth 83]: interpretation, prediction, diagnosis, design, planning, monitoring, debugging, repair, instruction, and control.

Interpretation — An interpretation system takes incoming data and explains their meaning by inferring the problem state to which the data correspond. For example, an architect's set of drawings may be represented in a symbolic or textual language and be interpreted to generate their counterpart in a graphic language.

Prediction — A prediction application would be a KBS for forecasting construction cost and time from existing project conditions.

Diagnosis — Diagnosis KBSs locate malfunctions from observed and interpreted data. A KBS for criticizing and comparing construction schedules, the subject of this thesis, is such an example.

Design — A design application consists of developing a configuration for an object which satisfies all applicable constraints The automatic generation of conceptual and detailed design specifications for a construction project represents such an example.

Planning — A planning KBS explores possible future actions to produce a series of plausible steps leading to a desired goal. An example in this category would be the automatic or interactive generation of construction schedules.

Monitoring — A monitoring application is one in which a KBS examines real-time data and compares the observations with planned behavior to determine flaws in the plan and malfunctions of the system. A KBS for explaining the reasons for lagging construction falls in this category.

Debugging — A debugging KBS can be used as an interactive aid to a design architect. For example, an architect might design a building on the KBS, and then tell the system exactly what the building was designed to be. A debugging KBS would test the design to determine if it would work as planned, and if it did not, the system would isolate the errors and suggest design changes to correct them.

Repair — A repair KBS combines the activities of diagnostic and planning systems to create and execute plans for repairing faulty systems. For example, it would be possible to develop a system to diagnose and prescribe repairs for malfunctioning construction equipment.

Instruction — An instruction KBS plays the role of teacher to a student by diagnosing the student's problem areas from errors in his solutions to exercises. A KBS for teaching the fundamentals of bidding construction projects is such an example.

Control — A control KBS encompasses many of the functions of the applications previously described. It would be theoretically possible, for example, to design a KBS to coordinate all construction project management functions for a specific type of building.

KBS Architecture

The basic structure of a KBS contains five components: (1) a knowledge base for storing the facts and the heuristic knowledge of the domain, (2) an inference engine for constructing proofs, (3) a data base or context for the facts that reflect the current state of the problem solution, (4) a user interface for carrying out consultation, and (5) an explanation module to rationalize the steps taken by the inference engine. Some components are not part of every KBS, but are desirable in a final product: (1) a knowledge acquisition facility for extracting the knowledge from the human experts and (2) an asynchronous data

handler to integrate all incoming information, e.g., real-time data, arbitrary user input, user answers, data base responses.

Knowledge Base

The knowledge base is the passive part of a KBS where the knowledge specific to the domain of the problem to be solved is stored. The way the knowledge is represented is similar to how it might be conceived by an expert. The knowledge base may include declarative or data-like knowledge as well as complex procedural knowledge, i.e., how to transform data.

The knowledge in a KBS may be categorized according to its location along a continuum from deep to shallow knowledge. Deep knowledge captures the causal, underlying structure and facilitates so-called reasoning from first principles, e.g., physics or chemistry. The statement "construction activities should be measurable" is an example of deep knowledge. By contrast, shallow knowledge consists of empirical associations or ad hoc rules which evolve from repeatedly solving similar problems, rather than from some elegant theoretical foundation. It is sufficient for performing the task, but typically lacks understanding of the underlying causal mechanisms. The statement "construction activities are assigned durations within a [minimum, maximum] range" is an example of shallow knowledge.

Inference Engine

The inference engine is an interpreter for a high-level language in which the knowledge is expressed. It operates on the data base and draws on the facts and rules in the knowledge base to build a derivation. A derivation typically includes addition, deletion, or modification of facts in the data base. The inference engine determines, by means of a control strategy like forward or backward chaining, not only which rules are currently applicable but also the order of application.

Data Base

The data base is dynamic, and at any point in time it represents a snapshot of the information (problem facts, solution status facts, assumptions, goals, relationships, etc.) being considered next to solve the problem.

User Interface

The user interface provides a textual and graphical interface between the KBS and the user. The acceptance or rejection of an operational KBS is directly correlated with this interface.

Explanation Module

This module's primary objective is the transformation of programs seen as "black boxes" into "glass boxes." This is achieved by providing the user with tools to solicit a description of the a priori or a posteriori inferring processes.

Knowledge Acquisition Module

The purpose of the knowledge acquisition module is to help extract the expert's knowledge and check for knowledge base consistency. It is an add-on system with its own interface.

Asynchronous Data Handler

The purpose of the asynchronous data handler is to process and translate all external incoming data, for example, user supplied input, real-time data, and external data base responses. The system described in this report reacts to data coming from an external PMS, on an IBM/AT personal computer.

Computational Concepts

In light of the basic requirements of the construction schedule criticism application (Chapter 6) and considering the basic features for representing and manipulating knowledge of different software tools (Chapter 6), ART was selected as one of the advanced computing tools on which to test ideas.

This section examines a computational model which is helpful in understanding ART's computing environment. It is not meant to be complete; rather, it is meant to provide a vocabulary with which to describe some implementation aspects of the ideas presented in this report. The reader is referred to [Clayton 87, Williams 85] for a detailed description of ART's features.

Facts, Schemata, and Rules

Facts. Facts are propositions about some particular thing. For example, sample propositions might take the following form: "The early start of forming the 2nd-floor slab is delayed," "Forming the 2nd-floor slab is followed by placing reinforcing steel," "The cost of forming the 2nd-floor slab is $10,000." Each of these propositions is a fundamental piece of information about the construction activity in question.

Schemata. Schemata (the plural of schema) are equivalent to frames, units, and objects. They provide a knowledge representation language that emphasizes object descriptions. A schema is a collection of facts that represents an object or class of objects that share certain properties. The properties of each schema are either attributes of the object or links to other schemata. Schema definitions can be organized into hierarchies in which knowledge about an object can automatically be deduced based on the classes to which it belongs. For example, a "2nd-floor slab" schema may be linked to "concrete," "superstructure," and "floor" schemata, and thereby inherit information about the types of weather impact, location, and geometry, respectively.

Rules. Rules define the procedural knowledge available to an application. Rules are statements about some general relationships within the domain. They modify the data base by allowing the derivation of new facts from old ones. For example, "Activities with zero total float are critical" and "Activities with materials sensitive to water are sensitive to climatic conditions" are rules. Rules can also execute procedures, and perform input and output operations. Generally a rule is a set of conditions associated with a proposed set of actions. The rule actions represent a proposed course of action if there are data in the data base that fit the rule's conditions. These are "proposed" actions because some circumstances result in activating several rules at once. In such a case ART breaks the tie and decides which rules to execute and when. Firing a rule is equivalent to executing actions that cause changes in the data base by adding, retracting, or modifying pieces of information. Depending on the circumstances, it is possible for a newly asserted fact to activate other rules. Similarly, its retraction can disable already activated, but not yet fired, rules.

Representation of Facts, Schemata, and Rules

Facts. A fact's structure is a list of elements which provides a way of representing a relationship among several things collectively. For example, (resource-shortage carpenters 3JAN87 15JAN87) is a fact that indicates a shortage of carpenters between 3 to 15 January 1987. The first term in every fact is a relation name, i.e., resource-shortage, and the remaining terms are values. The fact relation name

identifies how the values are linked to each other.

Schemata. A schema definition is also a list of elements. In the schema heading the keyword "defschema" is followed by the name of the object being defined, then an optional description in double-quotes. Following the heading are one or more "slots." Each slot is enclosed in parentheses and describes an attribute of the schema. For example:

```
(defschema Steel-erection-1st-floor
    "schedule data about an activity"
    (sub-task-of Beckman-Institute)
    (task-number 10)
    (task-level 2)
    (duration 60)
    (earliest-start 2MAR87)
    (earliest-finish 1MAY87)
    .
    .
    .)
```

A slot in a schema definition has a strong correlation to a fact proposition. The name of the slot appears first, followed by one or more values. The slot name is analogous to a fact relation name because it relates the schema object to the slot's values. A schema processor compiles each schema slot into a triplet fact comprising the slot name, the name of the schema, and the value of the slot, i.e., (slot-name schema-name slot-value). For example, some of the facts that result after compiling the schema Steel-erection-1st-floor are the following:

- fact-1 (sub-task-of Steel-erection-1st-floor Beckman-Institute)
- fact-2 (task-number Steel-erection-1st-floor 10)
- fact-3 (task-level Steel-erection-1st-floor 2)
- fact-4 (duration Steel-erection-1st-floor 60)
- fact-5 (earliest-start Steel-erection-1st-floor 2MAR87)
- fact-6 (earliest-finish Steel-erection-1st-floor 1MAY87)

Rules. A rule definition is also a list of elements. The rule heading contains the keyword "defrule," then a unique symbol that identifies the rule being defined, and an optional description in double-quotes containing descriptive or explanatory information that is not processed. Following the heading is the forward rule body, which contains the conditions (left-hand side) and actions (right-hand side) of the rule.

The forward rule body is made of patterns, which are descriptions of data. When used on the left-hand side, they form a connection among existing facts in the data base and the rule itself. When used on the right-hand side, they primarily modify the data base. Typically a rule's patterns are represented using structures built from general terms. These terms are represented by variables, which may represent different values at different times, and constants, which can only represent a single value.

A variable is represented as a series of characters, the first of which is ? (e.g., ?location). Once a variable is bound on the left-hand side of a rule, subsequent occurrences of the same variable act like constants. (This is not strictly true. Each rule's pattern acts independently of the other, and it is

instantiated in the pattern net by propositions as if acting alone. The rule. however, is not activated until the join net applies an additional condition which binds the first occurrence of a variable in the first pattern in which it was found, and tests to see if subsequent uses of the variables in the other patterns are equal to the bound variable.) A following example is a simplified rule that propagates scheduling constraints:

```
(defrule propagate-forward-pass-FS-constraint
    "given a new earliest finish for preceding tasks,
    propagate the earliest possible start for the
    finish-to-start constrained tasks"
    (phase cpm-forward-pass)
    (parent ?pre-task ?task ?rel)
    (instance-of ?rel FS)
    (earliest-finish ?pre-task ?time1)
    (earliest-start ?task ?time2&:(< ?time2 ?time1))
    =>
    (modify (earliest-start ?task ?time1)))
```

This rule propagates the earliest finish of an activity to all its successors, providing that the earliest start of the successor is not already greater than the earliest finish of its predecessor.

Model of Derivation

The reasoning process may be more formally represented by a directed cyclic graph (Figure 1). Each node represents a fact or a rule successfully activated. The directed edges connecting the nodes represent the flow of derivation. A, B, C, and G are labels for the facts, D is the activated rule, and E and F are the bindings in the rule application.

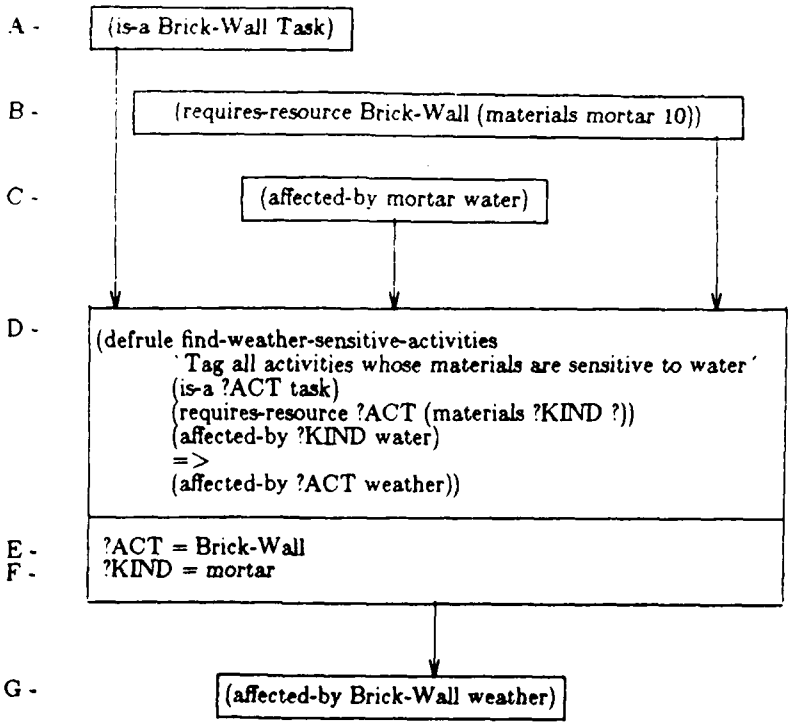

Figure 1. Portion of a directed acyclic reasoning graph.

Forward and Backward Chaining

Forward Chaining. Forward chaining ("bottom up," "data driven," and "antecedent driven" are terms equivalent to forward chaining) refers to the direction in which the inferring takes place. In this case, the presence of certain facts in the data base activates the rules to reach appropriate conclusions. This is reasoning in the forward direction, and when rules begin to build on one another's conclusions to develop a chain of logic, the process is called "forward chaining." Forward chaining corresponds to the following intuitive account:

1. A relevant rule: if an activity has materials sensitive to water, then that activity is sensitive to climatic conditions.

2. The following facts are added to the data base: brick-wall is an activity; brick-wall uses mortar and bricks; mortar reacts to water.

3. A consequence of this rule application to brick-wall and mortar is the derivation of the fact that brick-wall is sensitive to climatic conditions.

The most important concept in ART's data-driven computation is that the facts drive the rules. In this application, a rule waits for a fact from the data base to collide with it and activate one of the rule's patterns, meaning that an instance matching the general rule's pattern has been found.

Backward Chaining. Backward chaining ("top down," "goal driven," "hypothesis driven," and "consequent driven" are terms equivalent to backward chaining) reasoning identifies a desired conclusion and works backward along a chain of inferences in search of known facts that would support the conclusion. It is a process in which a chain of inferences from effect to cause is attempted, instead of the other way around. The intuitive line of reasoning would be, for example the following:

1. The current goal is to find out the types of impact on schedule to which the brick-wall activity is subject.

2. There is a rule in the knowledge base that states that if an activity has materials sensitive to water, then that activity is sensitive to climatic conditions.

3. A specialized form of this rule is that brick-wall is affected by climatic conditions if it requires materials that are sensitive to water.

4. So, does brick-wall use any material that reacts with water? A relevant fact in the data base is that brick-wall uses mortar and that mortar reacts to water.

5. Therefore, climatic conditions may have an impact on the schedule dates of brick-wall.

In ART, KEE, and other programming environments, forward and backward chaining can be integrated and combined in the same program. In the most general sense, backward chaining in ART is used to lend support and assistance to partially matched forward chaining rules.

In terms of construction schedule criticism, both forward and backward chaining rules may be combined to produce a bidirectional strategy. Forward chaining rules can be used to react to changes or inconsistencies in the schedule. The actions of these rules are mainly to post plausible hypotheses for explaining such events. Backward chaining rules can then pursue each of these hypotheses to seek supporting evidence.

Pattern Matching

The basic idea of pattern matching is that there is a pattern, a template for some set of facts, which is compared with some candidate facts in the data base. The comparison may be a failure if the facts in the data base fail to conform to the requirements of the pattern. Or it may be a success, in which case the information that defines how the facts match the pattern is preserved. Just as a fact or proposition is a sequence of terms, a proposition pattern is a sequence of term restrictions. A term restriction defines the limitations placed on the term. Each term restriction attempts to match a corresponding value within a proposition. A restriction initiates a test of the corresponding contents of a proposition. A proposition instantiates a pattern if and only if all the tests initiated by the pattern's restrictions are satisfied by the contents of the proposition. Each fact in the data base that instantiates a pattern is said to be an instance of the pattern.

Typically a pattern is represented using structures built up from general terms. These terms are represented by variables and constants. When a constant is included in a pattern, the matching test succeeds if there is a fact in the data base with the same sequence of characters at the corresponding position. For example, the pattern

(slack-time act-20 15)

consists of three constant restrictions which match the fact

fact-19 (slack-time act-20 15)

but do not match the fact

fact-12 (slack-time act-17 15).

A variable consists of a series of characters, the first of which is ?, i.e., ?project. A variable acts as a wild card restriction the first time it is encountered in a pattern, and as a constant each subsequent time. Patterns are matched against facts from left to right. For example, the pattern

(location ?project mid-west in-progress)

matches the following facts

fact-10 (location Beckman-Institute mid-west in-progress)

fact-17 (location Superior-courts mid-west in-progress)

fact-20 (location Microelectronics-Lab mid-west in-progress)

but does not match the facts

fact-15 (location World-Trade-Center east-coast in-progress)

fact-47 (location Sears-Tower mid-west built).

Contrast to Conventional Algorithmic Techniques

The most basic distinction between KBSs and conventional algorithmic techniques lies in the use and manipulation of knowledge [Cugini 87, Fenves 87, Waterman 86].

KBSs manipulate knowledge, while conventional algorithmic programs manipulate data. KBSs separate the explicit problem-solving knowledge from the control program, while conventional algorithmic programs reflect knowledge about how to transform data. Such procedural knowledge is implicitly and inflexibly encoded in the algorithm. Figure 2 depicts these two differences. In short, this ability to treat expertise as manipulable objects gives KBSs much of their power and flexibility.

Knowledge Engineering Process

Knowledge engineering is the software engineering discipline focusing on constructing KBSs. It is an incremental and cyclical process that requires the interaction of a domain expert(s) and a knowledge engineer. Knowledge engineers are people who are familiar with the process of formalizing the knowledge of a domain expert. The knowledge engineer interacts with the domain expert to define the components of the KBS, identifies and extracts the necessary knowledge, and uses KBS languages to configure this information for the computer.

Since this process is similar to building an algorithmic program, knowledge engineers will eventually disappear with the introduction of a new generation of KBS languages [Fenves 87]. Such KBS building tools will allow application programmers to become the knowledge engineers in a particular domain and to incorporate significant components of the domain knowledge base themselves.

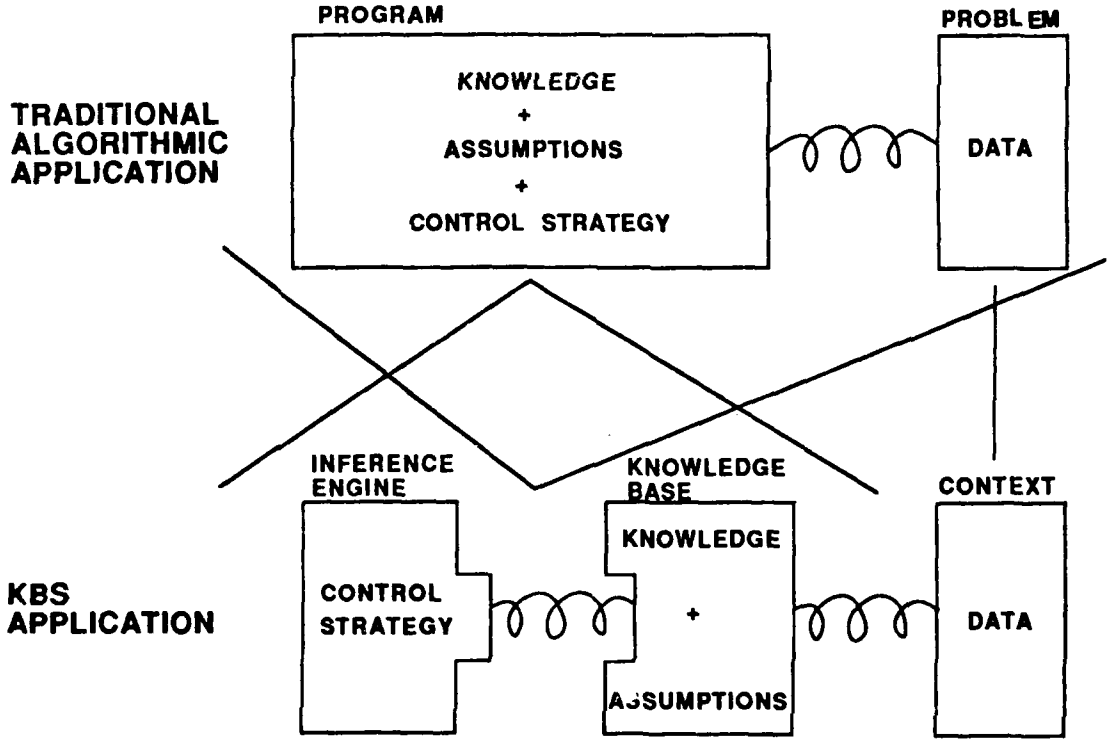

Figure 2. Algorithmic and KBS differences.

KBS Applications to Construction Engineering and Management

KBSs are being developed for a wide range of civil engineering applications. A recent survey discussed KBS applications grouped under the categories of construction, structural, geotechnical, environmental, and transportation engineering [Kim 86]. This study found distinctive characteristics of KBSs in civil engineering applications, for example, use of causal knowledge, interaction with algorithmic programs, interaction with data bases, and use of spatial attributes of objects.

In the field of construction engineering and management, the following four reasons explain why this field is a good candidate for employing KBS technology [Levitt 86]: (1) construction is an experience-based industry, (2) construction decisions must be made in real time, (3) construction decisions involve managerial issues, and (4) construction automation needs smart robots. By looking at some of the efforts devoted to implementing this technology, it is possible to verify the fact that these criteria are governing the development of such ideas. Evidence for this claim is given by the following applications.

[McGartland 85] describes how this new technology can be conceptually applied to cost, schedule, time control, and purchasing and inventory control.

[Nay 85] defines an expert system framework for analyzing construction project risks. This framework is based upon the different risk occurrences that can be associated with a work package, i.e., the lowest level of activity definition suitable for progress monitoring. Hitachi System Development Lab [Niwa 82, 83, 84] also reported on a KBS called Project Risk Assessment System for large construction projects. The KBS focuses on the project execution stage and attempts to identify risks and prevent them in advance. These goals are accomplished by first identifying risks and risk countermeasures and then mapping these to standard work packages.

[Diekmann 84] describes the Differing *Site Conditions Analysis System* (DSCAS) aimed at advising owners, engineers, and contractors in their efforts to negotiate claim settlements. This system was built to treat only differing site condition claims in federal jurisdiction.

[Levitt 85] presents a KBS which includes knowledge of project management and experience-based knowledge of a construction task. The project management knowledge consists of generic descriptions of activities, resources, risk factors, etc. The risk factors which could potentially impact the durations of planned activities are classified as being favorable or unfavorable. Their effects are then projected on the durations of unfinished activities, assuming the risk factors continue to be favorable or unfavorable. Such a system exercises project control by generating a meaningful update of the project schedule using explicit knowledge both of a particular construction domain and of project management techniques.

"What if" scenarios form the basis for exploring, merging, and eliminating hypothetical alternatives [Williams 85, Clayton 87, IntelliCorp 85]. Along these lines, [Kunz 86] discusses the use of contingent possibilities and time dependence situations in the design and construction of an offshore oil platform project. The proposed system represents anticipated contingencies such as different combinations of soil types and labor conditions, and then reasons the effects of these factors on the cost and duration of the project. The system allows distinguishing between choices, which are made with certainty by the user, and implication of choices, where the user generally has incomplete knowledge about which of several possible implications will actually occur. The use of contingent possibilities gives the user the opportunity of identifying scenarios of interest to realize the most favorable contingencies while minimizing the effect of less favorable ones.

[Hendrickson 86] presents preliminary experiences with Construction Planex, a KBS for construction planning. Its purpose is to synthesize construction schedules by (1) breaking down a project into subtasks,

(2) diagnosing each subtask's resource needs, (3) relating subtasks with precedence relationships that satisfy physical and resource constraints, and (4) predicting each subtask's duration and cost.

[Gray 86] reports on the cognitive processes used by expert construction planners. His study involves a knowledge elicitation exercise in which different expert schedulers participate and each generates a plan of the same building. The results of this exercise consist of (1) a set of rules which can be applied to identify the significant activities at the conceptual design phase, (2) a set of rules for predecessor selection, (3) a set of rules for precedence relationship definition, and (4) a set of rules for activity duration calculation based on its precedence relation.

[Levitt 87] articulates an excellent philosophic framework for the use of AI techniques as aids in project management. Such a framework is based on a taxonomy of project management functions (objective setting, planning, scheduling, and control) and of levels of management (executive, work package, and task). The study not only identifies the key cognitive skills for each project management function and level but also proposes specific AI techniques that might provide extensions to traditional PMSs. Knowledge-based interactive graphics are suggested as the core of all possible new decision support tools.

This sample of operational, developmental, and conceptual prototypes indicates a long-term commitment to KBS research by academics and, perhaps, use by practitioners.

3 OVERVIEW OF THE KNOWLEDGE BASE

Scope

The scope of the knowledge base has been restricted to a class of buildings in order to maintain a narrow focus and to concentrate the study on the scheduling process itself. The class of buildings the knowledge base addresses consists of midrise reinforced concrete or steel structures. The advantages of using this type of construction are that (1) it represents a large segment of the building construction market, (2) it is characterized by repetitive operations, and (3) solutions to early-diagnosed problems may be propagated to the rest of the structure. The upper limit on the number of stories is roughly 19. The midrise structure is chosen mainly because the construction of high-rise buildings, 20 stories and above, requires another set of elevators and, usually, different construction equipment and methods. The scope of the knowledge base has also been limited to commercial buildings since the construction methods for these types of buildings are similar. In contrast to this type of construction are projects such as hospitals, factories, warehouses, homes, schools, dams, bridges, highways, and so forth.

Knowledge Metamorphosis

The conversion of words of advice into executable procedures is an important task. The knowledge transformation is briefly presented here to provide an overview of the sequenced process and to clarify the transition from one stage to another. Figure 3 depicts the evolution of the knowledge base. More detailed discussions are the subject of Chapters 4, 5, and 6.

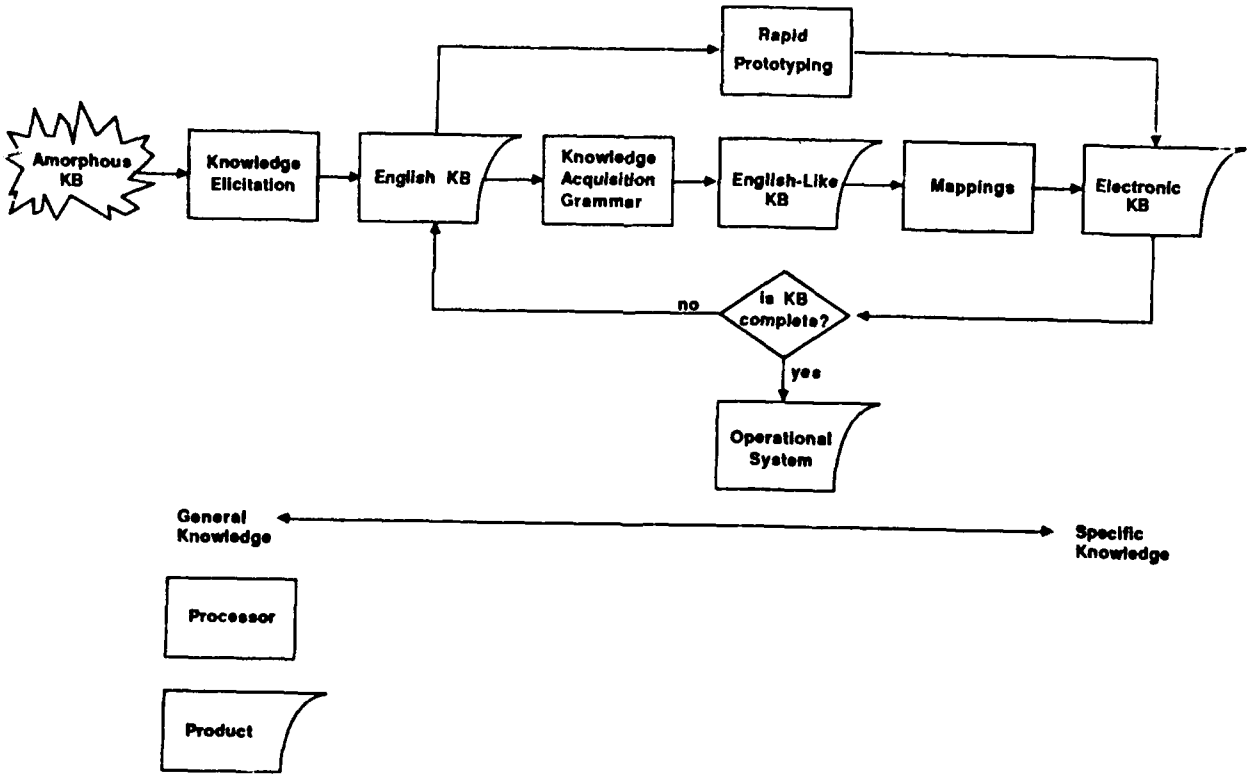

Figure 3. Knowledge metamorphosis.

Amorphous Knowledge Base

Despite the economic benefits of performing schedule analysis, many project managers do not do so because (1) it requires project managers to have a high level of technical expertise, which can only be acquired with years of experience or good education; (2) the time-consuming nature of most procedures dissuades project managers from analyzing the schedules effectively; and (3) reports of on-going work are often 1 or 2 weeks behind real time. Thus, construction managers do not take management information systems seriously.

Expertise-related problems occur because senior project managers cannot transmit all their knowledge to junior project managers. Some of the reasons for these phenomena are the following:

1. Senior project managers know more than they can explain about construction schedule analysis.

2. Most expert knowledge is empirical; large parts are not grounded on first principles.

3. Senior project managers learn to recognize thousands of associations between data and solutions to the extent they are unaware of the process. They learn to recognize patterns of solutions as opposed to specific instances.

4. The major difference between a junior and a senior project manager is the quantity, quality, retrievability, and organization of his or her knowledge.

5. Much of the knowledge to which senior project managers have access is idiosyncratic to the company they represent and to the type of projects with which they are most familiar.

In addition to the problems of insufficient expertise, comprehensive schedule analyses are impaired by the need to perform tedious hand calculations on reams of PMS reports.

Despite the proliferation of computerized PMSs, there has been little effort to quantify and structure the knowledge required to interpret the PMS information. Thus, the title amorphous knowledge base is appropriate.

English Knowledge Base

The English knowledge base consists of a set of conceptual scheduling provisions and procedures written in plain English. These provisions and procedures are the product of investigating and applying three knowledge elicitation techniques to the amorphous mass of construction scheduling expertise. Figure 4 shows the techniques used in this research and their output. The overlapping depicted in the provisions and methods boxes symbolically implies some duplication in their contents.

The union of all provisions defines the breadth of the knowledge base. An important attribute of these conceptual provisions is that they represent what is known but do not explicitly indicate how they should be interpreted or implemented. For example, a provision in the time category states: "Float should be broad enough to support the premise that it has not been manipulated." In this case, the intent of the provision is not intertwined with any float-sequestering technique.

The union of the methods generated by each knowledge elicitation technique delineates the depth of the knowledge base. These procedures symbolize how a conceptual provision may be interpreted. The conceptual provision stated above may be interpreted by relating it to preferential scheduling techniques, such as preferential logic ties, lead/lag activities, or extended activity durations.

English-Like Knowledge Base

Since much of the construction scheduling knowledge is procedural, the knowledge engineering approach at this stage has as its main objective the transformation of conceptual provisions into executable procedures.

Some knowledge a project manager exercises is generic, while other knowledge is idiosyncratic to the company with which he or she is affiliated. Thus, there can be several ways to implement a conceptual scheduling provision. The purpose of this phase is to identify at least one type of implementation and allow flexibility for further modifications or additions.

Once a procedure is attached to a provision, its representation is no longer in plain English. Rather, it is represented using a notation that is less idiomatic and more procedural. However, with this format, a person who is not familiar with the specifics of the computer language in which the KBS is written can still read and record additional knowledge.

Electronic Knowledge Base

The computer implementation of conceptual provisions requires the selection of programming languages. Commercially available AI tools that are applicable to the derivation end of the problem-solving spectrum are considered.

Three different programming environments are used for developing these ideas to the proof-of-concept level (that is, having enough functionality to demonstrate concepts but not really developed yet to the operational level): (1) Texas Instrument's Personal Consultant PlusuTmd (PC Plus), (2) ART, and (3) Gold Hill's GoldWorksuTmd. Each of these, or any other computing platform, requires a set of mappings whose function is to transform the knowledge written in the English-like notation into the specifics of the chosen programming language syntax. Once in this format, modifications to the knowledge base require the expertise of a programmer or the use of a high-level knowledge as interface that would allow nonprogrammers access to it.

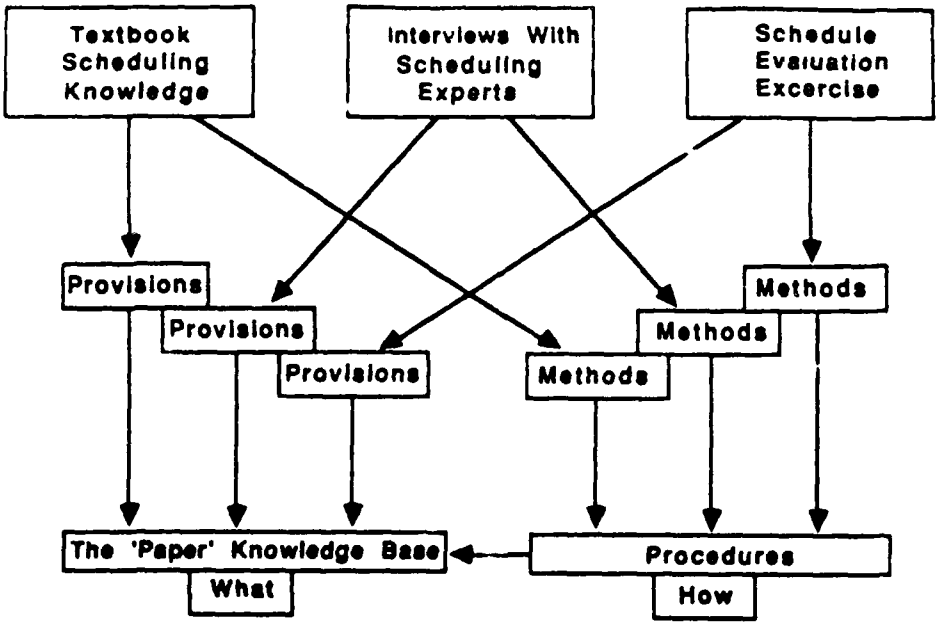

Figure 4. Knowledge elicitation techniques.

4 KNOWLEDGE ELICITATION

Purpose

Knowledge elicitation is perhaps the most ambitious, important, time-consuming, ill-structured, and challenging phase of any knowledge engineering effort. Its purpose here is to determine the practical feasibility of capturing construction scheduling knowledge for a KBS. This objective is accomplished by investigating knowledge acquisition methodologies capable of stimulating and structuring an amorphous mass of construction scheduling knowledge, while not getting diverted by the specifics of implementation.

The lessons learned during this process, as well as the captured structured knowledge base, represent major contributions of this research project.

Issues and Methods

The knowledge engineer works closely with a domain specialist whose knowledge needs to be transferred. The knowledge engineer attempts to transfer the specialist's expertise into rules which can be encoded in a machine system. No single methodology for the process of knowledge elicitation has proved universally effective In this section, a working classification of methods for eliciting expert knowledge is outlined [Hart 85, Hoffman 87, Trimble 86].

Unstructured Interview

The knowledge engineer does not have a list of detailed questions to ask. Rather, he or she asks spontaneous conceptual questions while the expert talks about the domain. For example, the knowledge engineer might ask the expert a question such as, What constitutes good scheduling practices? This technique can involve making an audiotape of the expert's discourse and producing a transcript. The technique produces a lot of relevant and irrelevant information, the analysis of which can be time-consuming.

Method of Familiar Tasks

This technique involves observing the expert at work. Studying experts includes identifying goals, subgoals, data to which they like to have access, and the information they produce. This method is usually applied to generate the first draft of the knowledge base. It also includes the collection and analysis of relevant public domain knowledge.

Structured Interview

In this technique, the knowledge engineer uses the knowledge base developed from the "method of familiar tasks" or the "unstructured interview." The expert comments on each entry in this knowledge base to confirm, expand, delete, qualify, or reorganize it. The knowledge engineer switches from general to more specific questions like, How do you know that the logic of this schedule would not work? The product of this method can be considered a second-generation knowledge base.

Limited Information Tasks

This technique tinkers with the thought process by restricting the amount and kind of information available to the expert. This technique is useful for stimulating the formulation of hypotheses and strategic

thinking. Because the method forces the expert to rely heavily on compiled and intuitive knowledge, it is likely to cause the expert frustration and discomfort.

Constrained Processing Tasks

This technique differs from the "limited information tasks" by imposing external constraints on the expert; that is, it either (1) limits the time within which the expert learns factual data and draws recommendations or (2) asks the expert a specific question rather than asking for a full analysis of a situation. As with the "limited information tasks" technique, experts might be hesitant to give advice in an unfriendly environment.

Method of Tough Cases

This technique attempts to unveil subtle aspects of the expert's thought process. These aspects may be stimulated during the analysis of cases that are either too simple or too complex. In such situations, experts are required to use knowledge and apply thought processes that are not common. Accordingly, experts provide additional insight and explanations of the domain's mainstream problem-solving knowledge.

Induction

The inductive approach encourages an expert to refer to specific examples which consist of decisions and parameters considered in reaching those decisions. A computer program iterates based on a set of positive and negative examples to induce general rules. Those rules cover and exclude the training set of positive and negative examples, respectively. The efficiency of the induced rules lies in the selection of the relevant parameters and the training sets of examples.

Sources of Expertise for This Research

An objective of this research is to generate a conceptual knowledge base for the development of a knowledge-based system, as opposed to an expert system. Therefore, the experts do not need to be the very best persons in the field, but, rather, individuals with enough knowledge to solve problems well. In fact, this is the case for many KBSs in development today.

The construction industry is such that often a good schedule is the product of multiple inputs. Thus, it is important to find and fuse knowledge from a number of experts. The advantages of using a diverse collection of experts, as outlined by [Mittal 85], may offset the extra time required to resolve conflicts and contradictions generated by multiple inputs. In fact, contradictions may be turned into advantages as they indicate areas for further research.

Four groups provided the construction scheduling expertise drawn upon for this project: (1) contractors, (2) owners, (3) consultants, and (4) in-house staff. W. E. O'Neil Construction Company, a large building contractor in Chicago, Illinois, collaborated on this knowledge engineering study by committing one senior project manager and one junior project scheduler to the development of the knowledge base. Senior representatives from the U.S. Army Construction Engineering Research Laboratory (USACERL) and a field engineer from the U.S. Army Corps of Engineers participated to articulate an owner's view. A senior consultant from Pinnell Engineering, Inc., who provides scheduling services to both owners and contractors, also took part in the knowledge elicitation process. Finally, expertise of several faculty members in the Department of Civil Engineering at the University of Illinois at Urbana-Champaign was drawn upon to contribute to the refinement and extension of both the contractor's and owner's views.

Knowledge Structure

The knowledge base consists primarily of scheduling decision rules, general construction knowledge, and project-specific knowledge.

The scheduling decision rules assure that the construction schedule complies with imposed general requirements, logic, time, and cost constraints. An example of a cost constraint: cash flow front-end loading is unlawful.

The general construction knowledge represents generic expertise that may be applied to a variety of projects. For example, a winter-sensitive activity should not be scheduled when ambient temperatures are expected to be below specified minimums.

The project-specific knowledge is specific to a particular type of schedule activity. For example, the enforcement of weather-sensitivity constraints on the hammock activity "install drywall" makes it possible to split it into three subtasks: metal studs, drywall, and taping.

Employed Techniques for Knowledge Acquisition

The approach to knowledge acquisition is based on successful methodologies adapted from [Freiling 85, Preran 87].

The techniques used here for capturing the knowledge base fall into three categories. The first involves analyzing textbook scheduling knowledge. This exploratory step is used to determine the breadth and depth of the construction schedule analysis domain. This phase identifies whether the initial and in-progress schedule analyses, as defined in this research, are sufficiently narrow and self-contained. The aim is not for a knowledge base intricately tied to other kinds of knowledge, i.e., automated schedule generation, work package risk identification, cost control, etc. Rather, the goal is to develop a knowledge base trustworthy in a limited domain area.

The second category is interviewing experts. The knowledge base produced in the previous phase is used to interact with the experts. By showing this knowledge base to the experts early in the knowledge elicitation process, it is possible to obtain a better understanding of the different kinds of expertise prevalent in the domain. In addition, the experts involved in the process better understand the scope and complexity of the project.

The third technique involves studying the experts' performance on specific problems. By limiting the information typically available to the experts and by constraining the problem experts are to work on, it is possible to stimulate intuitive knowledge. Following is a description of each of these techniques.

Textbook Scheduling Knowledge

To add structure to subsequent knowledge acquisition efforts, the first pass at a knowledge base is made by analyzing available texts and technical manuals on the subject [Avots 85, Clough 81, Gray 86, O'Connor 82, Ponce de Leon 84]. Figure 5 shows the evolution of textbook scheduling know-how into human "say-how." Appendix A describes this first-generation knowledge base, which consists of a categorized list of statements given in plain English.

The term "know-how" refers to the knowledge a person or computer possesses. The term "say-how" is used for that knowledge a person is able to articulate in plain English. The term "show-how" refers to the actual use of knowledge by a person or computer.

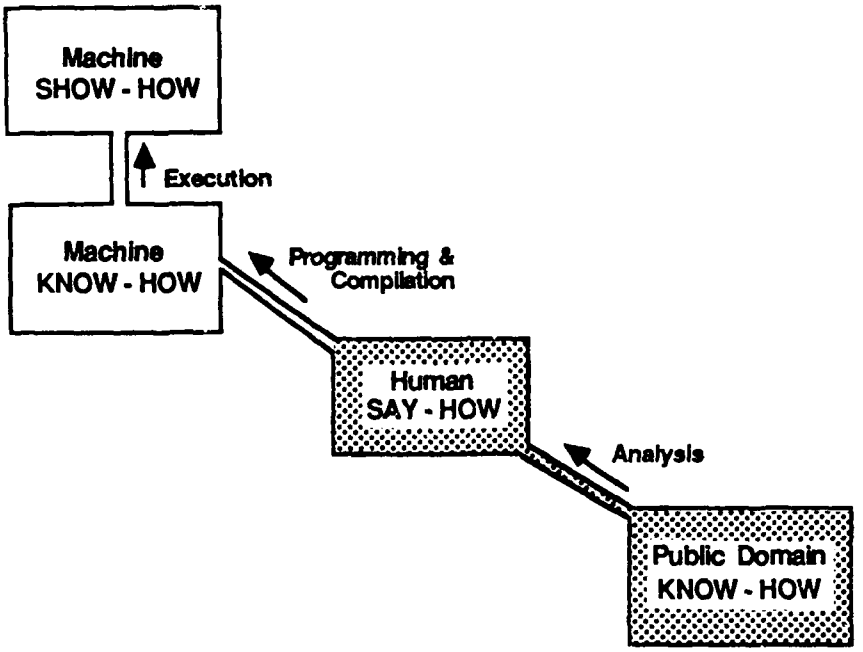

Figure 5. Analysis route map.

Interviews with Scheduling Experts

The dialogue acquisition technique is used to elicit and formalize the experts' knowledge. This method is depicted in Figure 6 as a knowledge engineer's route map. Its purpose is to convert human know-how into human say-how through a process of articulation, of which the expert is supposed to be capable. Once in this form, programming and compiling techniques can convert it into machine code, i.e., machine know-how. At run-time the computer generates the behavior the user expects, i.e., machine show-how.

In Figure 6 the articulation channel depicts what Artificial Intelligence researchers call the knowledge acquisition problem, also known as the "Feigenbaum bottleneck" [Feigenbaum 83]. Articulating knowledge is a complicated process because experts are expected to spell out their expertise, which is a task they are not used to or trained to perform. The narrowness of the bottleneck is related to the complexity of the task domain. [Michie 86] argues that the more complex the mental skill, the greater the proportion of it encoded in intuitive form and hence beyond the access of the expert.

Two experts from W. E. O'Neil Construction Co. were interviewed separately (one for 12 hours and one for 4 hours). The 16 hours' worth of interviewing were divided into four sessions of 4 hours each.

This technique is used mainly because it is assumed that senior project managers are confident about their ability to express the large body of pattern-based rules they know. The approach can be termed a "structured interview" since the knowledge developed from technical reports represents the first-generation knowledge base.

W. E. O'Neil's experts reviewed this first-generation knowledge base one entry at a time, making comments on each. This process led to the addition and deletion of entries, quantification of entries, and addition of categories. The knowledge gathered during the interviews is presented in Appendix B.

The combination of this approach and already published knowledge produced a more encompassing series of high-level concepts, i.e., the second pass to human say-how. However, the two experts often could not describe how much they know, how they keep track of it, or how they know when to use which information.

In light of the partial success of this technique, a less direct approach to knowledge acquisition is required.

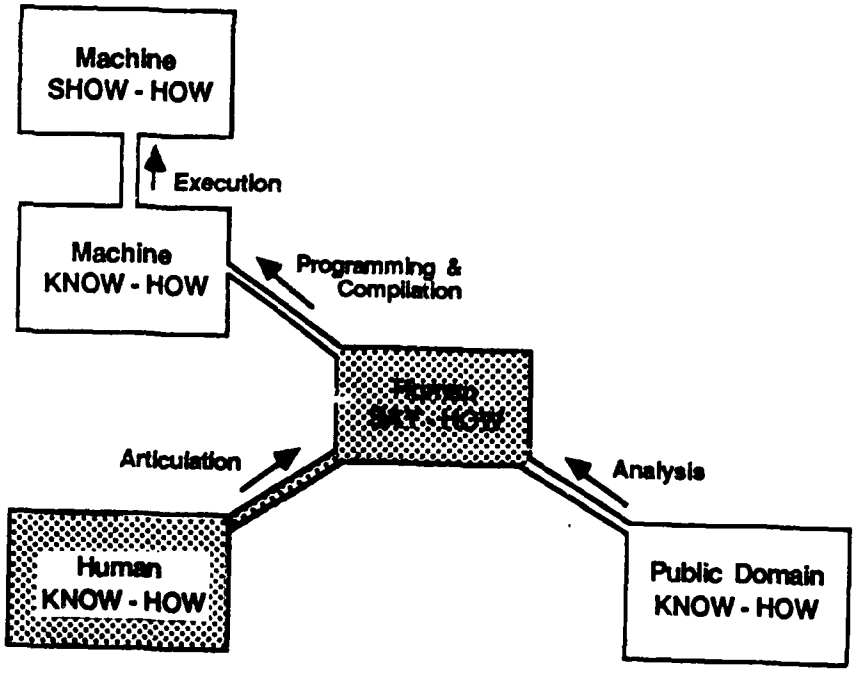

Figure 6. Dialogue route map.

Schedule Evaluation Exercise

A third alternative for eliciting knowledge assumes that the route map in Figure 6 is incomplete and that there is some other way of going from human know-how to machine know-how. Figure 7 shows a more thorough map than its counterpart in Figure 6. In this case, it is possible to proceed from human know-how into tutorial human show-how by stimulating the knowledge and reasoning skills of the experts. This process generates human-supplied advice. Moreover, it is possible to go from the second-generation human say-how to human show-how by hand simulating the application of this knowledge base. After having arrived at this point, a model of the experts' skill in explicit form can be translated into third-generation human say-how rules. Those rules can be subsequently compiled into machine know-how. This approach to knowledge acquisition provides a passage to circumvent the articulation bottleneck of Figure 6.

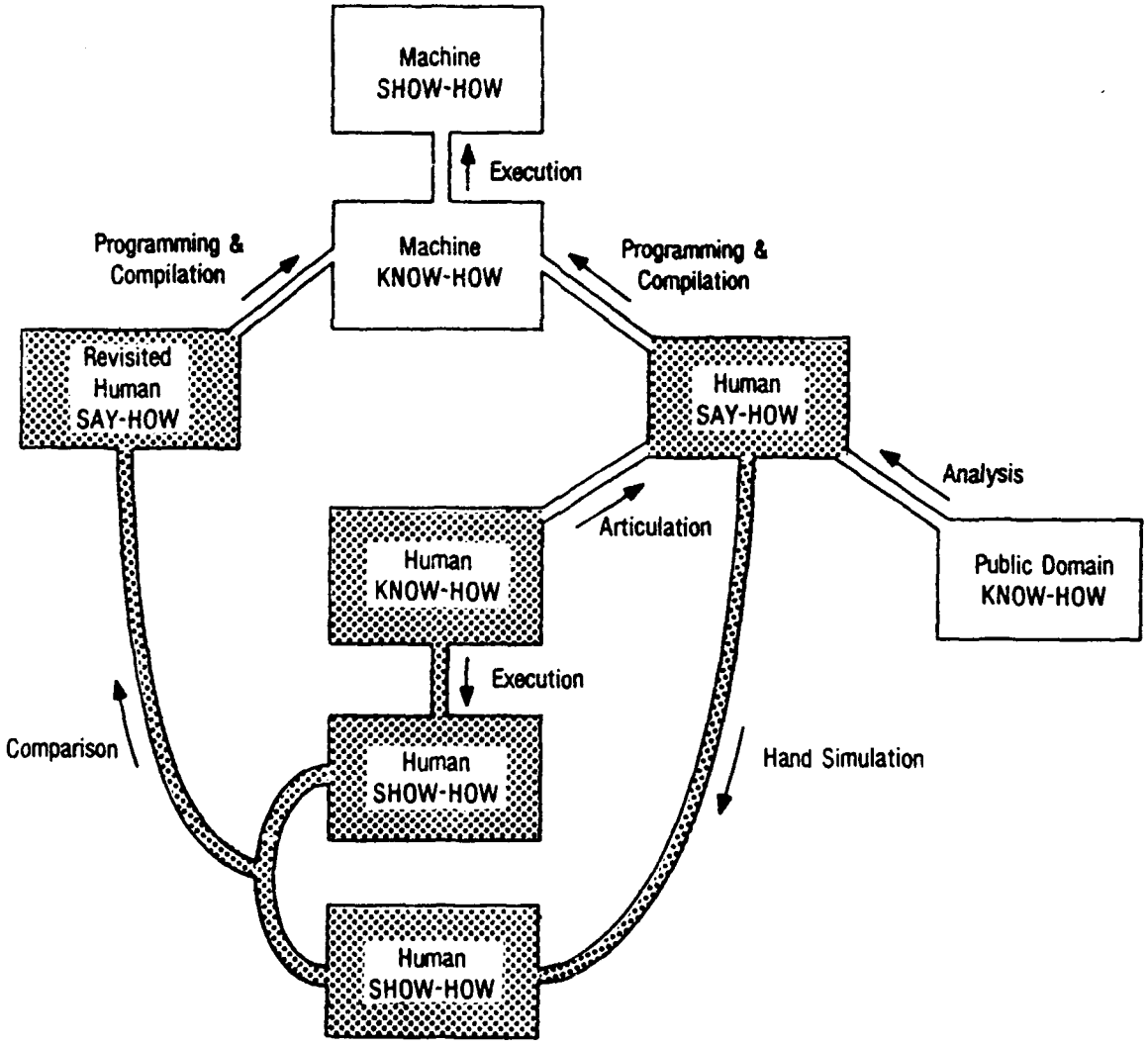

Figure 7. Execute route map.

Exercise Design. The critical processes for implementing this approach are represented by the bridges that link the human know-how and say-how with the human show-how, i.e., the execution and hand simulation channels. In order to simulate these channels, this study included an experiment using two video cameras, three senior project managers (experts), a rookie project manager, and a curtain. The aim was to mimic a computer by having the experts act as the KBS, the curtain act as the computer screen, and the rookie act as the user.

This exercise had the following primary goals: (1) to push the knowledge base toward completeness by eliciting new knowledge; (2) to validate, fine tune, and disprove existing knowledge; (3) to develop a model of interaction between the user and the knowledge-based computer system; and (4) to determine the practical feasibility of using this knowledge elicitation technique to capture construction scheduling knowledge.

In order to keep tight control on the information exchange between the rookie and the experts, the actual setting consisted of two separate rooms. In one room the rookie conducted a construction schedule audit. He used only the reports generated by one or two PMSs and the advice coming from the experts. Generally such PMS reports contain different sorts of information associated with each schedule activity, such as early start dates, late start dates, float, cost, successors, predecessors, duration, and description.

While the video cameras recorded the approach, the experts, who were located in a separate room, guided the rookie in the schedule analysis. The question/answer communication protocol between the rookie and the experts was implemented in electronic mail messages within a Local Area Network (LAN).

Figure 8 shows the physical layout and location of each participant. This researcher's role (depicted as the knowledge engineer) was one of observing, recording, and interpreting. Figure 9 illustrates the electronic mail message routing. In this routing scheme, this researcher was able to monitor all mail communication sent among the trio, rookie, and human say-how analyzer.

The amount of information available to the three experts was restricted. They had only a tablet and a pencil with which to work. The hypothesis was that if such project schedule information is withheld, we can force the experts to rely heavily upon, and provide additional evidence about, their knowledge and reasoning skills [Hoffman 87, Kneale 86]. By looking across the experts' tactics and procedures, one should be able to know what data they like to have available and the information that is produced.

While the trio and rookie interacted, an attempt was also made to use the existing rules and procedures which had been derived from the articulation approach and the transformation of textbook scheduling knowledge. A fourth individual conducted such assessment. This hand simulation process provided the knowledge engineer with the opportunity to simultaneously examine the reasoning of the second-generation knowledge base. This comparison identified points of disagreement and consensus, and generated a more solid human say-how.

Three points of view were fused during the exercise--those of (1) the contractor, represented by W. E. O'Neil Construction company; (2) the owner, portrayed by the U.S. Army Corps of Engineers; and (3) the project management software developer, represented by Pinnell Engineering, Inc.

Since the primary objective was to stimulate and elicit knowledge from the trio of experts, the rookie was as knowledgeable as the experts. However, the rookie pretended to have no construction experience. In this fashion, the rookie was able to steer the exercise in the direction that would best stimulate the trio's expertise. Appendix C contains in plain English the knowledge elicited during this exercise. Appendix D contains a few examples of electronic mail messages.

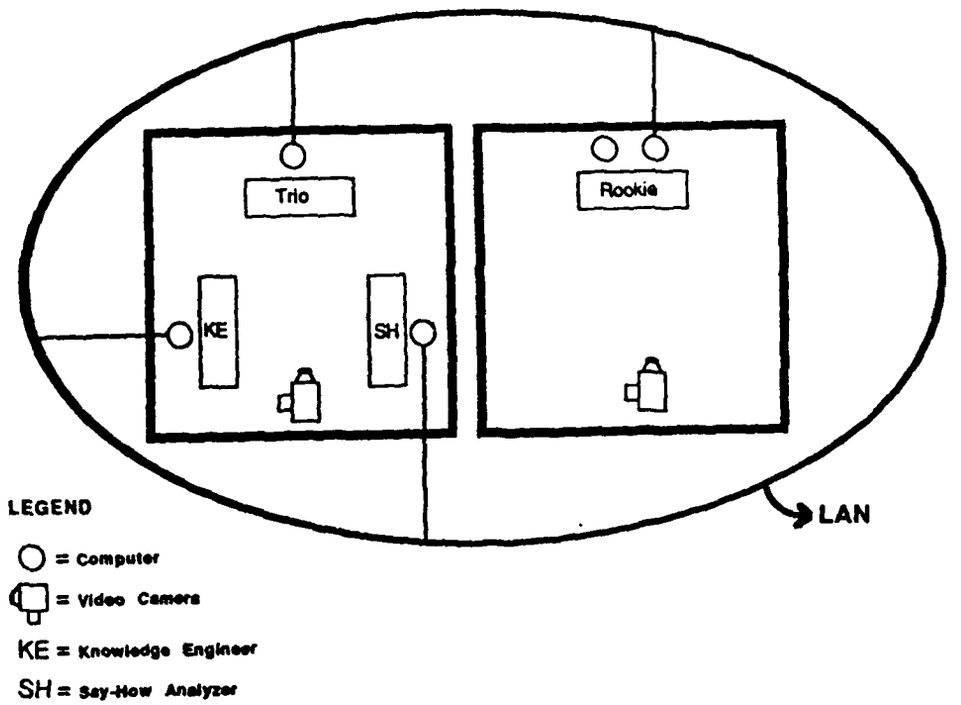

Figure 8. Exercise setting.

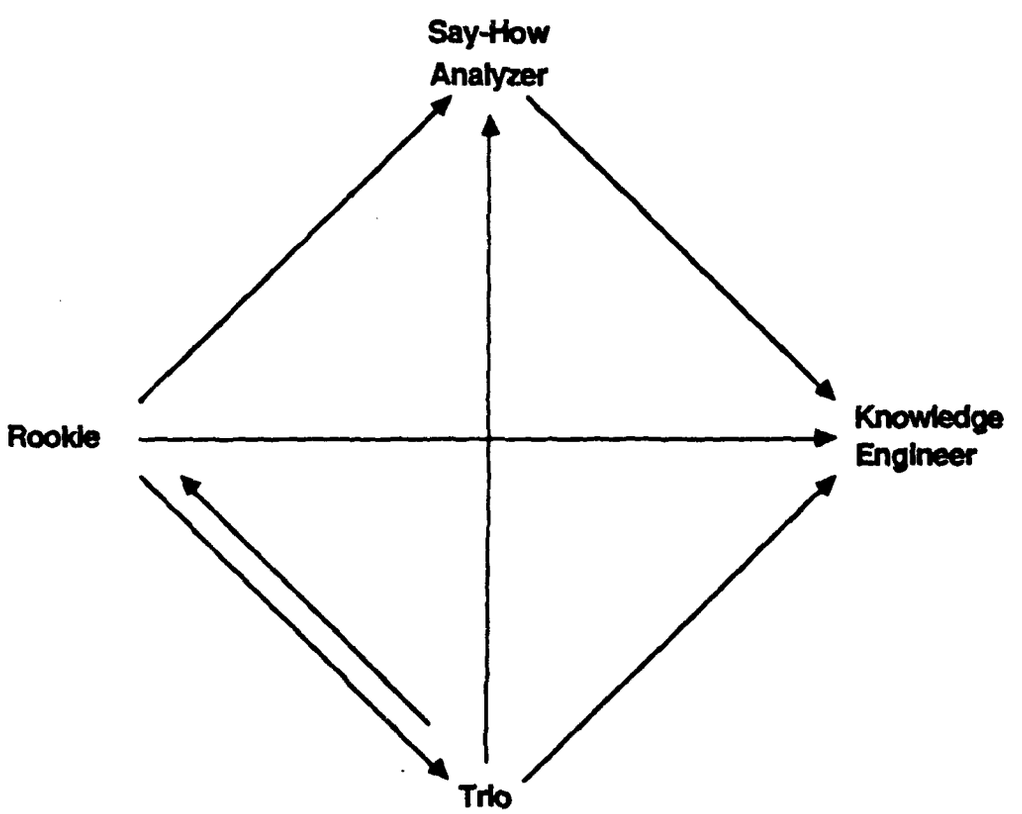

Figure 9. Electronic message routing map.

The 2-day exercise was divided into five phases: (1) ice-breaking, (2) initial schedule analysis, (3) in-progress schedule analysis, (4) overall schedule analysis, and (5) de-briefing.

The Ice-Breaking Phase. One of the criteria that clearly would affect the outcome of the exercise was the level of interaction among the trio of experts. Thus, the intent in this ice-breaking phase was to introduce the heterogeneous trio members to each other and allow for free interaction.

During this period, the trio were presented with an overview of knowledge-based systems, the objectives of this research project, a detailed description of the mechanics of the exercise, and a general statement regarding their expected contribution. They also had the opportunity to get acquainted with the Banyan local area network. Four hours were devoted to this stage.

The Initial Schedule Analysis Phase. During this phase, the rookie used the schedule of a low-rise medical dispensary building. The following general characteristics describe the construction schedule to which the rookie had access: (a) notation = activity on the arrow; (b) number of activities = 175; (c) assumed location = Champaign, Illinois; (d) cost = $600,000.00; (e) start = 1 July 1976; (f) finish = 14 August 1977; (g) structure = steel. The PMS utilized is Primavera Project Planner™. Three hours were allocated to this phase.

The In-Progress Schedule Analysis Phase. During this phase, the construction schedule is for a midrise, six-story, two-wing apartment building. The following general characteristics describe the construction schedule to which the rookie had access: (a) notation = activity on the node; (b) number of activities = 143; (c) location = Champaign, Illinois; (d) cost = known; (e) start = 1 June 1986; (f) finish = 27 October 1987; (g) structure = steel. The PMS used was PMS80™. Three hours were assigned to this stage.

The Overall Schedule Analysis Phase. This phase was designed specifically to address automatic schedule generation issues, as well as schedule logic analysis. No specific construction schedule was utilized by the rookie. Rather, the rookie and trio electronically conversed at a conceptual level only. Two hours were devoted to this period.

The Debriefing Phase. Because of the uniqueness of this technique's implementation for eliciting knowledge, it was of paramount importance to debrief the trio and rookie. This feedback will be used to modify future attempts at knowledge acquisition. The debriefing lasted 1 hour.

Exercise Mechanics. The methods by which knowledge was simulated represent an important aspect of knowledge acquisition techniques. The mechanics of eliciting knowledge with the novel implementation of the "limited information" technique play an important role. What follows is a retrospective analysis of such means in the form of issues, suggestions, and comments.

1. Because of the exercise's limited information approach implementation, the experts were at times bothered by the fact that they had to request all the information they wanted. They would have preferred a narrative description of the project to assimilate as much implicit detail as they could. This would have defeated the purpose of using the limited information approach, however, because the trio would not have been cognizant of all the facts and relationships being digested.

2. A more formal question/answer communication protocol was desired by both the trio and the rookie.

3. The experts should have provided the rookie with "words of advice" and let the rookie perform the necessary hand calculations to implement such advice.

4. The experts were frustrated by the number of hand calculations they had to perform. This was the case when they were provided with large sections of PMS reports. They were expected to assimilate both explicit and implicit information immediately.

5. The roles of the experts and rookie were dynamic. While the rookie was trying to stimulate and extract as much knowledge as possible from the efforts, they were trying to obtain relevant information from the rookie. This caused identity confusion a few times.

6. Sections of PMS reports were intentionally given to the experts to point out the disadvantages of using nonlimited information approaches to knowledge elicitation. When they had a PMS report containing all critical activities with basic parameters, they spent a considerable amount of time analyzing the report in silence. During these circumstances, their "thought process" was simulated but not captured. The experts were not thinking aloud. Thus, answers were not provided to questions about the kind of data relationships were they looking for, why they were looking and what was irrelevant information.

7. The experts' advice was not always derived directly from the information supplied by the rookie. Similarly, the rookie was not always provided with the justification for all information requests. In such cases, the rookie was forced to ask why and how. The trio would then characterize the rookie as stupid, obstinate, and too demanding. It is evident that these adjectives reflected the frustration of the experts when they had to think very hard to articulate their hidden knowledge patterns. Their level of frustration was an indication of when they were accessing deep knowledge to provide further demanded justifications.

8. Because the trio were required to think thoroughly and type all questions and answers, electronic mail was considered slow from a real-time feedback standpoint. This slowness also caused the trio and rookie to be asynchronized. This lack of synchronization decreased the efficiency of the trio. Their thought process was constantly interrupted with follow-ups to issues brought up earlier in the exercise.

9. The mechanics of reading, sending, and printing messages and attachments was not very efficient. It is likely that some scheduling knowledge was hindered by this deficiency.

10. The coordination of the "subject:" slot in the template of each electronic mail message was also difficult. The message's reply function on the LAN electronic mail system does not include the body of the message to which it is replying.

11. The trio began adjusting to the exercise mechanics towards the end of the 2-day exercise.

12. To release some frustration built up by the inability of the trio to converse with the rookie, the electronic communication protocol should allow for a "chat" mode between the experts and the rookie.

13. The chat mode in the communication protocol would have allowed the trio and rookie to exchange vague phrases like "you know what I mean." This would have made the exercise more realistic in terms of person-to-person communication, but not how someone interacts with a computer.

14. The time necessary for the trio to build a mental picture of the project was substantial.

15. The idiosyncrasies of a particular project were not part of the exercise. The trio found it much easier to discuss scheduling at a conceptual rather than specific-project level.

16. Three days' worth of knowledge elicitation time, instead of 3 hours, may be necessary in each of the schedule analysis phases (initial, in-progress, and overall). This indicates that only a fraction of the whole body of construction scheduling knowledge was stimulated.

Exercise Consensus. The consensus of the participants concerning the exercise can be summarized as follows:

1. A sufficient common body of construction knowledge can be meaningfully categorized, structured, and applied within a KBS to make it worth developing--even if only a reliable noncomputerized set of scheduling guidelines is developed first.

a. The trio summarized the level of difficulty in assessing the reasonableness of the schedule logic with the statement: "We have a bull by the tail." However, the trio also concluded that the logic aspect can be addressed at a sufficiently generic level and that the burden of structuring and collecting the knowledge will not render the development of a KBS infeasible. For example, a data base can be created with generic activity-sequencing patterns. Those patterns can take the following form: (predecessors, activity, successors).

b. In the short term a checklist approach for determining the reasonableness of the schedule logic for each building system and building type may be enough.

c. The process for evaluating schedule logic is more qualitative than quantitative. It also involves large amounts of knowledge for each building system.

d. The quantitative aspects of schedule criticism may be generalized to several building systems. For example a cost analysis technique may be applicable to more than one building system.

2. The limited information approach proved to be feasible and practical for stimulating and capturing knowledge.

3. The trio became aware of how difficult it is to articulate the thought process and the concepts they utilize intuitively.

4. Three operation modes for KBS can be implemented:

a. The KBS has access to all project information, performs all necessary calculations, and directly makes inferences based on those.

b. The KBS directs the user to perform hand calculations whose results are fed back to the KBS. The KBS makes inferences and suggests further analysis.

c. A hybrid approach may be necessary when the procedures have not been implemented in a way that the KBS can execute them; yet, the KBS is capable of reasoning about the results of the procedures.

5. The user of KBSs should be knowledgeable about construction, as opposed to being an apprentice. It is necessary for a user to detect when a KBS is giving bad advice.

6. Considering the problems caused by not understanding a construction schedule or by failing to enforce it, a KBS for construction scheduling criticism has a high payoff.

7. Owners and contractors share a large number of objectives, though there are a few that are conflicting. However, a single KBS for construction scheduling criticism may be used by both.

8. The knowledge elicitation exercise was interesting, unique, and frustrating. The discussion among the trio of experts was never constrained by the function each expert was representing.

The Human Say-How Analyzer. The consensus of the knowledge engineers concerning the hand simulation process is summarized as follows:

1. The performance of the second-generation knowledge base was tested during the exercise by comparing its recommendations with those of the trio.

2. The execution of the second-generation knowledge base was data driven, as opposed to the goal-driven approach followed by the trio. Therefore, it performed analysis on areas not explicitly covered by the trio, e.g., the search for multiple critical paths.

3. The second-generation knowledge base addressed a few of the concepts considered by the trio, e.g., front-end loading and weather sensitivity.

4. The second-generation knowledge base did not address many of the concepts covered by the trio, e.g., influence of liquidated damages on the project's criticality.

5. The exercise was designed with the primary idea of stimulating the trio's expertise. In the implementation of the exercise there were no provisions to facilitate the execution of the second-generation knowledge base.

The "Paper" Knowledge Base

The "paper" knowledge base consists of a set of conceptual schedule provisions written in plain English. These provisions are the result of analyzing and generalizing the knowledge produced by each of the three knowledge acquisition techniques. The provisions represent what construction schedules need to comply with, but provide no details about how they should be implemented. These provisions are synthesized from Appendices A, B, and C.

Abstraction Process

A retrospective analysis of the intuitive synthesizing process which produced the majority of the conceptual schedule provisions is outlined below.

For a given set of examples or versions of the same scheduling method, do the following:

1. For each entry in the set of examples find an underlying scheduling principle compatible with previous generalizations of the same scheduling method and that also includes the example under consideration. Repeat the process until all examples in the set are addressed.

2. If the synthesized scheduling provision of Step 1 does not seem to lend itself to adoption by some other firms, which did not participate in this knowledge engineering study, then continue its generalization. Repeat the process until there is an indication that the scheduling principle is general enough that companies not involved in the study might use it.

In order to illustrate the application of the synthesization process, the generation of Schedule Provision No. 1 is presented next.

Appendices A, B, and C contribute four examples which correspond to different methods of defining activities durations.

1. Activity duration limits usually range from 1 to 30 calendar days.

2. Reasonable activity durations range from 5 to 25 days.

3. Critical activities should have enough detail, that is, have a duration of less than 20 days.

4. A typical activity duration should be between 5 and 25 days.

The application of Step 1 to generate a scheduling principle corresponds to the following four iterations:

1st) Activities durations need to be constrained by a maximum duration.

2nd) Activities durations need to be constrained by a [minimum-maximum] range.

3rd) Same as 2nd.

4th) Same as 2nd.

The application of Step 2 to further generalize the scheduling provision of Step 1 corresponds to the actual Provision 1:

5th) All activities affecting progress should be included and defined in a way that they can be easily monitored and measured. The total number of activities should remain manageable.

General Requirements

1) All activities affecting construction progress should be included and defined in such a way that they can be easily monitored and measured. The total number of activities should remain manageable.

2) Activities should be described in a way that computers and anyone familiar with the construction work can understand.

3) Subcontractors responsible for significant phases of the project should participate in the contractor's developmental plan.

4) The contract schedule and technical specifications that affect the construction schedule should be met.

5) External and project-independent constraints should be an integral part of construction schedules.

6) A schedule should be flexible enough to incorporate changes.

7) Milestones should be an integral part of construction schedules.

8) Activity-on-the-node diagrams are more desirable than activity-on-the-arrow diagrams.

Time

9) The construction schedule should comply with the overall contract start and completion dates.

10) Managing simultaneous critical paths is difficult and should be avoided whenever possible.

11) Float should be broad enough to support the premise that it has not been manipulated.

12) Float should be considered as any other resource and, thus, it may be bought by either the owner or the contractor depending on who submitted the initial construction schedule.

13) An activity is considered critical if its first likely delay causes a delay in the overall completion date.

14) The estimated duration of activities should reflect the season of the year in which the activities are to be executed.

15) Schedule projections should be based on comparisons between what was planned and what actually happens.

16) Assessing progress on an activity should make both the percentage complete and the expected real remaining duration consistent.

17) If there is a schedule slippage, the entire path to which the lagging activities belong should be monitored, even if it contains noncritical activities.

Cost

18) Cash flow front-end loading should be avoided.

19) The monetary value assigned to each activity should represent a reasonable amount for that work.

20) The monetary value represented by the construction schedule should comply with the total contract amount.

21) The monetary value associated with an activity should play little role in constraining its duration.

22) The decision to include the cost of materials with the cost of their installation should be based on (a) whether or not the owner wants to reimburse for materials soon after they arrive at the site and (b) the ratio of cost of materials to total activity cost.

23) Progress payment requests should be reasonable and based upon scheduled activities which are in progress.

Logic

24) An activity that is weather sensitive should not be scheduled in a period when weather conditions are expected to be below specified minimums or above specified maximums.

25) An activity is weather sensitive if its materials and/or labor are affected by either water, temperature, or moisture.

26) Award, submittal, approval, and procurement lead times should be an integral part of construction schedules.

27) Activities' sequencing and interdependencies should be logical and represent a reasonable plan for accomplishing the work.

28) Activities may be overlapped in varying proportions. The primary consideration is to maintain continuity of production in the activity.

29) The contractor should revise the schedule when it no longer accurately represents the plan for completing the remaining work, or when actual progress has deviated from the planned schedule.

30) When an interim schedule is approved, it implies the acceptance of a practical and logical way to finish the remaining work on time.

31) For every schedule change, logic, time, and cost impacts should be assessed.

32) The logic, cost, duration, and number of activities on the critical path should be reasonable.

33) The building enclosure should be logically related to weather sensitive activities.

34) A building is considered enclosed when weather-sensitive work can proceed and the building can be heated.

Pool of Procedures

These procedures represent ways of interpreting "paper" knowledge base conceptual schedule provisions. Their identification is a by-product of each knowledge acquisition technique used in this study. No emphasis has been placed upon their generation--the experts produced them spontaneously as they attempted to clarify scheduling concepts. As a result, the number of procedures is rather small. Table 1 relates these procedures with their respective conceptual schedule provisions. Chapter 5 identifies some of these procedures and attaches them to "paper" knowledge base schedule provisions. This association represents yet another step in the knowledge metamorphosis process.

Table 1

Matrix for Schedule Provisions and Procedures

Provision No.	Appendix A	Appendix B	Appendix C
1.	5. 8, 9, 12, 15	3, 6, 7, 18, 22, 30, 34	2.2.a-c, 3.3, 3.4.a-e, 3.6.a-b, 9.1, 10.2.c, 10.4.a
2.	1. 2, 3, 4, 12	2, 3, 5	
3.	6	4, 27, 35	1.3, 3.5.a, 9.5
4.	7, 8, 12		1.1.c, 1.4, 1.6.d, 4.2
5.	7, 9, 16	37	1.5, 9.2, 9.4, 10.6.a
6.		36	
7.		17, 28	1.2.b, 2.1
8.	1	1	
9.	7	54	1.6.a-b
10.	11	13, 57	
11.	10	8. 14, 15, 16	2.6.a-b
12.	10	7, 9, 10, 11, 14	
13.		12	2.5, 5.4.d
14.	8	20	5.4.c
15.	25, 27, 28	59, 60, 61	5.4.a-b, 5.5.a-b, 6.4, 6.5
16.	21. 22, 23	50, 51, 52	5.6.a-b
17.	26	53, 62	5.1.a-c, 5.3.a-d, 6.1.a-b, 6.2.d, 6.4, 6.5, 7.1
18.	20	42, 43, 44, 45, 46, 47	3.2.a-e, 3.6.a
19.	5. 15, 18, 19	49	3.1.a
20.	17	49	1.6.c
21.		19	
22.	19	45, 46, 47	
23.	29	38, 39, 40, 41	

Table 1

Matrix for Schedule Provisions and Procedures (Cont.)

Provision No.	Appendix A	Appendix B	Appendix C
24.	16	32	2.4.a. 5.3.c. 7.1. 7.2. 7.4.a-d, 7.3. 10.2.a-b, 10.6.b
25.		30, 31, 32	10.2.a-b, 10.7.e-f. 10.8.a
26.	13, 14	7, 23, 24, 25, 26, 27	1.2.a. 5.4.d, 9.3.a. 10.2.c, 10.4.b. 10.5.a
27.	15	14, 22, 30, 33, 58, 59	1.1.b. 1.1.d, 2.3, 2.7, 4.1, 6.2.b-c, 10.1.a, 10.2.d-g, 10.3.a, 10.5.b-c. 10.7.b-d. 10.8.a
28.		14. 29. 57	6.2.b-c. 10.2.d-g
29.	24. 31, 34. 35	48, 49, 52, 54, 55	5.2
30.	30. 31. 34. 35		5.2
31.	24, 32, 33. 34, 35	21, 56, 57, 58	1.1.a
32.		58, 59	1.1.b. 2.2.a-c. 5.3.c. 6.2.a-d
33.		21, 28. 32	6.3.a-b, 7.3, 7.4.a-d. 10.7.a
34.		21, 28. 32	7.4.a-d. 10.7.a

5 KNOWLEDGE ORGANIZATION

Objective

AI researchers and users agree that there is nothing magic about KBSs. They generate conclusions already explicit or implicit in their knowledge bases. A KBS, however, cannot be built without understanding the task for which it is designed. Peter Denning [Denning 86 p 20] of the NASA Ames Research Center wrote: "If we know of a procedure to find a procedure, we can program that. But if we know nothing at all, we are stuck." This phase's function, among others, then is to transform "paper" knowledge base conceptual provisions into executable procedures.

These procedures correspond to the expertise utilized within a firm. They represent one firm's way of performing schedule analysis. The implication is that the interpretation of one "paper" knowledge base conceptual provision may differ from one firm to another. The differences in interpretation may just be the degree of sophistication, complexity, or detail given to a provision. Achieving homogeneous interpretations across all firms is not as important as having such interpretations concur with the original intention of the conceptual schedule provision. This flexibility in interpretations is consistent with the idea that a company will not use a KBS whose knowledge base represents another company's way of working.

It is extremely difficult in practice to investigate and exhaust all possible ways of interpreting a provision. Thus, it is the purpose of this research to pursue only one interpretation for a few selected "paper" knowledge base provisions. The provisions selected for transformation are those for which the knowledge elicitation techniques produced a procedure for implementation.

The attachment of procedures to the "paper" knowledge base generates a more specific kind of knowledge base. Because many procedures draw on a common pool of descriptive project parameters, this newly generated knowledge base begins to exhibit some regularity in the sense that expressions of similar form reappear frequently. These regularities are identified and captured by building an object-oriented English-like knowledge acquisition grammar. This grammar allows the ordering and expression of facts, rules, and concepts from the construction schedule analysis domain.

The use of this object-oriented English-like knowledge acquisition grammar reduces the effort expended to acquire additional expertise. It provides the scheduling experts, who may not be computer programmers, with a language in which they can write their own knowledge. In this way, the experts play a more active and decisive role in the final task of knowledge implementation. In addition, the knowledge represented in this generic English-like notation can be distributed with modest effort to a variety of inference engine designs.

English-Like Knowledge Acquisition Grammar

[Hayes-Roth 85] chronicles the history and describes the implementation of IF...THEN production rules as a framework for knowledge representation. In essence, the structure of a production rule is as follows: IF <conditions are satisfied> THEN <perform these actions>. This methodology has proved natural for expressing many human problem-solving techniques. It has therefore been the method of choice for representing construction scheduling procedures.

The syntax for the categories <rule>, <condition>, <action>, and <frame> can be represented as follows:

<rule> ::= IF <conditions> THEN <actions>

<condition> ::= <frame> HAS <parameter> OF <value>

<condition> ::= <frame> IS IN CLASS <frame>

<action> ::= <frame> HAS <parameter> OF <value>

<action> ::= <frame> IS IN CLASS <frame>

<frame> ::= object represented with parameter-value pairs.

Unless otherwise stated, an implied AND joins every <condition> and <action> within the rule's left-hand side (LHS) and right-hand side (RHS), respectively.

An additional advantage of using this grammar is that it forces the experts to think of objects that can be described with numerous attribute-value pairs. For example, experts will identify and make references to construction scheduling objects, e.g., individual activities, classes of activities, clusters of adverse impacts on activities, etc. The benefit of asking experts to think in terms of production rules and objects will prove to be invaluable during the electronic representation of expertise in hybrid AI tools.

Provisions Selected for Transformation

The following examples correspond to conceptual schedule provisions, which are taken from the "paper" knowledge base, represented in the English-like grammar. The criteria to select a "paper" knowledge base schedule provision for transformation follow. The schedule provision should (1) illustrate the use of the English-like grammar, (2) show the potential of distributing the knowledge in this form to a variety of inference engine designs, (3) exemplify how the grammar may be used by experts to articulate further expertise, (4) have a procedure which has been elicited and articulated during the knowledge acquisition phase, and (5) provide a conceptual foundation for further refinement and for use by other provisions' implementation.

Example No. 1

English Grammar:

Provision No. 2. Activities should be described in a way that computers and anyone familiar with the construction work can understand.

English-Like Grammar:

1. Activity HAS NOT responsible-party OF {owner, contractor, subcontractor}.

2. Activity HAS NOT type OF {procurement, deliver, submittal, install, dummy, approval}.

3. Activity IS NOT IN CLASS {general requirements, site work, concrete, masonry, metals, wood and plastics, moisture-thermal control, doors, windows, glass, finishes, specialties, equipment, furnishings, special construction, conveying systems, mechanical, electrical, foundation, substructure, superstructure, exterior closure, roofing, interior construction}.

THEN

1. Activity HAS code OF {invalid}.

Comments. The title or description of an activity generally indicates of the type of trades and materials involved, its function, and its location. From this description construction managers familiar with the construction work are capable of inferring (1) the preconditions that constrain the start of the task, (2) the effects produced by the finish of the task (3) the different calendars governing the task, and (4) the external impacts on the schedule of the task.

How, then, should the title of an activity be represented so computer programs infer an equivalent level of construction knowledge? The human inferring procedure briefly described above implies an associative cognitive process and presupposes an experienced project manager.

A natural computer implementation of such a cognitive process can be developed based on the following premises: (1) that the experience of project managers, with regard to construction scheduling, can be stored for subsequent manipulation; (2) that the activity title can be semantically parsed; and (3) that once an activity has been decomposed into trades, materials, function, and location, it can be associated with the stored knowledge.

A semantic network, like the one described in *ART-Based KBS in Chapter 7*, can be used to store activity-independent construction scheduling knowledge. Coding schemes like the Construction Specification Institute (CSI), Building System Institute (BSI), and Colin Gray's precedence selection rules [Gray 86] may be used to parse the activity title. This mechanism is proposed in lieu of a computerized text scanner. Multiple semantic class membership is defined as the different activity characteristics that are determined. The multiple-class association would then provide the activity in question with additional inherited properties.

In this rule, for example, all activities are expected to be coded according to their responsible party, type, and CSI and BSI specifications. To illustrate the intent of the provision, the verification in this rule is only performed at the highest CSI and BSI division levels.

Example No. 2

English Grammar:

Provision No. 9: The construction schedule should comply with the overall contract start and completion dates.

English-Like Grammar:

1. Project HAS notice to proceed OF {start date}

2. Schedule HAS earliest start OF {< start date}

THEN

1. Schedule HAS notice to proceed OF {violated}

IF

1. Project HAS notice to terminate OF {finish date}

2. Schedule HAS latest finish OF {> finish date}

THEN

1. Schedule HAS notice to terminate OF {violated}.

Comments. Overall contract start and completion dates can also be interpreted as staggered site releases and early occupancy dates, respectively. Therefore, the computer implementation of this provision should necessarily have the ability to reason about calendar dates. The basic question is how to use the available AI hybrid tools to accomplish this task effectively. Frame-based knowledge representations languages can be used to decode calendar dates so that rule-based languages can manipulate it. A feasible computational conceptual process to achieve reasoning about calendar dates is outlined below:

1. A "decoded-time" schema is created with at least four slots: "day," "month," "year," and "day-of-the-week."

2. Each calendar date, e.g., 18OCT1987, is transformed into a literal symbol to be used later as the name of an instance schema.

3. Using the name generated in the previous step, a schema is created and defined as an "instance-of" the schema "decoded-time."

4. The original calendar date, e.g., 18OCT1987, is parsed to extract the values of the four inherited slots.

5. A rule that reasons about time should at least contain a variation of the following two LHS conditions.

IF

a. <some conditions>

b. (early-start ?activity ?calendar-date)

c. (month ?calendar-date ?month-value)

d. <more conditions>

THEN

a. <some actions>.

6. Because each calendar date has been defined as a schema, ?calendar-date, which in condition (b) is initially located in the slot value position, can be used subsequently in the schema value position.

7. The type of additional LHS rule conditions and the kind of term restrictions imposed on ?month-value, for example, is limited only by the developer's creativity.

These two simple rules basically constrain the time window for the project's schedule. They verify that no activities should be scheduled before the notice to proceed, and also that no activities should be scheduled after the completion time.

Example No. 3

English Grammar:

Provision No. 15: Schedule projections should be based on comparisons between what was planned and what actually happens.

English-Like Grammar:

IF

1. Activities ARE IN CLASS {CSI, BSI}
2. Activities HAVE status OF {in-progress, finished}
3. Activities HAVE assessment OF {slow progress}
4. Activities HAVE population size OF {> statistical minimum}

THEN

1. Activities HAVE delay factor OF {average delay of population}
2. Unfinished-activities ARE IN CLASS {CSI, BSI}
3. Unfinished-activities HAVE status OF {unfinished}
4. Unfinished-activities HAVE new duration OF {old duration * delay factor}.

Comments. Failure to incorporate the accumulated experience of a project on the revision of the completion date may result in unrealistic forecasts. On-going schedule projections should be based upon revisited (1) activities durations and (2) logic relationships. When projections are based on the analysis of activities' duration experience, three scenarios are viable. In the first case, groups of activities exhibit longer durations than originally anticipated. Second, groups of activities show shorter durations than initially planned. A plausible explanation for this phenomena can be found in the assumption that original activities' durations were intentionally extended in an attempt to sequester float. Finally, groups of activities perform as planned. Meaningful schedule projections should consider the first and second scenarios.

Estimates of the final completion date can also be based on analysis of "as built" logic relationships. The intended purposes of logic-based projections should be (1) to reflect the contractor's preferential logic and (2) to neutralize the effects of any artificial lead/lag factors built into the original plan.

In this section example, previous job experience with a particular class of work activities is scrutinized for one overall delay factor. If found, that modifier is then applied to all subsequent unfinished activities in that class to develop a new projected schedule duration. The size of the population has to be statistically significant for the average delay factor to be meaningful. This rule could also have been implemented in a slightly different way: by first identifying the classes that contain unfinished activities and then calculating delay factors for only those classes. In either case, the same rule's design may be applied to activities experiencing faster-than-planned progress. Thus, the average delay factor can be greater or less than one. If the same class contains sets of faster and slower activities, the delay factor of the slower set should prevail.

Example No. 4

English Grammar.

Provision No. 18: Cash flow front-end loading should be avoided.

English-Like Grammar:

IF

1. Activity HAS responsibility OF {contractor}
2. Activity HAS type OF {installation}

or

1. Activity HAS $/day index OF {> 90 percentile}
2. Activity HAS early start OF {first quarter}
3. Activity HAS $/day index OF {< 10 percentile}
4. Activity HAS early start OF {fourth quarter}

THEN

1. Activity HAS front-end criteria OF {yes}.

Comments. Two questions deserve consideration: (1) What are some of the possible ways to detect front-end loading in a schedule? (2) How can contractors be discouraged from front-end loading construction schedules?

The application of the object-oriented knowledge acquisition grammar to Provision No. 18 represents an alternative method to front-end loading detection. This provision's implementation correlates very high $/day index values with early-scheduled start times and very low $/day index values with late-scheduled start times. As defined, the rule groups all activities regardless of the work class to which they belong. Additional constraints may be added to the rule's LHS to further limit its scope, for example, to limit it only to concrete-related items.

The following examples correspond to other scenarios which could be used in lieu of or to complement the one described above.

1. Calculate the cumulative cash flow "S" curve. For midrise buildings it would show one-fourth of the cost committed at the end of the first third of the project, and three-fourths of the cost at the end of the second third. This measure should be applied with caution. The fact that the cost committed at one-third and two-thirds of the project schedule is greater than one-fourth and three-fourths of the project cost, respectively, also is an indication that the last one-third of the schedule is too conservative and that the preceding two-thirds of the project schedule were scheduled too optimistically.

2. Assess whether the mobilization costs are reasonable, given the contractor's preliminary equipment list.

3. Determine if there are items that are likely to overrun quantities. If so, assess whether their costs appear reasonable in relation to similar items with larger quantities.

4. Examine the unit prices of early activities to determine if they are greater than unit prices of similar activities scheduled towards the end of the project.

5. Determine the amount of work subcontracted. The higher the amount of such work, the less likely there is front-end loading.

6. Target labor-intensive activities for detailed scrutiny with any of the methods cited thus far.

7. Determine if the cost associated with each activity falls within an acceptable minimum/maximum percentage range.

On the other hand, owners can use specific mechanisms to dissuade contractors from front-end loading construction schedules. The following examples provide broad guidelines to accomplish this.

1. The monetary aspect of the bid should be determined on a net present value (NPV) basis. This practice heavily penalizes front-end loading. However, note that it also discourages early completion of noncritical activities.

2. Provide a financial investment mechanism, e.g., U.S. bonds, for all retainage.

3. The cost of expensive materials and equipment should be separated from the cost of their installation.

Example No. 5

English Grammar:

Provision No. 24: An activity that is weather sensitive should not be scheduled in a period when weather conditions are expected to be below specified minimums or above specified maximums.

English-Like Grammar:

IF

1. Activity HAS resource requirements OF {labor, materials}

2. {labor, materials} HAS affected by OF {water, temperature, moisture}

THEN

1. Activity HAS sensitivity OF {weather}.

IF

1. Activity IS IN CLASS {CSI, BSI}

2. Activity HAS sensitivity OF {weather}

3. Activity HAS weather conditions OF {present}

4. Activity HAS weather protection OF {absent}

THEN

1. Activity HAS assessment OF {likely delay}.

Comments. Weather sensitivity raises three questions: (1) What makes an activity sensitive to certain climatic conditions? (2) How can an activity be scheduled so its weather constraints are not violated? and (3) How can the concept of weather sensitivity be computerized?

A finding of this research has been that an activity is sensitive to weather when the following conditions occur:

1. Water is one of its resources; e.g., the water component of concrete either freezes or evaporates under extreme cold or heat.

2. Its resources react chemically with water; e.g., drywall expands in the presence of water or in high humidity.

Each activity in the schedule should have attached a weather profile for all climate conditions (e.g., rain, temperature, humidity, etc.) that might inhibit the execution of the activity. This weather profile should define the range within which an activity may be executed in normal conditions. The range may be specified in terms of time-dependent minimum and maximum values.

Once a CPM time window has been assigned to each activity, the expected weather conditions to which an activity is sensitive may be compared with those of the activity's appropriate weather profile.

In the examples of this section, the intent of the first rule is to characterize an activity as sensitive or insensitive to weather conditions. Being sensitive to weather does not automatically cause a delay in an activity. The second rule verifies that weather conditions are present and that the activity lacks weather protection. A further refinement of this interpretation could, for example, make sure that the weather conditions be present for more than a specified fraction of this activity's free float.

Example No. 6

English Grammar:

Provision No. 26: Award, submittal, approval, and procurement lead time should be an integral part of construction schedules.

English-Like Grammar:

IF

1. Activity-i IS IN CLASS {activities}

2. Activity-i HAS type OF {submittal}

3. Activity-i IS IN CLASS {CSI}

4. Activity-j IS IN CLASS {activities}

5. Activity-j HAS type OF {approval}

6. Activity-j IS IN CLASS {CSI}

7. Activity-i HAS NOT followed by OF {activity-j}

THEN

1. Activity-i HAS submittal approval logic OF {bad}.

Comments. Lack of coordination between material approvals and deliveries and their respective installations is among the most typical cause for construction delays. Two alternatives are feasible for keeping track of award, submittal, approval, and procurement activities. In the first scenario, all activities relevant to the materials procurement cycle form an integral part of the construction schedule. In the second case, previously stored project-independent knowledge about the materials procurement cycle is inherited by special activities as they are attached to a semantic network.

The rules used in this section correspond to implementations of the first alternative. The intent of the first rule is to verify that something that has been submitted has been approved. It may be the case, though, that the submittal activity includes the approval phase, in which case, splitting the activity in two is recommended. The same rule's design can be used to test the other nonconsumption activities.

Discussion

The representation of these six schedule provisions provides better insight into the intention and design reasons behind the generic object-oriented English-like knowledge acquisition grammar. It supplies a set of ground rules with which to discuss ways to automate the provisions and provides insight into the nature of scheduling knowledge.

6 KNOWLEDGE REPRESENTATION

Objective

This chapter describes knowledge representation issues in terms of (1) the search for a programming language environment, (2) the functional requirements for the preferred computing platform, (3) the set of mappings that bridge the gap between the object-oriented English-like grammar and the syntax of the selected language, and (4) the actual electronic knowledge representation of scheduling concepts.

Programming Language Survey

The implementation of construction scheduling issues requires the selection of commercial KBS development tools that provide standard ways for representing knowledge. In this section, major off-the-shelf application products available to support the encoding of a KBS are described. The tools considered are all serious development environments that run on high-performance workstations or minicomputers. Software that runs on microcomputers is not considered in this survey, although two computer packages have been chosen for both prototype and operational system development. The selection criterion for the two microcomputer-based tools (Personal Consultant Plus and GoldWorks) was in essence simple: Among the current commercially available tools, select the one that more closely incorporates the language features of the four major tools considered herein. Since tools are always evolving, this criterion is time dependent; thus, future versions or new software may include features that are currently absent.

The products described herein include ART, KEE, Carnegie Group's Knowledge CraftTM, and Teknowledge's S.1TM. They represented commercially available state-of-the-art systems when this research began in mid-1985. The criteria for tool evaluation are limited to language features for representing knowledge and to control and inference mechanisms. More comprehensive evaluations which include not only basic features but also development environment, functionality, support, and cost factors are found in [Mettrey 87, Richer 86].

The features for representing symbolic knowledge are facts, definitions, heuristic judgments, and procedures. As described in Chapter 2, the techniques available to represent these features include predicate calculus, logic programming, rules, semantic networks, frames, and a combination of these to generate hybrid programming systems. The built-in inference mechanisms include pure deduction, backward chaining, forward chaining, class inheritance, and message passing.

A summary of the basic features of the four tools considered is presented next. No overall rating is given to the systems reviewed because too many nonconsidered factors are involved for such a rating to be useful. An unresolved issue is whether choosing among these four alternative tools is a matter of programming style or if it makes a significant difference in solving the construction schedule analysis problem. What follows is a summary of each tool's features.

ART (release 3.0)fR [Clayton 87, Williams 85]:

- Rules: Variables, complex pattern-matching.
- Logic: Propositions, logical dependencies, and extents.
- Frames: Taxonomic and user-definable inheritance, methods, active values, object-oriented programming system.

- Inference and Control: Data driven, forward and backward chaining, agenda, hypothetical and temporal reasoning.

KEE (release 3.0)fR [IntelliCorp 85]:

- Rules: Variables, fully integrated with frame system.
- Logic: Tell and Ask language integrated with rules and frames.
- Frames: Taxonomic and user-definable inheritance, methods, active values, and slot value constraints; well-integrated object-oriented programming system.
- Inference and Control: Forward and backward chaining, hypothetical and temporal reasoning.

Knowledge Craft (release 3.0) [Buday 86]:

- Rules: OPS-5 rule language that can reference frames.
- Logic: Prolog equivalent language.
- Frames: Taxonomic and user-definable inheritance, methods, demons, contexts, and slot value constraints; object-oriented programming.
- Inference and Control: Backward and forward chaining, and agenda.

S.1 (release 2.1) [Schindler 86]:

- Rules: Object-attribute-value triplets and certainty factors.
- Frames: Very limited, no inheritance, and not easily extended.
- Inference and Control: Backward chaining with uncertainty, control blocks (very high-level procedural language), assumption-based reasoning.

Functional Requirements for the Inference Engine

The most basic requirements are that the inference engine be able to adequately represent the knowledge and perform the operations involved in the construction schedule criticism application.

The nature of the construction scheduling domain calls for a knowledge representation that is object-oriented. Schedule activities, precedence relationships, and resources, among others, are all construction concepts suited for attribute-value definitions as well as for hierarchical representation. Object-oriented programming provides the facilities to structure information which describes a physical item, a concept, or an activity. Each object is represented as a frame, containing declarative, procedural, and structural information associated with the object.

Construction schedule criticism is also a data-driven, opportunistic process. It is data driven because the process goes from schedule information to detect as many inconsistencies as possible. It is opportunistic because resolution of inconsistencies is subject to the satisfaction of all other scheduling constraints. These constraints may or may not be activated at the time of satisfying a given scheduling constraint.

Targeted Inference Engines

Three different programming environments have been actually chosen for representing scheduling concepts. They are (1) PC Plus, (2) ART, and (3) GoldWorks.

The first set of ideas was implemented at USACERL on a personal computer in 1985. PC Plus was the tool selected. Its important features included external program interface, certainty factors, help and explanation facilities, and a limited frame representation language. The system's inference control was mainly backward chaining. It lacked a pattern-matching language as well as relational data base capabilities.

ART was subsequently selected by the University of Illinois Construction Engineering Expert Systems Laboratory (CEESL) as the high-performance inference engine to process the knowledge. As described before, ART's knowledge representation language supports the expression of a wide variety of different types of problem-solving knowledge. This representation power, combined with its standard mode of data-driven inferring, provided a good match for the functional requirements of the construction schedule criticism domain.

GoldWorks is (as of second quarter 1988) the only microcomputer-based tool that in many respects provides similar capabilities to those found in ART. Broadly speaking, GoldWorks lacks only ART's Assumption-based Truth Maintenance System and the object-oriented graphics. It has been adopted by the USACERL as the next advanced computing platform. Such software is currently being used to develop the operational KBS described in Chapter 8.

Mappings

These mappings relate the object-oriented English-like knowledge acquisition grammar to the knowledge representation language syntax. A different set of mappings needs to be designed for every different inference engine.

A large number of features are typical of reasonably complex inference engines like ART. As a result, the mappings defined in this section should not be expected to produce ready-to-execute code. Rather, they produce portions of code and suggest or imply additional pieces that need to be present to make it all work.

Because the knowledge represented in the PC Plus pilot system followed the rapid prototyping path of Figure 3, and because GoldWorks has just begun to be explored, only a set of mappings tailored to meet the basic ART specifications has been defined in this research.

The essential mappings are shown below:

English-like ::= ART

<rule> ::= (defrule <rule-name> {<documentation>}

{(declare (salience <value>))}

<conditions>

=>

<actions>)

<frame> ::= <schema-name>

<parameter> ::= <slot-name>

<value> ::= <value>

<symbol> ::= <restriction>

<variable> ::= <restriction>

<condition> ::= (<slot-name> <schema-name> <value>)

<condition> ::= (<restrictions>)

<action> ::= (assert (schema <schema-name>

(<slot-name> <value>)))

<action> ::= (modify (schema <schema-name>

(<slot-name> <value>)))

<action> ::= (retract (schema <schema-name>

(<slot-name> <value>)))

<action> ::= <LISP function>

<action> ::= (assert <restrictions>)

<action> ::= (bind <restriction> <LISP function>)

"Defrule" indicates that this construct is an ART rule. "Rule-name" is a symbol that identifies the rule and must be unique. The "documentation" is optional. It is enclosed in double quotes and contains information that is not processed. The "declare" statement is optional. It is used to hange the priority of a rule's activation when it is competing with other rules prepared to fire at the same time.

Electronic Representation of Scheduling Concepts

The following examples show how this set of mappings can transform the English-like conceptual schedule provisions of Chapter 5 into ART-like .ules.

Example No. 1

(defrule check-activity-codes

"Activities should be described in a way that computers and anyone familiar with the construction work can understand them."

```
(project-to-criticize ?project)

(has-schedule ?project ?schedule)

(sub-task-of ?activity ?schedule)

(or  (not (responsible-party ?activity ?))

   (not (type ?activity ?))

   (not (class-of ?activity ?)))

=>

(assert (schema ?activity (code invalid))))
```

Comments. This rule follows very closely its counterpart represented in the English-like notation. Yet, the scheduling expert is unaware of ART's rule syntax.

Example No. 2

```
(defrule overali-start-time

   "The construction schedule should comply with the overall contract start date."

(project-to-criticize ?project)

(has-schedule ?project ?schedule)

(notice-to-proceed ?project ?start-date)

(earliest-start ?schedule ?es&:(< ?es ?start-date))

=>

(assert (schema ?schedule (notice-to-proceed violated))))

(defrule overall-completion-time

   "The construction schedule should comply with the overall contract completion date."

(project-to-criticize ?project)

(has-schedule ?project ?schedule)

(notice-to-terminate ?project ?finish-date)
(latest-finish ?schedule ?lf&:(> ?lf ?finish-date))

=>

(assert (schema ?schedule (notice-to-terminate violated))))
```

Comments. With the exception of the term restriction syntax imposed on ?es and ?lf, these two rules match their English-like equivalent. The ART's "&:" construct appends a predicate test to a term within a pattern.

Example No. 3

```
(defrule make-time-projections

    "Schedule projections should be based on comparisons between what has happened and what
    was planned."

    (declare (salience 100))

    (project-to-criticize ?project)

    (has-schedule ?project ?schedule)

    (sub-task-of ?activity ?schedule)

    (class-of ?activity ?class)

    (status ?activity in-progress | finished)

    (assessment ?activity slow-progress)

    =>

    (bind ?delay (LISP-function-1 ?activity))

    (assert (schema ?class (delay-factor ?delay))))
```

Comments. The purpose of LISP-function-1 is to calculate the delay factor for each binding of the variable ?activity. This value is accumulated in the multivalued delay-factor slot of ?class. The ART's " | " construct is equivalent to an "or" test.

```
(defrule class-delay

    "Determine the overall delay factor for this class"

    (declare (salience 50))
    (project-to-criticize ?project)

    (has-schedule ?project ?schedule)

      (sub-task-of ?activity ?schedule)

      (class-of ?activity ?class)
      (status ?activity unfinished)

      (duration ?activity ?dur)

      (delay-factor ?class ?delay)
```

```
=>

(bind ?overall-class-delay (LISP-function-2 ?delay))

(assert (schema ?activity (duration =(* ?dur ?overall-class-delay))))

(send ?schedule recalculate-cpm))
```

Comments. The purpose of LISP-function-2 is to calculate an overall delay factor for ?class. This function assumes that ?delay is bound to a list of values, each corresponding to the delay factor of ?activity. The overall delay factor for ?class is subsequently applied to all members of ?class whose status is unfinished. The impact of new durations is assessed by recalculating the CPM. The developer of an operational KBS has a few options from which to choose to implement such CPM recalculations. Partial scheduling, for example, is an option where the effects of the activities with new durations are propagated only to those activities belonging to their forward cone of influence. The backward cone of influence can subsequently be determined for each activity modified in the partial forward pass. If desired, new latest dates can be computed. This rule suggests the need for a control pattern that will prevent firing until all activations of the "make-time-projections" rule have fired.

Example No. 4

```
(defrule detect-front-end-loading-1

    "Cash flow front-end loading should be avoided"

    (project-to-criticize ?project)

    (has-schedule ?project ?schedule)

    (sub-task-of ?activity ?schedule)

    (class-of ?activity ?class)

    (responsible-party ?activity contractor)

    (type ?activity installation)

    ($-day-index ?activity ?index&:(> ?index *90-percentile*))

    (early-start ?activity ?es&:(< ?es *first-quarter*))

    =>

    (assert (schema ?activity (front-end-criteria yes))))

    (defrule detect-front-end-loading-2

    "Cash flow front-end loading should be avoided"

    (project-to-criticize ?project)

    (has-schedule ?project ?schedule)
```

(sub-task-of ?activity ?schedule)

(class-of ?activity ?class)

(responsible-party ?activity contractor)

(type ?activity installation)

($-day-index ?activity ?index&:(< ?index *10-percentile*))

(early-start ?activity ?es&:(> ?es *fourth-quarter*))

=>

(assert (schema ?activity (front-end-criteria yes))))

Comments. There are four global LISP variables in these two rules. Their possible bindings are (1) *90-percentile* should be bound to a value representing approximately one mean plus one standard deviation of the $/day index distribution, (2) *10-percentile* should be bound to a value representing approximately one mean minus one standard deviation of the $/day index distribution, (3) *first-quarter* represents the first quarter in the schedule execution period, and (4) *fourth-quarter* is bound to the period represented by the last quarter in the schedule execution time.

Example No. 5

(defrule impacted-by-weather

"An activity is weather sensitive if its materials and or labor are affected by either water, temperature, or moisture."

(project-to-criticize ?project)

(has-schedule ?project ?schedule)

(sub-task-of ?activity ?schedule)

(requires-resources ?activity ?resource-kind)

(affected-by ?resource-kind ?what&water | temperature | moisture)

=>

(assert (schema ?activity (sensitive-to ?what)

(weather-sensitivity yes))))

(defrule weather-causes-delay

"An activity that is weather sensitive should not be scheduled in a period when weather conditions are expected to be below specified minimums or above specified maximums."

(project-to-criticize ?project)

(has-schedule ?project ?schedule)

(sub-task-of ?activity ?schedule)

(class-of ?activity concrete)

(weather-sensitivity ?activity yes)

(weather-conditions ?activity present)

(weather-protection ?activity absent)

=>

(assert (schema ?activity (assessment likely-delay))))

Comments. The ART code of these two rules is a mirror of their English-like counterparts. Thus, the participation of scheduling experts in the writing of rules using the object-oriented English-like notation pays off.

Example No. 6

(defrule submittal-approval-relationship

"Award, submittal, approval, and procurement lead time should be an integral part of construction schedules."

(project-to-criticize ?project)

(has-schedule ?project ?schedule)

(sub-task-of ?activity ?schedule)

(class-of ?activity ?class)

(type ?activity submittal)

(sub-task-of ?other-activity ?schedule)

(type ?other-activity approval)

(class-of ?other-activity ?class)

(not (followed-by ?activity ?other-activity))

=>

(assert (schema ?activity (submittal-approval-logic bad))))

Comments. Again, the resemblance of this rule to the less intimidating English-like representation is obvious.

Discussion

It is clear from these six examples that the generation of ART-like code is greatly facilitated by the existence of English-like forms. This ease of transformation raises some questions: How close is the English-like notation to the ART syntax? Should the English-like notation be farther from ART and closer to English? Should there be an English-like grammar to begin with?

These questions can be answered in many different ways so that a compromise can be reached. However, the real test of where the English-like grammar should be located will be decided by the people in charge of maintaining operational KBSs.

7 KNOWLEDGE IMPLEMENTATION

Aim

Once the English-like knowledge acquisition grammar and the ART knowledge representation language are related to each other, they can be used to guide the coding of the "electronic" knowledge base. As depicted in Figure 10, this is a phase of programming, compilation, and code execution.

The aim of this chapter is to describe a microcomputer-based prototype, outline the design philosophy of an operational KBS, and provide an object-oriented infrastructure on which to build.

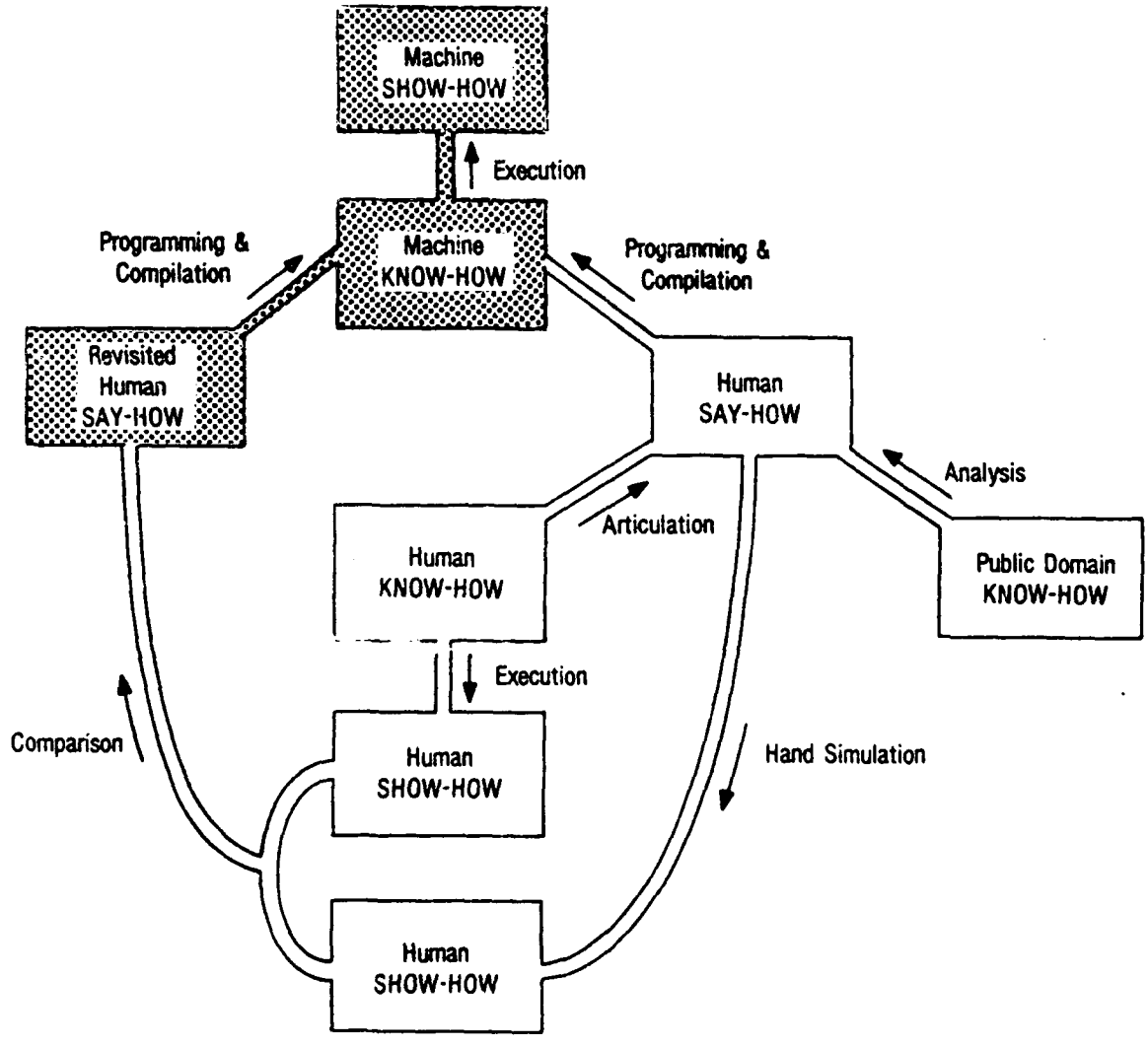

Figure 10. Execute route map.

Personal Consultant Plus KBS

This implementation represents the first attempt to codify the scheduling concepts developed in Chapter 4. The main thrust of it was to find out whether it is possible to encode construction scheduling expertise in a computer language. For this reason, the code is not necessarily the most elegant or efficient.

Since the USACERL specifications call for a microcomputer-based tool, the following hardware-software environment has been selected.

1. Software:

   a. Personal Consultant Plus--Inference Engine

   b. Primavera Project Planner--Project Management System

   c. dBASE III Plus--Relational Data Base Management System

2. Hardware. IBM PC/XT, AT, compatibles, and TI Professional.

The first system design problem encountered was how to make the PMS information available to the inference engine other than by retyping it. The solution includes the splicing of a relational data base management system (RDMS) between the PMS and the inference engine. This RDMS acts as a repository of schedule information, in addition to providing its own command language to perform computations on schedule information. By solving the problem in this manner, it is conceptually possible to access all kinds of external data bases, e.g., an estimating data base.

Figure 11 schematically represents the software configuration employed. *Separate communication* files are built to effect the interface between the various software packages. Neither Primavera nor dBASE III Plus nor PC Plus has access to the other's data structure. Communicating among these programs means writing messages in a file, providing this file's name to the other program, and letting the other program read the contents of such file. The nomenclature used defines these communication files as *.xxx. Individually, their purposes are:

*.dbf Primavera's output file: It consists of computed information, such as early start, percentage complete, dollars expended, etc. This file is created in a format that can immediately be accessed by dBASE III Plus.

*.prg dBASE III Plus command file: It contains dBASE III Plus commands to be executed sequentially. This file is sent to dBASE III Plus by PC Plus when there is need for schedule information that is not known by the inference engine.

*.txt dBASE III Plus results file: This is an ASCII file generated by dBASE III Plus responding to the inference engine information requests. This file is subsequently read by PC Plus to obtain the desired parameter values.

For example, the sequence of steps applied when PC Plus needs to retrieve a value from the schedule data base or when it requires performing a calculation using the same external data base is as follows:

1. PC Plus stops rule execution.

2. PC Plus frees Random Access Memory (RAM) by saving most of itself on the hard disk.

3. PC Plus opens an MS-DOS shell.

4. PC Plus loads dBASE III Plus into memory.

5. PC Plus sends a message to dBASE III Plus with the name of the command file that should be used to perform the desired action. This step implies the existence of a library of dBASE III Plus command files from which to choose.

6. dBASE III Plus executes the selected command file. The last thing that every command file does before closing is open an ASCII file and write the results of its execution.

7. dBASE III Plus sends a message to PC Plus communicating that its task has been completed.

8. PC Plus restores itself from the hard disk. This restoration brings the environment back to where it was before executing step 1.

9. PC Plus opens and reads the file created by dBASE III Plus in Step 6.

10. PC Plus continues processing.

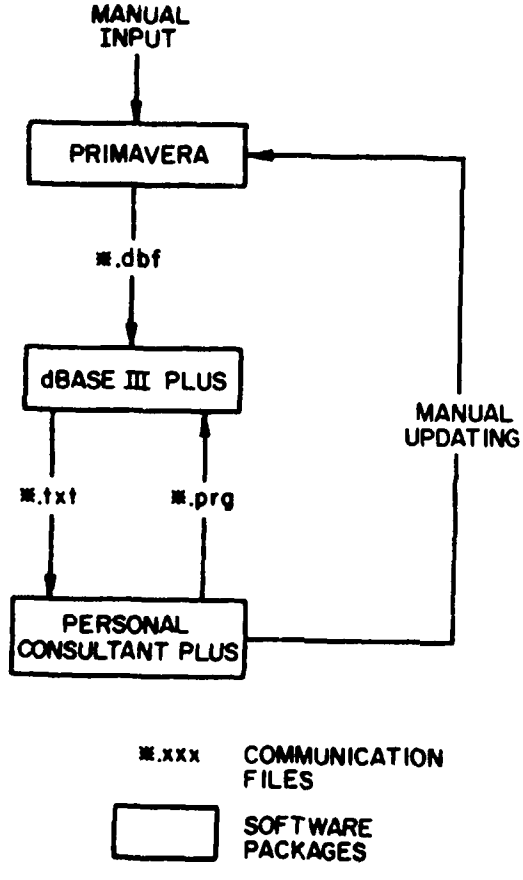

Figure 11. File-based communication mechanism.

Since the PC Plus versions utilized during this implementation lack data base-like relational capabilities and a pattern-matching language, all such work is done by dBASE III Plus. This system's design decision suggests that for every calculation to be performed on the schedule data base, there has to be a dBASE III Plus command file capable of executing the PC Plus request. In this regard, the function of the PC Plus rules is mainly to decide which command file to select and execute. By the time the prototype was finished, it became evident that it could have been implemented in dBASE III Plus alone without using PC Plus as the anchor.

The system has been developed to the proof-of-concept level, proving that it has enough functionality to demonstrate viability. Appendices E and F contain examples of PC Plus code and dBASE III Plus procedures, respectively. Appendix G includes some LISP I/O code, which enabled PC Plus to open and read ASCII files.

Advanced KBS Environment

The KBS trend is toward complex, sophisticated, and integrated systems like the one shown in Figure 12. The benefits of linking together data base, project management, and KBS technologies to help optimize information use and to assist the project manager in its interpretation far outweigh the complexity and cost of the system's implementation.

As shown in Figure 12, the development environment consists of two computers: (1) an IBM-PC/AT hosting Primavera, an interactive microcomputer-based PMS, and (2) a TI-Explorer LISP machine housing ART. Communication between the two computers is established across a serial line using the RS232 port. The communication across the serial link uses an error-free protocol to prevent data errors. The sending system divides the file into "blocks" of data and performs a calculation on each block before sending it. The receiving system performs the same calculation after receiving the block, and the two systems compare results. If the results agree, the data is assumed to be error-free (99 percent confidence level), and the remaining blocks are sent the same way. Appendix H contains examples of LISP functions that control the file transfer between the TI-Explorer and any other machine with XMODEM communication software, e.g., Procomm, Smartcom, Xtalk, Kermit.

Advanced KBS Conceptual Operation

The process is initiated by the project manager, who inputs initial or in-progress project data to be processed. The input goes to the PMS where schedule calculations are performed. This results in schedules which are to be criticized by the ART-based KBS. In one of this research's computing environments, Primavera's output schedules are represented in two customized ASCII reports. The customization consists of choosing both the activity data Primavera is going to export and its file format (e.g., ASCII, LOTUS, or dBASE III Plus).

The first report contains all activity data, with the exception of logic. The second report mainly consists of the activity's logic relationships. These two output reports contain the minimum project information from which inferences and alternative scenarios are explored. Appendix I contains examples of ART and LISP code used to parse the Primavera output files.

When ART first requests information about the project, it opens communication to the external Primavera program, waits for the schedule data to be sent across the serial link, stores it in schemata, and starts processing. Once the transfer of data is completed, a set of rules in the knowledge base is triggered to assign each activity to the corresponding frames in the semantic network, e.g., concrete, electrical, mechanical, activity, weather sensitive, etc.

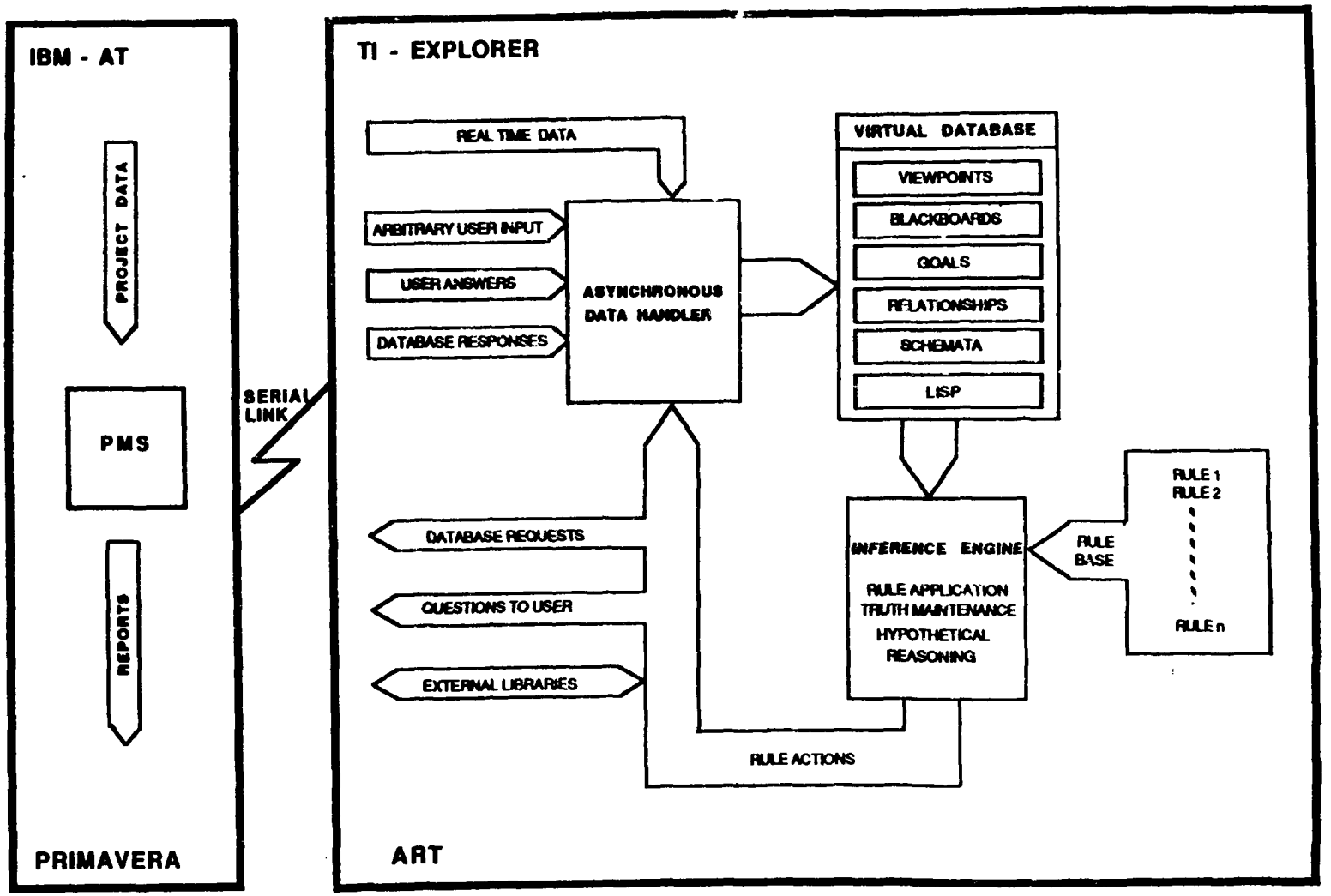

Figure 12. Advanced KBS environment.

At this point schedule evaluation can continue. The inference engine can modify activity attributes whose effect on the project can only be estimated by recalculating the schedule. In such cases, the knowledge base can create new schedule data structures and send them to a CPM-Kernel for processing. When the new results return, they are inserted into the appropriate schemata, and schedule processing continues. There are, thus, no limits on the number of schedule recalculations that may be needed before the original input schedule evaluation is terminated, with ART still maintaining control.

ART-Based KBS

The PC-based tools of Personal Consultant Plus KBS, impose severe constraints on the system's development. They are, among others, (1) insufficient computational power, (2) limited potential for system growth, (3) lack of a truly frame-based representation language, (4) lack of a pattern-matching language, and (5) lack of object-oriented programming capabilities. These limitations have led to the selection of a computing environment able to partially or wholly surmount some of these constraints. By the same token, some new ones arise; e.g., the development environment is too expensive to really be considered a viable delivery platform.

In this implementation, it has been decided to develop a conceptual semantic network infrastructure, rather than recode the scheduling concepts previously implemented on a personal computer. This permits and stimulates system growth, as well as serves as the foundation for rule creation.

The semantic network works as follows. During the construction planning phase, a work breakdown structure (WBS) is routinely defined, based on project phases, goals and organization. Milestone descriptions are derived from the WBS as tasks suitable for scheduling and monitoring. Traditionally milestone descriptions and codes are defined to convey both a building and a construction process; e.g., "*cast in place 2nd floor slab.*" The hierarchical relationship as well as the inheritance path of this milestone are shown in Figure 13.

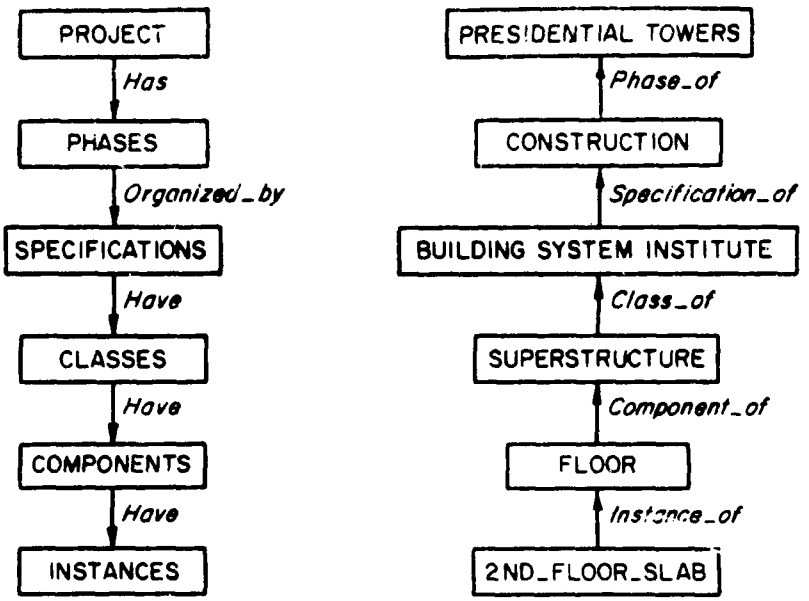

Figure 13. Knowledge base taxonomy.

A relation connects a schema to one or more other schemata. The inclusion of one or more relations in a schema serves to establish it as a node in a hierarchy. In other words, relations are the links that establish a schema hierarchy and permit inheritance of attributes. Note that the arrows shown in the diagram are significant because they originate with the object being defined. They point to each schema that is listed as the value of the relation.

Figures 14 and 15 depict a section of the semantic network. The primary objective of this representation is to provide a semantic interpretation of every milestone in the construction schedule. For example, when an activity like "cast in place 2nd floor slab" is found in the schedule, the KBS immediately deduces a series of facts and compilations about it. It is known, for example, that this activity (1) contains all basic schedule parameters, e.g., early start, percent complete, etc.; (2) represents a slab in the superstructure; (3) is made of cast-in-place concrete; (4) consists of formwork, reinforcing steel, and concrete placing, curing and stripping; (5) is sensitive to cold temperatures, snow, rain, labor productivity, etc.; and (6) has followers and successors as well.

The semantic network consists of three layers of classes. The first level contains 16 CSI classes and 16 BSI classes. The CSI branch is trade-based, whereas the BSI branch is area-based. Each abstract class at Level 1 is subdivided into more specific subclasses. Similarly, each subclass at Level 2 is further subdivided into subclasses. A specific schedule activity may be attached at any level in this three-tier hierarchy and thus inherit general or specific attributes. Appendix J contains examples of ART and GoldWorks code that define parts of this semantic network.

CPM-Kernel

The CPM-Kernel is a rule-based and frame-based framework that has been written using ART and Common LISP. Much of the CPM-Kernel design is adapted from [Jackson 86, Yoshimoto 85]. This initial version provides the basic features of a PMS. However, expansion to incorporate a number of research ideas, ranging from interactive graphical interfaces to automatic schedule generation, is feasible.

The justifications for having a CPM-Kernel coexist within the same KBS computing platform are as follows: (1) the KBS and the PMS reside in different computers, and thus the KBS and the PMS are not easily integrated; (2) the evaluation of alternate schedule scenarios resulting from inconsistencies found in the original input schedule may be made more efficient; (3) sending modified project data across a communications serial link every time the schedule needs to be recalculated should be avoided; (4) if the basic PMS is written using the same computing languages the KBS utilizes, then both systems share the same schedule's object-oriented representation, data structures, and procedures; and (5) testing and evaluating the implementation of conventional CPM techniques uses knowledge-based and inference control schemes.

The CPM-Kernel consists of a project-scheduling system and a resource conflict resolution system. The project-scheduling system falls into a chain of phases. The preprocessing phase involves processing the activity templates and creating data structures to control and represent the schedule solution.

The forward-pass and backward-pass phases apply CPM representations and techniques to compute the earliest and latest times for each activity. ART's "root" viewpoint contains all of the facts and schemata which define the schedule, its activities, and required resources. Each activity has a time window associated with it. This time window is defined as the difference between its latest finish and its earliest start. Activities can then be scheduled anywhere within the time window without delaying the project's completion date. Each activity is assigned a default usage window which is bound by its earliest start and earliest finish.

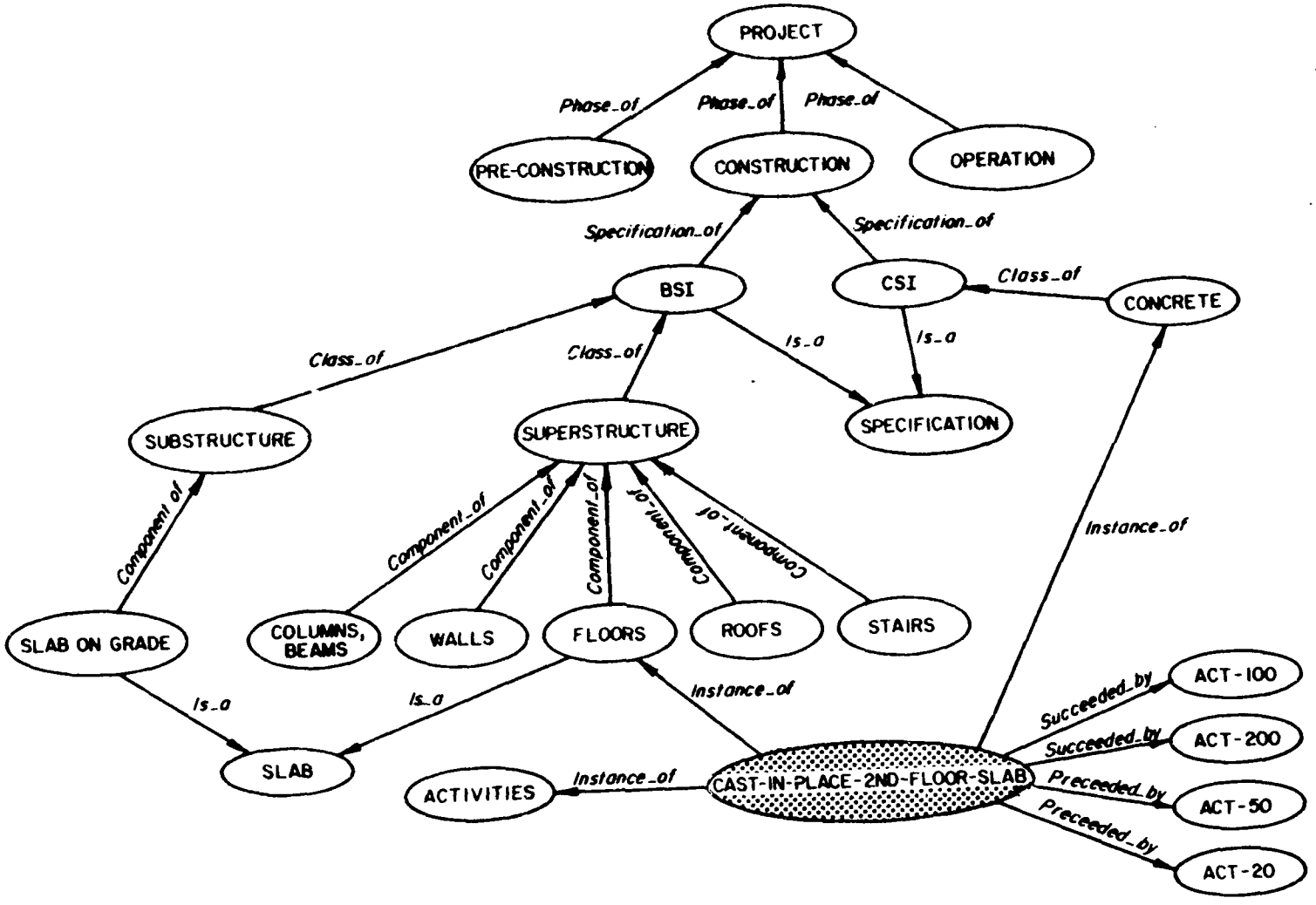

Figure 14. Sematic network--Part A.

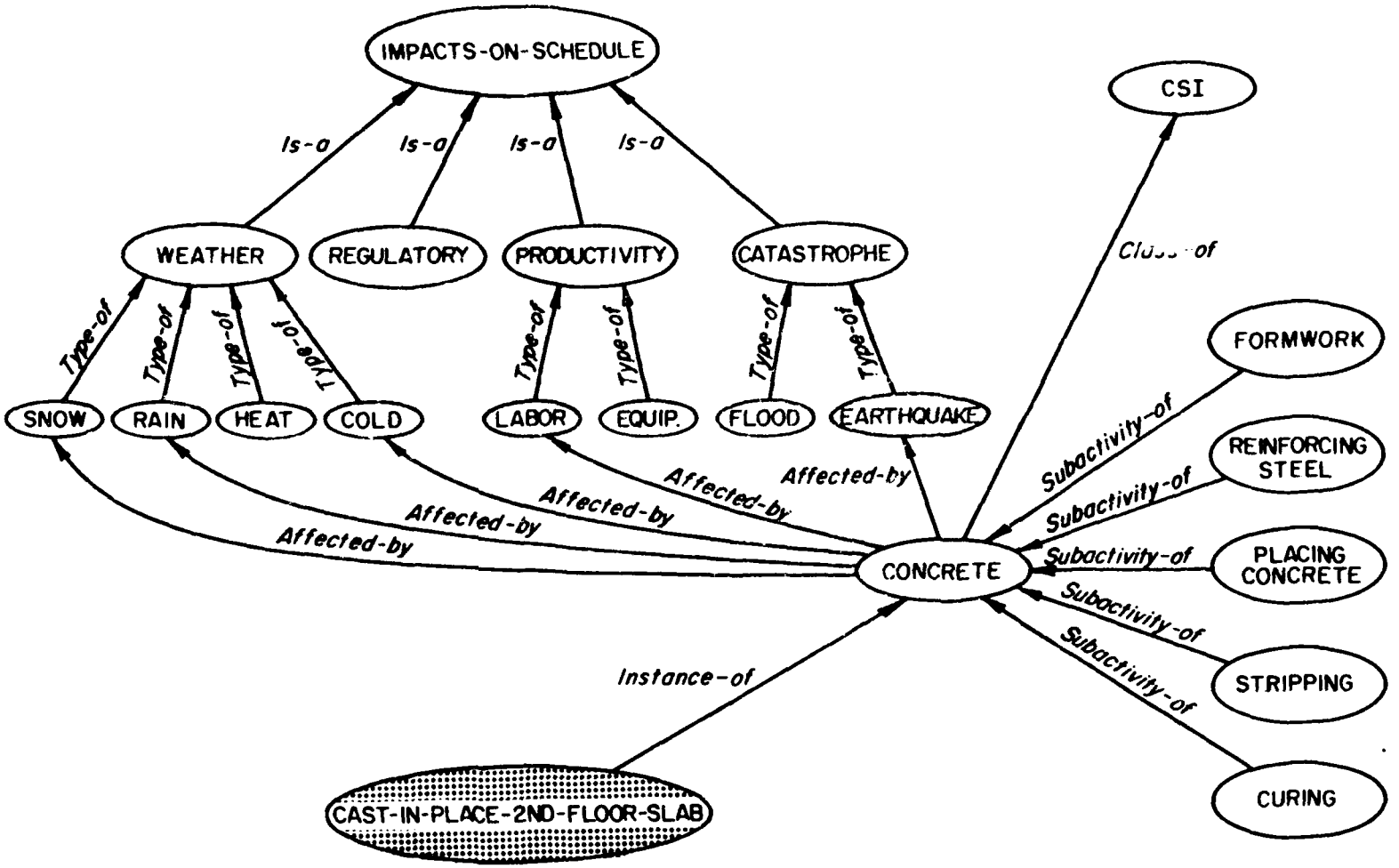

Figure 15. Sematic network--Part B.

The activity input can have an arbitrary number of levels of subtasks. However, both scheduling and resource allocation occur only at the most detailed level, that is, in activities without subtasks.

In the following phase, the CPM results are preliminarily screened to check for time window consistency in the case of synchronized activities and for tasks with a shared resource. The fifth phase performs an abstraction function that updates information for aggregate activities. This provides a rudimentary hammock capability for scheduling parameters.

The resource conflict resolution system follows to allocate resources to tasks. It subsequently attempts to solve resource conflicts. The method applies in circumstances where resource limitations can be corrected and time constraints are binding. Resource allocation is implemented using ART's viewpoint mechanism [Jackson 86].

A network of alternative solution paths is created and scanned by rules that define constraints and recognize a solution. Toward this end, rules determine whether any resource shortages exist in the "root" viewpoint. For each resource shortage alternative, viewpoints are sprouted. Each viewpoint represents an attempt to eliminate resource contention by shifting tasks within their float time to the right (the decision is whether or not to shift the implicated task). Each newly created viewpoint is examined for further shortages, which may cause additional viewpoint sprouting and merging beneath them. Constraint rules terminate the search along nonviable solution paths (e.g., whenever a task's usage window is shifted outside its CPM time window) and prune the search space. Figure 16 shows the different contexts generated to solve resource contention between two activities.

Solutions can now involve exhaustive search of a combinatorial space. Additional rules implementing the late-start sort heuristic, for example, may result in more effective searches. Such heuristics are not part of the current resource allocation system; however, they represent natural extensions. Appendix K contains examples of the data structures and procedures for scheduling and resource allocation.

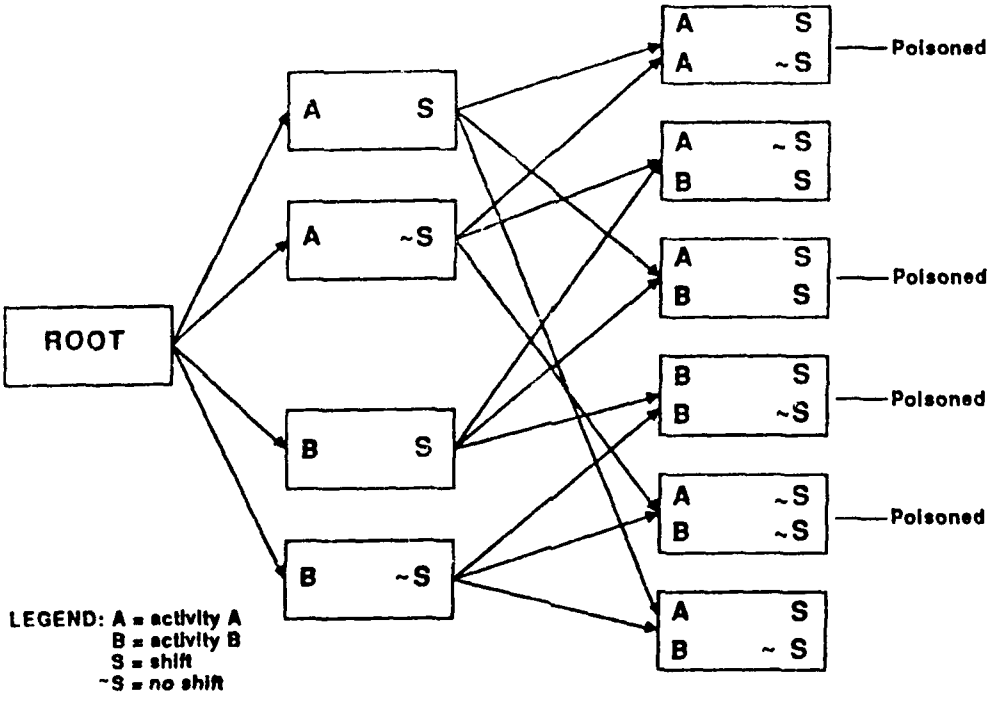

Figure 16. Partial set of viewpoints to solve resource contention.

8 CONCLUSIONS

Summary

This study contributes an additional step towards the development of a critic KBS in the construction scheduling domain. The work has demonstrated that the pursued knowledge engineering approach is broadly sound, and that there is a set of conceptual provisions which can be applied to assess the degree of reasonableness of construction schedules. This research represents a modest contribution for the improvement of schedule management automation.

The development of the "paper" knowledge base for the most part has involved extracting, articulating, and synthesizing construction scheduling knowledge from the literature and from scheduling experts. The research also has required finding the appropriate computer tools and the proper representations for the knowledge. Fully programming an operational KBS has not been part of this study because it was considered necessary to first have a better understanding of the knowledge base's breadth and depth.

This research has led to fundamental advances in construction scheduling thought in several distinct ways. First, this investigation adapts a knowledge engineering methodology capable of transforming ill-defined construction scheduling knowledge into the specifics of an operational KBS. Within this knowledge metamorphosis process, one of the three knowledge elicitation techniques features a novel implementation of the "limited information" approach to knowledge acquisition, namely, the simulation of a blind scheduling expert, i.e., a sightless scheduling expert who explicitly has to ask for only the relevant information which he/she uses when criticizing construction schedules.

The second important advance of this research is a by-product of the knowledge engineering methodology. This study synthesizes construction scheduling knowledge that successful construction engineers use to criticize schedules. To the extent that the scheduling principles represented by the "paper" knowledge base are not committed to a single interpretation, they may be considered conceptual in nature. Thus, they should be applicable to a vast majority of large, medium, and small owners and contractors. In this way, the synthesized construction scheduling principles can be used as the guidelines for performing construction schedule criticism.

This work can be thought of as a natural extension of important research conducted by a previous generation of researchers like Fondahl, Crandall, and Halpin. They helped turn scheduling theory and its practices into the refined science that it is today. Now that the analytical models and mathematical theory are so mature, it seemed quite proper, indeed critical, to attend to the practice-oriented features of construction scheduling.

The third advance produced by this research is the development and adaptation of an object-oriented knowledge acquisition grammar. Such grammar allows the transformation of scheduling knowledge written in plain English into an English-like notation. The primary benefit is that the grammar allows the scheduling experts to remain in the knowledge engineering loop. Furthermore, the knowledge in this form can be ported to a variety of inference engines.

The fourth contribution of this research relates to the derivation of a set of mappings to transform object-oriented English-like knowledge into ART-like code.

Finally, the fifth advance of this research is in applications. The prototype systems developed in Personal Consultant Plus, ART, and GoldWorks endorse the proposition that KBS technology offers programming techniques which facilitate the representation and manipulation of scheduling knowledge. The infrastructure of knowledge representation schemes provided by these prototype systems will be of

value during the future development of an operational KBS. Such a critic KBS can be designed at a sufficiently generic level because a large body of construction scheduling knowledge (the "paper" knowledge base) has been categorized and structured. This system should allow project managers to become proactive rather than reactive.

Issues in Construction Scheduling Modeling

Generation and criticism of construction schedules is a process based partly on art and partly on engineering principles. Arguments can be offered as to which is dominant, though it is not important to determine their respective contributions precisely. Instead, what is relevant is that their presence carries some implications concerning the kinds of knowledge (empirical versus causal, generic versus idiosyncratic) that constitute the scheduling process.

Most of the scheduling knowledge elicited in this study is empirical and idiosyncratic. It is empirical in that it involves expertise considered practical, although not necessarily having an underlying knowledge of causality. It is idiosyncratic to the extent it reproduces the knowledge and problem-solving style of the experts consulted. After all, if everything is held constant, a different but equally good schedule may be produced by another contractor. This point strongly suggests that for an operational KBS to be used by a firm, the KBS should reproduce quite accurately that company's natural way of doing things. This in turn indicates a need for more user participation in building construction-scheduling KBSs.

These observations reflect the highly fragmented nature of the construction industry.

Suggested Directions for Future Research

The development of a knowledge base is an exploratory and evolutionary process. It is a foregone conclusion that the completeness of a knowledge base will determine the accuracy of an operational KBS. With this in mind, the research described herein should be only a starting point for the development of a KBS in the construction scheduling domain.

The synthesized "paper" knowledge base needs to be implemented in an operational KBS, and verified, expanded, and integrated into a comprehensive PMS. The following are major related areas suggested for further investigation.

Knowledge Base Representation

An underlying premise supporting this research is that construction schedule criticism is not formally recognized as a project management task. This fact is mainly a result of the time-consuming aspect of most schedule analyses. It follows that the development of an evaluative KBS for the largely heuristic part of the scheduling analysis process is needed. Because the preliminary conceptual knowledge has been synthesized in this report, the primary purpose of this study should be to explore the issues that arise in the development of an operational KBS, whose role is that of a critic.

The resulting system should function as a project manager's assistant in the criticism and evaluation of construction schedules. Hybrid tools at the high end of the AI spectrum should provide the necessary representation and computational power, combined with enough flexibility to allow such implementation. Possible object-oriented organization schemes are described in this report.

To assure some degree of success, i.e., actual use of the system, the computer implementation must allow for knowledge modification, addition, and deletion. Form-filling tools should help in this regard.

These actions will be logically expected and should be performed by an individual responsible for the system's maintenance within each company. This type of user interaction will require the development of a high-level interface to the knowledge base. The interface should permit people access to modify the knowledge base without LISP programming experience. Justification for having such a facility built in is exemplified by the need of each company to customize the system's default behavior.

The system should be designed with the aim of becoming another integral module of computer-aided engineering and of operating in construction practice real-time.

Knowledge Base and KBS Validation

This area of need encompasses two facets. The first concerns knowledge base validation and consistency checking; the second evaluates the degree to which the user has accepted the emerging KBS technology.

In light of the potential shown by this study's microcomputer-based prototype, the development of an operational KBS should demonstrate that this new approach is satisfactory for accelerating many brute-force analyses and calculations typical of routine scheduling. However, this methodology cannot be shown to be a sufficient solution through the development of an operational KBS alone. Subsequent experimentation and analyses are necessary to increase this level of sufficiency. This experimentation, or knowledge base validation, should establish the reliability, scope of applicability, weaknesses, strengths, and boundaries of the knowledge base. These attributes may be determined by evaluating the operational KBS against test cases. Construction schedules to be selected as test cases should fall within, on the boundaries, and outside the scope of applicability of the system. Validation should also ensure that the system degrades gracefully, as might be expected of a human. Table 2 represents a data-gathering form that may be used to carry out some of the validation proposed herein.

The other aspect which this study should address is the readiness of the construction industry to adopt the KBS technology for everyday operations. An assessment of the construction industry's infrastructure for embracing this technology is of paramount importance. This infrastructure may take the form of (1) recent graduates with AI background joining the white collar work force, (2) continuing AI-based education programs oriented toward former professionals, (3) training courses focusing on computer literacy, (4) identification of technology transfer programs, (5) classification of university-based programs that allow industry professionals to re-enter the classroom at the graduate level, (6) cataloging basic research products ready to go into the developmental phase, and so forth. If this infrastructure is nonexistent, then the adoption of KBS technology by practitioners will be as difficult as the adoption of CPM technology was, or perhaps still is.

Ties to a KBS for Schedule Generation

A feasible construction schedule can be thought of as a sequence of preformulated constraints which need to be satisfied to successfully build a project. These constraints include, for example, precedence relationships, resource synchronization, managerial preferences, and so forth. The scheduling provisions represented by the "paper" knowledge base illustrate another set of constraints, which are not widely disseminated or accessible among construction planners. These constraints are not part of the typical planner's thought process, but are often part of the project manager's problem-solving knowledge base. When these constraints are applied, they are expected to criticize the proposed candidate schedule and state whether the constraints are satisfied. It follows that the construction planner presented with such negative critiques should be able to modify his or her thought process to avoid future criticism. In practice, however, schedule generation and schedule execution are performed by different people. As a result, construction planners do not receive direct feedback concerning the reasonableness of their generated schedules.

Table 2

Form to Validate the "Paper" Knowledge Base

	QUESTION:	PROVISION 1	PROVISION 2
A)	Is it reasonable? (yes/no)	_____	_____
B)	Is it a company standard provision? (yes/no)	_____	_____
C)	Is it enforced? (yes/no)	_____	_____
D)	If it is not enforced, should it have been? (yes/no)	_____	_____
E)	How is its enforcement? (strict/informal)	_____	_____
F)	If it is used, to whom does it provide an advantage? (Owner/Contractor/CM/AE)	_____	_____
G)	If it is used, to whom does it provide an additional risk? (Owner/Contractor/CM/AE)	_____	_____
H)	If it is not used, to whom does it provide an advantage? (Owner/Contractor/CM/AE)	_____	_____
I)	If it is not used, to whom does it provide an additional risk? (Owner/Contractor/CM/AE)	_____	_____
J)	Should it be improved? (yes/no)	_____	_____
K)	Should it be part of contract documents? (yes/no)	_____	_____
L)	Do you agree with its intent? (yes/no)	_____	_____

Research and development in generative KBS for computer-assisted construction planning is being actively pursued at research centers like the University of California at Berkeley, the University of Illinois at Urbana-Champaign, Carnegie-Mellon University, MIT, and Stanford University. A natural extension of this work would be to pass the candidate computer-generated schedules to a critic KBS and relay the resulting criticism back to the schedule generators. It would then be the task of the generator to suggest modifications to the proposed schedule to satisfy the "paper" knowledge base provisions. The passive criticisms of the schedule provisions developed in this research can be components of generative KBSs for schedule formation.

Use of Qualitative Reasoning Methodology

In the domain of digital circuit electronics, [Davis 84] described a technique called constraint suspension that provides a powerful tool for troubleshooting. As the knowledge base of this research is expanded and an operational KBS is developed, the constraint suspension technique can be used to determine the factors responsible for an observed set of schedule misbehavior symptoms which can later be used to indicate defects at the planning level.

Given the symptoms of a particular instance of schedule misbehavior (e.g., the final completion time is delayed), the constraint suspension technique generates candidate activities which could plausibly be responsible for the schedule's failure. CPM technology models the intended behavior of the schedule as a network of interconnected activities wherein each activity has its own model of behavior, e.g., crew's productivity rate, resources used, materials consumed, calendar, etc. If the schedule is malfunctioning, then the output predicted by simulating the entire schedule will not match actual construction progress. The purpose of constraint suspension is to look for some constraint (i.e., an activity's own model of behavior) whose modification will make the schedule predict actual progress. Thus, each constraint found corresponds to an activity whose misbehavior can account for some observed symptoms of schedule misbehavior.

Analogic Reasoning with Historic Data Bases

In the construction industry, historic data bases are used when construction operations are governed by a closed-loop process. In these processes, a feedback loop updates a static bank of knowledge. The updating task depends on the expected use of the information contained within the data base. For example, updating may take the form of accumulating the experience generated by executing a new project or of modifying existing historic records to reflect new trends.

Construction applications that benefit from using historic data bases are, among others, (1) construction schedule generation, (2) cost estimating, (3) cost control, and (4) cost forecasting. In the first and second categories, global project parameters can be used to retrieve equal or similar projects. Those projects' actual schedules and costs can be applied to guide the development of the new project's schedule and cost estimate. For the third and fourth situations, a snapshot of the current project can be matched against the historic data base to plausibly explain current behavior or to modify current trends based on previous experience.

The human cognitive process which typifies the task of matching a current problem against a historic data base is generally goal driven. Construction managers generate a set of plausible scenarios with the data that delineate the current problem. They subsequently match each scenario against cases contained within the historic data base. Finally, construction managers seek supporting evidence on each hypothesis by matching the conditions of each relevant case within the historic data base with those of the current problem.

This section proposes a data-driven computer approach to the equivalent goal-driven human cognitive process. This proposition is based on the conceptual understanding of the mechanics of the RETE algorithm [Forgy 82], which is central to opportunistic rule execution. ART and GoldWorks, for example, are tools based on RETE principles.

ART [Clayton 87] takes the conditions of rules and compiles them into a "pattern net" and a "join net." The objective of the pattern net is to sort facts into the patterns that they instantiate. The pattern net is a simple tree in which the leaves correspond to the patterns of the rules. The overall structure of the pattern net is independent of the order in which the patterns are used in rules.

The purpose of the join net is to combine instantiated patterns to build rule activations. The join net topology is, indeed, sensitive to the pattern order within rules.

The pattern and join nets can be viewed as fact processors in which facts are placed in at the top and rule activations are produced at the bottom. Figure 17 graphically depicts such a fact processing mechanism. In this figure, the data base component portrays the historic data base, the pattern and join nets represent the current problem, and the agenda corresponds to situations in which the current problem matches previously documented instances. This match or rule activation occurs only when a historic data base fact instantiates each pattern in a rule. An even more interesting scenario is generated when the description of the current problem does not cause rule activations; that is, exact matches are not found.

A partial match occurs when a group of historic data base facts compatibly instantiate only some of the patterns of a rule. In such cases, rule activations will not happen. The keyword in the partial match definition is "compatibly" because the compatibility test is performed in the join net. There are three practical implications of this phenomena. Partial matches can occur when (1) all the patterns of a rule have been instantiated, yet the compatibility tests on the join net fail; (2) a rule has some patterns instantiated, which pass the join net compatibility tests, and has some others uninstantiated; and (3) a rule has both conditions present.

In practice, circumstance (1) arises when the current problem combines the conditions of several individual projects, which are separately contained within the historic data base. In such situations, the pattern net can be recursively decomposed to treat the current problem as several independent subproblems. After successful matches are found for each subproblem, an aggregated match can be composed to represent the match of the original monolithic problem. Situation (2) can be solved by gradually relaxing the term restrictions present in the uninstantiated rule's patterns. Fuzzy set or approximate reasoning theory may prove valuable in these cases as the concept of similarity is steadily redefined.

The application of this data-driven approach suggests obvious object-oriented representation requirements for the historic data base. A less evident requisite is imposed on the semantic representation of the current problem. Because the intent is to knit pattern and join nets, the current problem should be represented in rule syntax, as opposed to the more apparent object-oriented representation.

Self Research Assessment

A retrospective analysis of the methodology and results obtained in this study illustrates several weaknesses. They are presented here in hopes that prospective researchers wishing to build on this work may address them carefully.

Breadth and Depth

The number of conceptual schedule provisions synthesized is 34. No attempt is made in this study to precisely determine the boundaries of construction schedule criticism. Whether there are 35 provisions (in which case the "paper" knowledge base is missing only one) or 1000 provisions (in which case the "paper" knowledge base is lacking over 900) is not accurately known. What can be stated in this regard is that the 34 provisions represent a large body of construction scheduling knowledge, making it worth developing a KBS.

Rookie's Role

The fact that the rookie was not a true apprentice may indicate a flaw in the methodology. However, knowing what questions to ask was important in this novel implementation of the "limited information" approach. Further experimentation with variations of this implementation may indicate that a true rookie can be used with success.

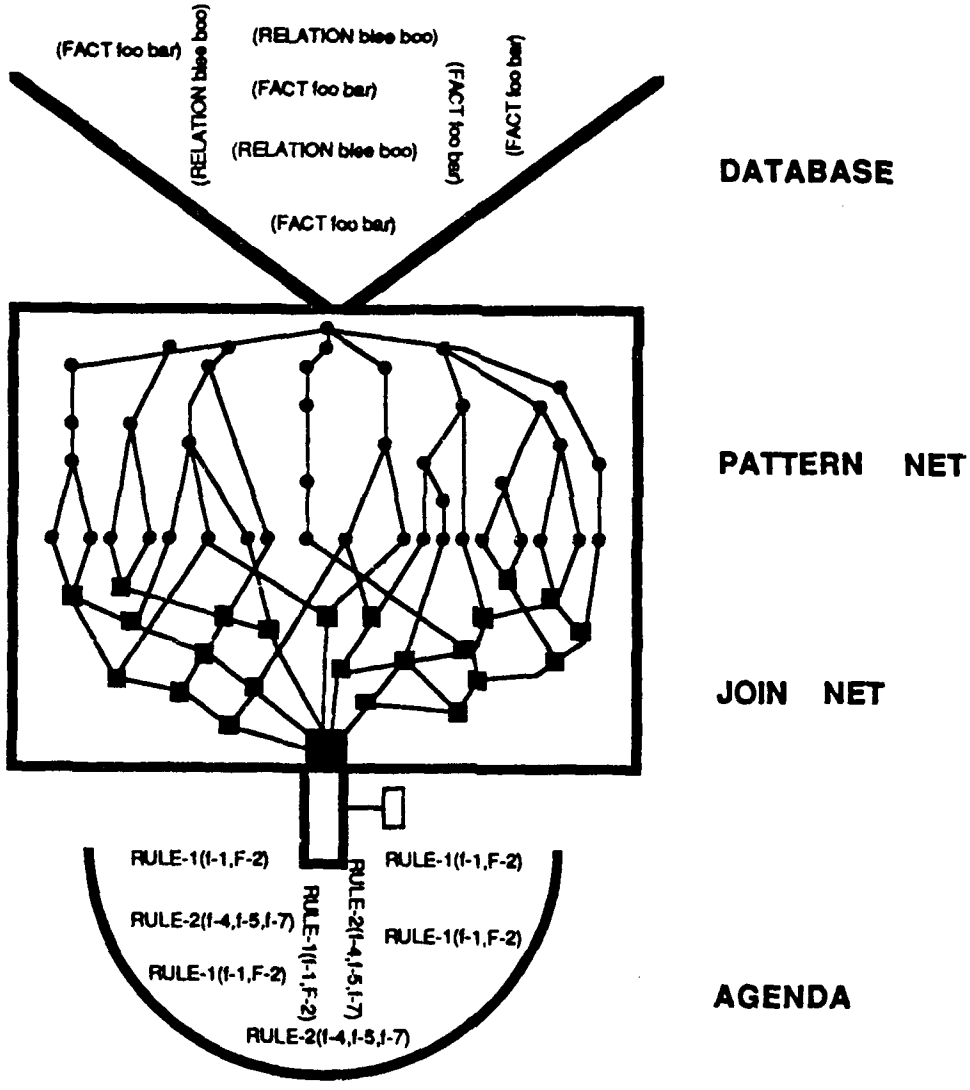

Figure 17. Pattern and join nets.

APPENDIX A:

TEXTBOOK SCHEDULING KNOWLEDGE

This appendix corresponds to the analysis of public domain scheduling knowledge. It represents the first-generation human say-how. The main source utilized in its generation was [O'Connor 82].

Initial Analysis

General Requirements

1. I-J Numbers. The user should skip enough numbers to allow revision without disrupting the scheme of ascending numbers.

2. Description and codes. Each activity must be coded and include a description of the work involved. Codes and descriptions should match the work breakdown structure specified in the contract.

3. Activity descriptions. Each activity must be written so anyone familiar with the construction work can understand it. Each description should be unique, and nonstandard abbreviations should be avoided.

4. Coding. The codes must be complete, correct, and reflect the nature of the work. They should be based on the Building System Institute (BSI) or on the Construction Specifications Institute (CSI) schemes.

5. Government activities. Government activities which affect construction progress should be included in the network. Reasonable durations should be assigned by the contractor and approved by the government. A dollar value must not be placed on these activities.

6. Participation of major subcontractors. Any subcontractor performing 10% or more of the total contract work is considered a major subcontractor, and should participate in the contractor's development plan. All major subcontractors may not have been identified at the time the contractor prepares and submits his schedule, but those who have should be involved in its preparation.

Time

7. Overall completion date. This date must comply with the contract requirements. The contract specifications define the time period within which the work is to be completed from Notice to Proceed. A schedule consuming more than the specified number of days is not acceptable. However, a schedule showing early completion is acceptable, provided unreasonable time constraints are not placed on governmental activities.

8. Durations for contractor activities. These should be reasonable and conform to the limits specified in the contract. Activity duration limits usually range from 1 to 30 calendar days. Generally, very few 1-day durations are needed for activities of significant nature. Long-term activities (more than 30 days) of continuous duration should be broken into parts not exceeding 30 days each. However, activities with durations of 30 days and above may still be acceptable if they do not lend themselves to further breakdown, or if the dollar value is small enough to allow accurate progress reporting and payment. Durations should also be based on the time of the year the operations are to be executed.

9. Durations for procurement activities. These should reflect market conditions. Procurement lead time must be an integral part of the construction schedule.

10. Float range. Float should be broad enough to support the premise that it has not been manipulated. Schedules having an inordinate number of critical activities are not acceptable. Zero floating a network defeats its fundamental purpose. One must know which activities are critical and which are not to effectively manage the work. Float is a significant element of the honestly developed network. Float may be used as necessary by the contractor without a change in the contract price. When the Corps wishes to use float to absorb change order work, it must negotiate an equitable price for the amount used. This solution lies in a policy that says, "Float has value to the contractor and must be treated as any other resource when pricing a change order." However, the price of the float is not constant: (1) overall, it is of more value early in the life of the job than when the job approaches completion because as the job progresses, the remaining risk factors diminish; and (2) for a given activity, it increases in value inversely to its quantity because uncontrollable events causing small periods of delay are more likely to happen than those producing longer delays.

11. The critical path. This usually consists of relatively few activities. If many parallel paths and/or a large number of critical activities exist, it is likely that some durations have been overstated for the purpose of eliminating float. In addition, managing simultaneous critical paths is harder than managing a single one.

Logic

12. The scope of work. It must be reflected in the network. The network must contain all work activities (with clear descriptions) to be performed under the contract.

13. Submittal activities. These activities include the contractor's preparation and submittal of shop drawings, catalog cuts, samples, etc., and the Corps' review and approval actions. All materials and/or methods requiring prior approval must have their submittal activities represented in the network.

14. Procurement activities. These generally occur after the proposed material is approved, but before the construction activity using the material occurs. As with submittal activities, all materials requiring approval should have their procurement lead-time activities represented in the network.

15. Sequencing and interdependencies. These must be logical. While it is the contractor's responsibility to plan and accomplish the work, many conventions can limit his options. For example, foundations will have to be completed before the roof is erected, etc. The Corps reviews the contractor's proposed plan to (1) confirm that it represents a reasonable plan for accomplishing the work, (2) ensure the work is broken down and identified well enough to permit adequate monitoring and reporting progress, and (3) ensure that monetary values are correct for payment purposes.

16. Constraints. External constraints should be considered, including site access, work of other contractors, local climate and environmental conditions, working schedules of local suppliers, contract-specific dates, etc. For example, built-up roofing should not be scheduled if expected ambient temperatures are expected to be below specified minimums.

Cost

17. Total cost. The total monetary values assigned to individual activities must equal the contract amount.

18. Activity's cost. The monetary value of each activity should conform to the range stipulated in the contract. The usual range is between 0.1 and 2.5 percent of the total contract amount. In addition, the monetary value assigned to each activity should represent a reasonable amount for that work. This analysis may be based on the cost of similar work completed recently.

19. Administrative activities. The monetary value of administrative activities should be zero. The cost of preparing submittals is considered part of the overhead the contractor must distribute to other activities. Government activities such as submittal reviews represent no cost to the contractor. A monetary value should not be placed on procurement and delivery activities since the cost of materials is usually included in the price of the construction activity using them. Payment for on-site materials (when allowed) should be handled outside the network framework.
20. Front-end loading. Front-end loading, the practice of placing an excessively high monetary value on activities scheduled for completion early in the project, is unlawful.

In-Progress Analysis

21. What is the schedule status of the work?

22. What is the actual status of the work?

23. What is the earned status of the work?

24. The contractor should revise the schedule when it no longer accurately represents his plan for completing the remaining work, or because actual progress has not kept up with the schedule. Ordinarily, only the durations and/or the logic of the remaining activities will be changed, with shortened durations implying the assignment of increased resources by the contractor. No increase in contract price or the value of the various activities is permitted, since the need for schedule revision stems from contractor-responsible causes.

25. Use current trends (float consumption, duration usage, etc.) to forecast the degree of success or failure to be expected in meeting future contractual milestones.

26. Identify areas that need remedial action while there is still time for such action to produce a positive effect.

27. Make projections based on what was planned and what actually happens.

28. Predict any significant deviations from the official schedule.

29. Make sure that the progress payment request is reasonable.

30. When a progress schedule is approved, it implies the acceptance of a practical way to finish the work on time.

31. It is essential to successful contract administration that the approved planning schedule be current.

32. When a change order occurs, the time factor for the modification can be unilaterally defined by identifying the network activities whose dollar value or duration are affected.

33. Any revised logic or durations may be developed by the contractor and/or owner and approved by both parties.

34. To estimate the effect of a change order, the current status of the work and the contractor's approved plan for completing the remaining activities must be known.

35. The change order's time effects can be observed by comparing the current approved plan to the plan generated after the changes to the network have been processed. At this point, it is known whether the change order would delay the completion of the remaining work, and if so, the length of the delay. This analysis will show those activities both directly and indirectly affected by the change order; the estimate of the dollar amount must take both into consideration. The estimate must also consider that time has value, whether or not it is on the critical path.

APPENDIX B:

INTERVIEWS WITH SCHEDULING EXPERTS

This appendix contains the scheduling knowledge generated by W. E. O'Neil Construction Co. It represents the second-generation human say-how.

Initial Analysis

General Requirements

1. Although precedence diagrams are desirable, the network representation notation is irrelevant for schedule analysis.

2. O'Neil allows a block of numbers to bracket activities or operations. For example, procurement activities may have numbers in the 1 to 400 range. In high-rise buildings O'Neil uses a repetitive numbering scheme, i.e., 1207 means activity 07 on the 12th floor.

3. In reinforced concrete high-rise buildings, O'Neil typically generates activity descriptions that include formwork, reinforcing steel, and placing concrete, i.e., "cast in place 2nd floor slab." This practice is followed unless there is an indication that work other than reinforcing steel can go on simultaneously, e.g., formwork in another nearby area.

4. The construction schedules reflect the starting and completion dates for all other prime contractors assigned.

5. O'Neil prefers to code the activities by the building process. In this way, interdisciplinary schedules are obtained. However, O'Neil uses both building (BSI) and construction (CSI) processes to code the activities.

6. Construction operations are defined only if they can be monitored.

Time

7. The duration of "approval" activities is estimated as follows: (1) the owner specifies it, (2) O'Neil uses 3 weeks for the first pass, (3) a backward pass is run to indicate the latest date before they impact on the schedule, and (4) a specifications look-up table is used. Options 3 and 4 are less desirable because they may or may not have float left. Float is a contractor's resource.

8. O'Neil shows float on construction schedules.

9. Float belongs to the general contractor. It should be considered like any other resource.

10. Float is an important resource to the contractor because in the majority of projects many considerations not anticipated from the drawings and specifications arise. No matter how definitive the schedule, it is extremely difficult to encompass all the things that will happen in the future.

11. If the owner uses up float to accommodate changes to the project, the contractor is entitled to a compensation claim not only for the direct effects of such a change but also for the indirect

impact such as manpower utilization, an inordinate amount of workers in the same period of time, and so forth. This is equivalent to selling the float to the owner.

12. An activity is critical if it has fewer than 5 days of float.

13. Construction schedules always have a critical path and a series of three or four "low float" paths to accomplish intermediate milestones, i.e., closures. This is typical of high-rise buildings with different occupancy areas.

14. O'Neil does not like to see activities with a three-digit float. Although the activity's logic may be correct, this implies that the activity has not been integrated to optimize manpower and other resources. It is rare to have an activity so discrete from the main flow of the schedule.

15. O'Neil does not show float on milestone schedules, which are used in sales presentations, because a full analysis of the project has not been made.

16. In preliminary milestone schedules it is common to avoid showing float because the idea is to block off time to get the overall completion date. These preliminary milestone schedules consist of 30 to 40 activities to represent $30 million to $40 million.

17. Important milestones to consider in a high-level schedule are: foundations, end of superstructure, enclosures, roofing, windows, exterior skin, elevators, hoist removal, release of structural steel design, etc.

18. Reasonable activity durations range from 5 to 25 days.

19. The cost associated with an activity plays no role in constraining its duration. This is especially true in the case of procurement activities where they have short durations and high costs.

20. The initial activity duration is based on known standard productivity for a type of crew. This duration is independent of the season of the year the activity is to be scheduled. O'Neil does not include in the activities duration a weather contingency that may or may not occur.

21. Weather-related time extensions are extremely difficult to get, unless very abnormal weather conditions are present.

22. Typical projects at O'Neil range from 9 to 15 months.

Logic

23. The "procurement chain" of activities O'Neil uses for major milestones consists of award, submit, approve, procure, deliver, and install. In high-rise buildings, O'Neil normally ties this chain of activities to the first area in which an item is scheduled. This assumes that once delivery starts, it is going to be consistent with erection. For example, in the case of structural steel, O'Neil may break the erection into various stages, but O'Neil really is not going to break the delivery of steel down by stages unless it is known that delivery is going to be discrete.

24. O'Neil includes the "procurement chain" in all networks.

25. Reinforcing steel and design mixes are examples of activities preceded by the "procurement chain."

26. If O'Neil can identify pieces of equipment that are unique, O'Neil will include a "procurement chain" on them.

27. Most submittals are generated by subcontractors and suppliers. O'Neil acts only as a clearing house.

28. Roofing and exterior closure are considered milestones because they carry over into when the wet trades (plaster, drywall, taping) can start. Dry trades (piping, duckwork) can do without the exterior closure.

29. In a multistory project the main logic diagnostic checks are done by area, e.g., floors, and by trade, e.g., drywall crew. Trade continuity should be observed throughout the network.

30. In a multistory reinforced concrete building O'Neil prefers to have "form-rebar-pour" as one activity. Yet, partitions should be divided into metal studs, drywall, and taping. The rationale is that the studs crew moves faster than the drywall crew. The drywall crew moves faster than the taping crew. The drywall is sensitive to water, whereas taping is sensitive to temperature. Studs are not affected by weather.

31. An activity is weather sensitive if its materials and/or labor are affected by water and/or temperature, i.e., roofing, placing concrete, drywall, taping, paving, outside brickwork, etc.

32. When an activity is going to be affected by weather, the expert system should ask the user whether there is going to be weather protection. The user also should have the ability to override the decisions made by the expert system concerning which activities are or are not weather sensitive.

33. In a multi-story project the cladding should be six to eight floors below the concrete crew, and glazing should be six to eight floors behind cladding.

34. If any part of the design is not finalized, then that part should be an activity. That is, owner-controlled activities which affect construction progress should be included in the network.

35. Individuals of a particular trade want to see a "detailed" description of their tasks with interfaces to the general plan.

36. Since changes to the project always arise, the schedule should be flexible enough to incorporate them.

37. Physical constraints like site access, crane selection, etc., are part of the general conditions.

Cost

38. O'Neil gets paid by progress, but not necessarily progress by construction operations. Usually, a schedule with the least amount of detail is used for billing purposes. This schedule is quite different from the one used to monitor progress. Cost and time should be correlated, but they are not.

39. Cost and schedule are not related because the level of detail or generality in the cost estimate and the construction schedule do not match.

40. Because most of the work is negotiated (guaranteed maximum price) and since O'Neil awards a large number of subcontracts, it is not extremely important that the cost and the schedule correspond.

41. The items O'Neil uses to request payment are basically discipline related (electrical, mechanical) and discipline within area related (2nd floor concrete), as opposed to physical area related, e.g., 3rd floor complete, which entails not only electrical work, but mechanical work, concrete, drywall, and so forth.

42. Front-end loading is a practice difficult to implement in O'Neil because the general conditions and concrete items represent a small portion of the total project cost.

43. Labor-intensive activities should be likely candidates for front-end loading because there is no retainage on what O'Neil pays to the workers, even though owners usually include a retainage on each payment request. On the other hand, labor-intensive activities are very quantitative, and thus, they are difficult to front-end load anyway.

44. By the time the project is 33 percent and 66 percent complete, the contractor should have recovered 25 percent and 75 percent of the contract dollar value, respectively.

45. Money spent on electrical and mechanical activities is not only a function of performance and productivity, but also of delivery of initial equipment. This may give the false impression of front-end loading.

46. An operation such as "installation and hook up of motor control center" can be legally front-end loaded if the procurement of the motor control center is not represented by a single activity. If this is the case, it would be wrong to prorate the cost of such an item or to say that the owner does not pay for it until it is all hooked up.

47. The decision to include the cost of materials with the cost of the actual installation is based on whether O'Neil can be reimbursed for the materials soon after they arrive. For example, reinforcing steel is installed as it arrives and thus can be included in the payment request. On the other hand, an expensive switch gear may be delivered and remain uninstalled for quite some time. In this case O'Neil includes the cost of materials in the delivery phase.

In-Progress Analysis

48. Changes in the schedule should be made when the current schedule no longer can be used to predict what the future is going to be, rather than on a fixed-time basis.

49. If O'Neil cannot show in the schedule where the money is spent, then O'Neil will change it.

50. O'Neil prepares biweekly progress reports on its schedules.

51. The progress on an activity is assessed by calculating the expected real remaining duration.

52. A biweekly variance progress report with all major milestones (concrete, masonry, windows, roofing, and so forth) is utilized as the triggering device to appraise the validity of the current schedule.

53. Variance = original late finish - {current early finish or actual finish} is used as the indicator to flag lagging activities. This parameter compares the current schedule against the original

schedule. It is used instead of the total float, which only takes into account the current schedule, i.e., total float = (current late finish - current early finish).

54. The current schedule is subject to review every time the project manager does not agree with the overall completion date.

55. The current schedule is also subject to review when the variance report shows significant slippage and the project manager does not agree with such slippage. At this time, it is necessary to incorporate into the construction schedule what the project manager knows that is not currently reflected in the schedule.

56. Changes in the current construction schedule definitely imply different logic and/or activities durations.

57. If there is slippage, logic changes most likely will show work done concurrently, though O'Neil would prefer to do it sequentially.

58. "What if" logic scenarios are calculated by changing the preferential logic and comparing the results.

59. If O'Neil needs to extend the duration of critical unfinished activities like "placing slab concrete in a high-rise" based on the progress of the project to this point, it will also modify the logic of other activities (electrical, plumbing, etc.) to compensate for such delays.

60. If the variance report shows the project ahead of schedule, say by 2 weeks, O'Neil will notify the owner of such gain. O'Neil, however, is not likely to commit to some future gain (e.g., if the work keeps moving at the present pace, the project will be four weeks ahead at completion) because the gain has not been realized yet. O'Neil is hopeful that the other 2 weeks will crystalize, but those will be used as contingency for unforseen events.

61. O'Neil is even more reluctant to change the logic of unfinished activities just because the project is ahead of schedule.

62. If there is a schedule slippage, the entire path to which the lagging activities belong should be marked for monitoring.

APPENDIX C:

SCHEDULE EVALUATION EXERCISE

This appendix contains the scheduling knowledge generated by the trio, rookie, and human say-how analyzer. It represents the third-generation human say-how.

Initial Analysis

1. Does it meet the contract requirements?

1.1. Prior to the schedule analysis, a project should be summarized in terms of dollar value, duration, type of facility, building system, construction method, location, type of contract, customer, liquidated damages, degree of criticality, notice to proceed, completion date, owner-furnished equipment and material, material type, extent of landscaping, finish work, specialty work, and trades involved.

a. The type of contract, e.g., fixed price, cost plus, etc., is important because it affects the cost of accelerating the work.

b. The type of facility, e.g., mid-rise apartment building, highway, etc., is important because each project varies in regard to trades, materials, and resources.

c. Liquidated damages indicate the user's perceived importance and criticality of the project.

d. The building system, e.g., steel structure, reinforced concrete, etc., and construction methods should be known to perform meaningful schedule logic analysis.

1.2. Contract schedule specifications should be determined, for example, maximum and minimum activities durations, maximum number of activities, maxium and minimum activities costs, update requirements, submittal log, resource loading, intermediate milestones, software/hardware compatibility.

a. If there is a submittal log, submittals should be tied to the schedule.

b. The intermediate milestone dates are usually found in the contract special provisions, for example, availability of site, and building enclosure.

1.3. The number and name of subcontractors should be identified, so that the percentage of work by subcontractors is calculated. The work of each subcontractor should be identified with a special coding scheme.

1.4. Technical specifications in the contract that affect the construction schedule should also be identified, for example, the concrete curing sequence, required sequence of activities, reshoring, restrictions on painting, paving, and planting season for landscaping.

1.5. Permits and environmental constraints are also important to know about because they are likely causes of project delays.

1.6. A consistency analysis should be performed to ensure that the construction schedule does not violate the contract requirements, for example:

a. Notice to proceed versus planned start

b. Completion date versus planned finish

c. Contract dollar value versus proposed total cost

d. Schedule and technical specifications verification.

2. Is the critical path reasonable?

2.1. Locate intermediate milestones.

2.2. Analyze activities duration.

a. Isolate those with the longest duration for further analysis. A high percentage of critical activities with long durations is not acceptable.

b. Critical activities should have enough detail, that is, have a duration of less than 20 days. This is especially true in the case of finish work, which is a typical cause for delays.

c. Critical or near-critical activities should have a duration that is less than one pay period or that can be easily measured in terms of percentage complete.

2.3. Analyze the logic based on the type of facility, the building system, and the construction methods.

2.4. Analyze the timing of weather-sensitive activities; for weather-sensitive activities scheduled during adverse weather, determine whether temporary weather protection is planned, e.g., heating.

2.5. Analyze the near critical activities, i.e., those with less than 10 days of total float, as closely as the critical activities.

2.6. Analyze the ratio of (critical/total) activities to make sure that it falls within a reasonable min/max range.

a. A small index would indicate too little detail in the critical path and too much detail in noncritical activities.

b. An index that is too high would suggest that float has been sequestered.

2.7. Analyze the ratio of (critical path cost/total project cost) to make sure that it falls within a min/max range.

3. Does the schedule follow "good" scheduling practices?

3.1. Analyze the cash flow curve to make sure it is reasonable. The curve should actually be derived from the schedule activities. A typical "S" curve for midrise buildings would show one-quarter of the cost committed at the end of the first third of the project, and three-quarters of the cost at the end of the second third.

3.2. Analyze the cost distribution throughout the project duration to determine if there is evidence of cash flow front-end loading.

a. Assess if the mobilization costs are reasonable, given the contractor's equipment list.

b. Are there items that are likely to overrun quantities? If so, does their cost appear reasonable in relation to similar items with larger quantities?

c. Are unit prices of early activities greater than unit prices of similar activities scheduled towards the end of the project?

d. Is the percentage of cost committed at one-third and two-thirds of the project greater than one-quarter and three-quarters, respectively? Note: This measure should be applied with caution because the existence of this fact also indicates that the contractor is taking too long to finish out, i.e., it has taken too much time to complete the last one-third of the project. It also indicates that the preceding two-thirds of the project were scheduled too early, thus making the schedule too optimistic.

e. Perform a variance analysis on the $/day index to determine the activities that fall outside an acceptable range, i.e., (1 standard deviation - mean + 1 standard deviation). Such activities may then be correlated with their planned time, i.e., one quarter versus four quarters.

3.3. Determine if the total number of activities in the schedule is manageable, i.e., fewer than 250 activities.

3.4. Determine for each activity if the relationships among duration, cost, criticality, and measurability are consistent.

a. A typical activity duration should be between 5 and 25 days.

b. An activity that cannot be measured should be rejected.

c. Activities with long durations are difficult to measure. A long duration usually indicates the aggregation of many subactivities into a more general one, i.e., the hammock effect. It is difficult to assess progress on these type of activities.

d. An activity that is not easily measured should be divided into more detailed activities.

e. An activity with long duration, low value, and float may be regarded as nonimportant and allowed to keep its duration.

3.5. Subcontractor subnets should be an integral part of the construction schedule. Subcontractor work is a likely cause for delays because of the submittals.

3.6. Avoid splitting activities into misleading categories, e.g., 30 percent complete, 70 percent complete, and 100 percent complete.

a. This practice gives the impression of trying to manipulate float by extending activities duration, which in turn makes the $/day index drop to a level that suggests front-end loading of earlier activities.

b. The practice is not acceptable on the grounds that all activities then would be labeled as "100 percent complete ...". In addition, the detail necessary to assess the percentage complete is lost, thus making it a difficult activity to measure.

4. Conclusions.

4.1. Lack of knowledge about the building system and construction methods make schedule logic analysis extremely difficult.

4.2. Since contract documents are necessary to evaluate construction schedules, a KBS should have access to the technical and schedule specifications and be able to reason about them.

In-Progress Analysis

5. Delay analysis.

5.1. Assess the magnitude of the delay by identifying all activities contributing to the delay or time extension claim.

a. Is it one delay or a combination of delays?

b. Base the analysis on the number of activities whose status is in progress and finished that are already behind, as opposed to projected behind.

c. Determine if there has been an "on-going" delay, or whether it has been discrete.

5.2. Make sure that previously authorized time extensions are reflected in the schedule.

5.3. Identify possible delay causes, for example:

a. Equipment/material acquisitions.

b. Subcontractor.

c. If abnormal weather conditions are present, verify that weather-sensitive work on the critical path is (temporarily or permanently) weather protected.

d. Labor productivity.

5.4. Determine whether the delay is likely to continue.

a. Do unfinished activities have anything in common, i.e., subcontractor, material type, crew, etc., with the activities involved in the delay?

b. Make schedule projections based on the project's performance history.

c. Are activities durations consistently optimistic?

d. Analyze procurement activities that are critical or near critical.

5.5. Comparison analysis across schedules.

a. It should include a variance analysis between current and target schedules.

b. If the comparison is not made against the target schedule, the float analysis may be misleading. For example, if an activity is delayed because of delays in predecessor activities, the absolute value of the total float might still be positive, yet the activity might have negative float if compared against its target.

5.6. Perform a consistency analysis between remaining duration, percent complete, and original duration.

a. If an activity shows 100 percent complete and still remaining duration, it probably means that the work was already paid for, but there is still work left to execute.

b. Update unfinished activities durations based on days remaining, as opposed to percentage complete.

6. How to fix a delay.

6.1. Determine if the purpose is to prevent the delay from reoccurring or to make up for the time lost.

a. Determine the cost of further delays.

b. Perform a job acceleration analysis based on time/cost trade-off.

6.2. Analysis of the critical path.

a. Activities with the longest duration in the critical path should be considered for crashing.

b. Determine if sequential critical activities can be performed concurrently.

c. Work double shifts for strictly sequential activities.

d. In general, work to be accelerated belonging to a general contractor will be easier to control than that belonging to a subcontractor.

6.3. Verify consistency of logic to satisfy weather constraints.

a. Drywall should be tied to exterior closure.

b. Roofing and sheathing should be tied to at least the upper floor drywall.

6.4. Analyze activities whose remaining duration exceeds 20 days.

6.5. Ensure that the ratio of (days to makeup/total days remaining) is less than 0.2

7. Time extensions due to weather.

7.1. Determine if weather was indeed the cause of delay for weather sensitive activities.

7.2. Determine if the weather causing the delays was "very abnormal."

7.3. Determine if weather prevented the contractor from doing weather sensitive work, or lack of weather protection damaged work already in place, i.e., drywall.

7.4. Determine if the building enclosure is tied to weather-sensitive work.

a. If the schedule logic does not reflect this relationship, a time extension for rework may not be granted should the work be damaged.

b. If the schedule logic does not reflect this relationship, a time extension for being unable to work may not be granted. This is based on the grounds that this missing relationship could have had an impact on the project completion date and the contractor chose to hide it.

c. If the schedule logic does not reflect this relationship, temporary weather protection may be necessary.

d. If the schedule logic does reflect this relationship but building enclosure is being delayed, then temporary weather protection may be required.

Overall Analysis

8. Definitions.

8.1. A mid-rise building has between 11 and 19 floors. The cutoff is determined by the need to have another set of elevators after the 19th floor.

9. Preconstruction activities.

9.1. Design completion.

9.2. Building permits.

9.3. Long-lead acquisitions should be represented with the procurement chain: submittal, approval, fabrication/deliver, install. For example, reinforcing steel, structural steel, hollow metal, window systems, skin systems, elevators, mechanical and electrical systems, specialty items, millwork, intelligent building systems, special hardware items, etc.

9.4. Location of a landfill to dispose of material produced by demolitions.

9.5. Identification of subcontractors.

10. Construction activities.

10.1. Foundations. Soil conditions, type of soil, water level, type of foundation, amount of excavation, shoring and support requirements.

10.2. Structure.

a. Fire protection in steel buildings is a "water" application, and the steel gets very cold in winter. Fire protection for steel structures may not be scheduled during winter without excessive dripping and heating costs.

b. Concrete structures are also affected by cold temperatures, however; their treatment may be less expensive because the concrete generates a large amount of heat.

c. Delivery of steel, owner furnished equipment, long-lead deliveries, etc.

d. The structure may be scheduled by tiers.

e. The core construction of a steel structure may be represented by: steel delivery, steel erection, metal deck, supported slab pours, and fireproofing.

f. The core construction of a cast-in-place structure may be represented by rebar delivery, forming, reinforcing, placing concrete, curing, stripping, and moving forms.

g. The core construction of a precast structure may be represented by fabrication and delivery, erection, pouring tie strips, caulking, and sealing.

10.3. Core construction. Elevator lobby, washrooms, stairs.

10.4. Elevators.

a. Permanent power.

b. Delivery, installation, and testing.

10.5. Mechanical and electrical.

a. Procurement and delivery of long-lead equipment.

b. Systems installation.

c. Mechanical rooms should be scheduled as early as possible to give access to the installation crews.

10.6. Site work and site conditions.

a. Logistics, site layout, unusual conditions.

b. Most of the work in these categories is winter sensitive.

10.7. Exterior closure.

a. A building is considered enclosed when weather-sensitive work can proceed and the building can be heated (some work may not be able to start, even if the building is enclosed, unless the building is heated to eliminate humidity). Closure can be temporary or permanent. Permanent closure consists of the roof, the window system, and the building skin.

b. Separate the window system from the skin system. Caulking is an important activity of most skin systems.

c. The building skin of the first and second floors is usually scheduled last.

d. From the schedule logic point of view, the start of building enclosure is not affected by the material of which the skin of the building is made. On the other hand, the time to enclose the building will be directly affected by such material.

e. An activity is considered weather sensitive if it is affected by either moisture, temperature, or water.

f. The following items are weather sensitive: taping, painting, ceramic tile, masonry, acoustical ceilings, etc.

10.8. Finishes. Because the components of drywall partitions are not affected equally by weather, the following sequence of activities is recommended: metal studs, drywall, and taping.

APPENDIX D:

ELECTRONIC BANYAN MESSAGES

This appendix contains a sample of electronic mail messages that were sent among the trio, rookie, and human say-how analyzer. The complete set of messages is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

To: rookie,ke

Cc:

Bcc:

From: Trio@CPM@CPM

Subject: E6: overview of schedule

Date: Tuesday, September 1, 1987 at 2:31:52 pm CDT

Attach:

Certify: N

Forwarded by:

---

a:List the activities on the critical path.

b:Does the schedule completion date meet the contract requirement?

c:Are dollars assigned to activities? If so. is the sum equal to the contract amount?

To: gurus

Cc:

Bcc:

From: Rookie@CPM@CPM

Subject: Resp to E6

Date: Tuesday, September 1, 1987 at 3:01:20 pm CDT

Attach: a:1

Certify: N

Forwarded by:

---

a: see attached

b: yes completion date is met.

c: yes dollars are assigned to activities...

yes the sum is equal to the contract amount..

To: rookie,ke

Cc:

Bcc:

From: Trio@CPM@CPM

Subject: E7: Additional cost information

Date: Tuesday, September 1, 1987 at 2:35:19 pm CDT

Attach:

Certify: N

Forwarded by:

---,----

a: Is there a cash flow curve? If so. does the curve appear reasonable? i e. a slow start, steady increase and taper off (rule of thumb: 1/4 cost at 1/3 time and 3/4 cost at 2/3 time). Is the curve actually derived from the activities?

b: Are all activities dollar loaded? Is the job front end loaded? i.e is the unit cost of early items higher than the unit cost of late items? Is mobilization reasonable given contractor's projected equipment list? Are there items that are likely to overrun quantities? If so. does their cost appear reasonable in relation to similar items with larger quantities?

To: gurus

Cc:

Bcc:

From: Rookie@CPM@CPM

Subject: Resp to E7

Date: Tuesday, September 1, 1987 at 3:19:06 pm CDT

Attach:

Certify: N

Forwarded by:

---,----

a: I am checking your rule of thumb against the project, will advise.....

b: Not all activities have dollar amounts.
Regarding front.end loading, which items should I review ???
Given the list how do I determine what is reasonable ???
Regarding overrun quantities, do you want me to do a quantity take-off ?????

Which items do you want me to look at first, do you have a prioritized list ??????

To: gurus

Cc:

Bcc:

From: Rookie@CPM@CPM

Subject: Resp to E7

Date: Tuesday, September 1, 1987 at 4:08:37 pm CDT

Attach:

Certify: N

Forwarded by:

RE Your analysis of cost and time...

1/3 time = 36% cost

2/3 time = 92% cost

To: all

Cc:

Bcc:

From: Trio@CPM@CPM

Subject: E13 Response to cost & schedule

Date: Tuesday, September 1, 1987 at 4:49:26 pm CDT

Attach:

Certify: N

Forwarded by:

---

a: Project appears to be badly front.end loaded: at 1/3 earnings are @ 45% greater than "rule of thumb". Note that 3 Paint & Finish activities which take 91 calendar days have total value of $13.000 (<$150/day) A closer review of cost loading is warranted. Suggest printout of activities in order of dollar value, listing start date and duration.

b: We need more information on structural system to determine if logic of early activities is reasonable or could be shortened. Also, the contractor has many activities on the critical path with too.long duration (i.e. the paint & finish activity). Recommend you reject the schedule and require more detail on the critical activities.

APPENDIX E:

PERSONAL CONSULTANT PLUS CODE

This appendix includes examples of PC Plus code. The complete set of PC Plus rules and LISP functions is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

The purpose of this rule is to project new durations for a particular class of unfinished activities. These new durations are based on the performance of finished or in-progress activities which belong to the same class. The parameter "concrete-villains," whose value tells PC plus whether a statistically meaningful delay factor was found, has a method associated with it.

This method is triggered when the value of "concrete-villains" is needed. The method calls the LISP function "dynamic" with "slow-csi1='03'" as the argument. This argument corresponds to a name of a dBASE III procedure. The left part of the argument's name represents the assessment of the activity, e.g., slow, and the right part corresponds to the class to which it belongs, e.g., 03.

```
RULE076 [LOOK-AHEAD-RULES]

-------
PREMISE:  ($AND
          (SAME AME CLASS CONCRETE)
          (SAME AME PRODUCTIVITY SLOW)
          (GREATERP
            (VAL1 AME CONCRETE-VILLAINS) 2))

ACTION:  (DO-ALL
          (CONCLUDE AME CONCRETE-LIST-OF-VILLAINS
            (GET-LIST) TALLY 100)
          (CONCLUDETEXT AME LOOK-AHEAD-STATUS
            (TEXT  :LINE :LINE :TAB
              (TEXT TEXTG4) "CONCRETE" :LINE :LINE
                          :TAB "  I   J  OLD  NEW
               ACT. TITLE " :LINE :TAB "          DUR  DUR" :LINE
              (VAL1 AME CONCRETE-LIST-OF-VILLAINS ))

            TALLY 100))
UTILITY:  50

CONCRETE-LIST-OF-VILLAINS [LOOK-AHEAD-PARMS]

-----------------------
TRANSLATION:  ("the list of unfinished activities whose
               duration is extended based on past
               experience")
TYPE:  SINGLEVALUED
EXPECT:  MULTI-LINE-INPUT
UPDATED-BY:  RULE076

CONCRETE-VILLAINS [LOOK-AHEAD-PARMS]

---------------
TRANSLATION:  ("the number of finished/in-progress
               activities experiencing slower than
               anticipated productivity")
METHOD:  (DYNAMIC "slow-csi1='03'")
TYPE:  SINGLEVALUED
EXPECT:  POSITIVE-NUMBER
USED-BY:  RULE078 RULE076
```

APPENDIX F:

dBASE III PLUS CODE

This appendix represents an example of a procedure taken out of the library of dBASE III Plus procedures. The complete library is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

The procedure corresponds to the argument passed to the "dynamic" LISP function of Appendix E. Because the class of activities to which these procedures may be applied forms an integral part of the argument's name, this same dBASE III Plus procedure is executed for different classes of activities. Notice that the results of this procedure are recorded in two files: (1) flag-1.txt and (2) flag-5.txt. These files will subsequently be read by PC Plus to retrieve the procedure's results.

```
PROCEDURE SLOW
*
* This file expands the duration of a CLASS of activities
* based on past data.
* this program assumes that the data base PROGRESS.DBF is created
* by Primavera 2.6
*
* &class is to be replaced with the CLASS desired, e.g.,
*        CSI1 = '03' .AND. BSI1 = '04'
*
PARAMETERS class
*
SELECT 1
USE PROGRESS
GO TOP
STORE 0 TO HOWMANY
SELECT 2
USE BRIDGE
SELECT 3
USE GRALINFO
STORE DATA_DATE TO PROC_DATE
*
* activities in-progress with slow progress
*
SELECT 1
GO TOP
DO WHILE .NOT. EOF()

   SELECT 1
*
   IF PCT > 0 .AND. PCT < 100 .AND. TYPE <> '4' .AND.     ;
      AS <= PROC_DATE .AND.                                ;
      PCT < ZERO ((PROC_DATE - AS),OD_T1) * 100 .AND.     ;
      &class
*
      HOWMANY = HOWMANY + 1
      STORE ((PROC_DATE - AS) / (PCT/100 * OD_T1)) TO DELAY
*
      SELECT 2
      APPEND BLANK
      REPLACE FLAG_1 WITH DELAY
      SELECT 1
   ENDIF
   SKIP
ENDDO
*
* activities finished with slow progress
*
SELECT 1
GO TOP
DO WHILE .NOT. EOF()
```

```
   SELECT 1
*
   IF PCT = 100 .AND. TYPE <> '4' .AND.      ;
      AF <= PROC_DATE .AND.                  ;        1 < ZERO((AF - AS),OD_T1) .AND.
        ;
      &class
*
      HOWMANY = HOWMANY + 1
      STORE (AF - AS) / OD_T1 TO DELAY
*
      SELECT 2
      APPEND BLANK
      REPLACE FLAG_1 WITH DELAY
      SELECT 1
   ENDIF
   SKIP
ENDDO
*
SELECT 2
GO TOP
*
IF HOWMANY = 0
   APPEND BLANK
   REPLACE FLAG_1 WITH HOWMANY
   COPY TO FLAG-1 RECORD 1 DELIMITED ELD FLAG_1
   DELETE ALL
   PACK
ENDIF
*
* calculate delay factor
*
IF HOWMANY > 0
   AVERAGE FLAG_1 TO DELAY_FTOR
   GO TOP
   REPLACE FLAG_1 WITH HOWMANY
   COPY TO FLAG-1 RECORD 1 DELIMITED ELD FLAG_1
   DELETE ALL
   PACK
ENDIF
*
* calculate new duration
*
IF HOWMANY > 0
   SELECT 1
   GO TOP
   DO WHILE .NOT. EOF()
      IF PCT <> 100 .AND.  ;
         &class
         STORE SUBSTR (STR (PNO), 1, 4) TO I
         STORE SUBSTR (STR (SNO), 1, 4) TO J
         STORE SUBSTR (STR (OD_T1), 1, 4) TO OLD
         STORE SUBSTR (STR (OD_T1 * DELAY_FTOR), 7, 4) ;
               TO NEW
```

```
        STORE I + J + OLD + " " + NEW + "       " + TITLE ;
              TO ACTIVITY
        SELECT 2
        APPEND BLANK
        REPLACE FLAG_5 WITH ACTIVITY          SELECT 1
      ENDIF
      SKIP
   ENDDO
   SELECT 2
   GO TOP
   COPY TO FLAG-5 ALL DELIMITED ELD FLAG_5
   DELETE ALL
   PACK
*
ENDIF
QUIT
RETURN
*
```

APPENDIX G:

LISP FUNCTIONS FOR PC PLUS

This appendix shows examples of LISP functions used by PC Plus to send messages to dBASE III and to read dBASE III results. The complete set of functions is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

```
; This function takes a file written by dBASE-III and
; returns a LIST of all its contents.
;
; The function call looks like: (readlst "filename")
;
;  (define readlst
     (named-lambda (readlst file)
      (let ((ifile (open-input-file (string-append file ".txt")))          (ilist #!false))
           (do ((item (read ifile) (read ifile)))
               ((eof-object? item) (close-input-port ifile) ilist)
               (set! ilist (append! ilist
                                    (list ':line
                                    ':tab
                                    (string-append item ","))))))))

   DYNAMIC [FUNCTIONS]
   -------    TEMPLATE:  (VALU)
   TYPE:  EXPRESSION
   TRANSLATION:  (the retrieval of project specific information)
   SOURCE:  (LAMBDA
              (PROCEDURE)
              (TURN-OFF-MONITOR)
              (PICTURE "g:transfer")
              (SET-VIDEO-MODE! 2)
              (TURN-ON-MONITOR)
              (DOS-CHANGE-DRIVE "g:")
              (DOS-CALL "dbaseiii.exe" PROCEDURE 12700)
              (DOS-CHANGE-DRIVE "e:")
              (FORMAT "g:flag-1")
              (TURN-OFF-MONITOR)
              (SET-VIDEO-MODE! 3)
              (LIST
                (READII "g:flag-1")))
```

GET-LIST [FUNCTIONS]

TEMPLATE: ()
TYPE: EXPRESSION
TRANSLATION: (retrieved from the file FLAG-5.TXT)
SOURCE: (LAMBDA
(READLST "g:flag-5"))

APPENDIX H:

FILE TRANSFER COMMUNICATION PROTOCOL

This appendix consists of examples of LISP functions that implement the XMODEM (checksum version) communication protocol. The initial code was written by Professor Jerry Morgan of the Linguistics Department at the University of Illinois. The functions herein have been modified to allow for batch mode communication. The complete set of functions is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

```
(DEFUN xmodem-in (filename ascii-flag &aux (block-nr 0) instream)
                        ; if the ascii-flag is set,
                        ; ascii is translated to explorer
                        ; character set
  (WITH-OPEN-FILE (outstream filename :direction :output)
    (SETQ instream (si:make-serial-stream ))
    (configure-serial-stream instream 19200 8 1 :none)
                        ; then wait until the ibm is set up on the
                        ; other end and looking for a nak
                        ; then send the nak to signal
;;;;    (FORMAT t "Press any key when the ibm is ready to send")
;;;;    (READ-CHAR)
;;;;    (FORMAT t "~%Sending  NAK~%")
    (SEND instream :tyo nak)            ; ready-to-receive
    (SEND instream :clear-input) ; does this do anything useful?
                        ; then read in data 128 characters at a
                        ; time until the ibm sends eot to signal end
                        ; of transmission
    (LOOP while  (xmod-receive-block ascii-flag
                  instream
                  outstream
                  (SETQ block-nr (1+ block-nr)))))
  (send instream :close)
;;;;  (FORMAT t "~%EOT received. Transmission completed.~%")
  )
.DE
.PP
.DS
;;; this function does the work of reading in
;;; the data a character at a time, checking
;;; for errors, and storing in the destination
;;; file.  If EOT was received, nil is returned,
;;; signalling done; else t is returned.

(DEFUN xmod-receive-block (ascii-flag
                        ;if ascii-flag =t do the translation
                        ; from ascii to explorer
                                instream
                                outstream
                                block-nr
                                &aux
                        ; these variables are for the three
                        ; header bytes that precede
                        ; each received block
                                head
                                in-block-nr
                                in-block-compl
                                checksum-in
                                inbuffer
                                block
                                char
                                (checksum 0))
 (prog ()
```

```
(setq inbuffer (MAKE-STRING-OUTPUT-STREAM))
                ;inbuffer is where the data is
                ; stored 1 char at a time
loop (SETQ checksum 0)
                ; if the first character is eot,
                ; the transmission is done
                ; so acknowledge the EOT and quit
(IF (= (SETQ head (SEND instream :tyi)) eot)
    (PROGN (SEND instream :tyo ack)
                                        (RETURN nil)))
(SETQ in-block-nr (SEND instream :tyi))
(SETQ in-block-compl (SEND instream :tyi))
  (SETQ block
                ; inbuffer is really a stream pointer,
                ; not a string, so we need 'block' as
                ; the string to be written to the file.
        (WITH-OUTPUT-TO-STRING (inbuffer)
                ; get 128 chars, if the ascii-flag is on

          (DOTIMES (n 128)
                ; translate them, and store them in the
                ; input buffer; keep a checksum by
                ; simply summing the characters
                ; (before they're translated, of course)
            (SETQ char (SEND instream :tyi))
            (SETQ checksum (make-8-bits (+ char checksum)))
            (IF ascii-flag
                                    (SETQ char (translate-ibm-exp char)))
            (IF (NOT (AND ascii-flag
                ; if it's a LF and ascii-flag is on,
                ; throw it away
                                                        (= 10 char)))
                                    (WRITE-CHAR char
                                                            inbuffer)))))
                ; then read in what the ibm thinks the
                ; checksum should be
(SETQ checksum-in (SEND instream :tyi))
                ; check the header and checksum for errors
(IF (NOT (AND  (= head soh)
                                    (= block-nr in-block-nr)
                                    (= in-block-compl (bit-complement block-nr))))
                ; if there's an error, send nak
                ; as a request for re-transmission
    (PROGN
 ;;;; (FORMAT t "~%Header error on block ~D" block-nr)
                                (SEND instream :tyo nak)
                                (GO loop)))
(IF (NOT (= checksum checksum-in))
    (PROGN
  ;;;; (FORMAT t "~%Checksum error on block ~D" block-nr)
                                (SEND instream :tyo nak)
                                (GO loop)))                    ; otherwise, if header and
checksum
```

```
                    ; are ok, then chances are the
                    ; data is ok, so send ack as a signal
                    ; of successful receipt of the block,
                    ; write the block to the file and
                    ; return t to signal there is
        (SEND instream :tyo ack)
;;;;    (FORMAT t "~%Block ~D received" block-nr)
        (PRINC block outstream)
        (RETURN t)))
```

APPENDIX I:

PROJECT MANAGEMENT SYSTEM PARSER

This appendix contains an ART rule and a LISP function used to parse the received file and translate its contents into ART data structures. The rule and the function assume that data are generated by a specific PMS, e.g., Primavera. More general rules may be desired to interpret data from several PMS. The complete set of ART rules and LISP functions is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

```
(defun interpret-PRIMAVERA-data (instream)
  (with-open-file
    (infile instream :direction :input)
    (read-line infile nil :eof)          ;;; remove headings
    (read-line infile nil :eof)          ;;; remove headings
    (read-line infile nil :eof)          ;;; remove headings
    (let (incoming act-symbol)
      (while (and
              (not (eq (setq incoming (read-line infile nil :eof)) :eof))
              (> (length incoming) 25))

      do

      (setf act-symbol (gensym)) ; generate a symbol for each activity
      (assert

           (act = act-symbol
                = (nil? (read-from-string incoming nil nil :start 0  :end 10)))
           (es = act-symbol
                = (nil? (read-from-string incoming nil nil :start 11 :end 18)))
           (ef = act-symbol
                = (nil? (read-from-string incoming nil nil :start 20 :end 27)))
           (ls = act-symbol
                = (nil? (read-from-string incoming nil nil :start 29 :end 36)))
           (lf = act-symbol
                = (nil? (read-from-string incoming nil nil :start 38 :end 45)))
           (tf = act-symbol
                = (nil? (read-from-string incoming nil nil :start 47 :end 52)))
           (ffl = act-symbol
                = (nil? (read-from-string incoming nil nil :start 53 :end 58)))
           (od = act-symbol
                = (nil? (read-from-string incoming nil nil :start 59 :end 63)))
           (title = act-symbol
                = (nil? (subseq incoming 64 90))))))))


(defrule convert-PRIMAVERA-to-ART
    "A rule to open already transferred PRIMAVERA
     files for input to ART. These files were sent
     to the Explorer from the IBM. The LISP function
     get-PMS-files does the transfer work on the TI end."
  ?x <- (read-data ?stream1)
  ?y <- (read-logic ?stream2)
  =>
  (retract ?x ?y)
  (interpret-PRIMAVERA-data ?stream1)
  (interpret-PRIMAVERA-logic ?stream2))
```

APPENDIX J:

ART CODE FOR THE SEMANTIC NETWORK

This appendix shows examples of ART schemata used to define the semantic network. The complete set of ART and GoldWorks definitions is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

```
(defschema class-of
        "inheritance relation with behavior equal to is-a"
  (instance-of inh-relation)
  (transitivity
    (repeat (step class-of $) 1 inf))
  (inverse has-classes))

(defschema affected-by
        "NON-inheritance relation"
  (instance-of relation)
  ;;; (slot-what copy-value)     add if necessary
  (inverse has-effect-on))

(defschema concrete
        "concrete object"
  (class-of csi)
  (affected-by snow rain cold ;;; weather
               labor          ;;; productivity
               quake)         ;;; catastrophe
  (has-primitives forming
                  reinforcing-steel
                  placing-concrete
                  stripping
                  curing))

(defschema Primavera-project
        "typical Primavera project"
  (is-a task)
  (activity-description "Execution of Primavera Schedule")
  (task-level 1))

(defschema primitive-of
        "NON-inheritance relation"
  (instance-of relation)
  (inverse has-primitives))

(defschema sub-task-of
        "schema that allows hammock activities"
  (instance-of inh-relation)
  (transitivity-generate-function standard-sub-task-of)
  (transitivity (repeat (step sub-task-of $) 1 inf))
  (inverse has-sub-tasks))

defschema critical-path-sub-task-of
        "relation for critical path flagging"
  (instance-of inh-relation)
  (transitivity-generate-function standard-critical-path-sub-task-of)
  (transitivity (repeat (step critical-path-sub-task-of $) 1 inf))
  (inverse has-critical-path-sub-tasks))
```

APPENDIX K:

CPM-KERNEL DATA STRUCTURES

This appendix contains an overview of the CPM-Kernel data structures and procedures utilized. The initial CPM-Kernel code which resulted in an ART implementation was submitted by Mr. Glenn M. Yoshimoto of Lockheed Corp. at the Workshop on Expert Systems for Construction Scheduling [O'Connor 87]. The ART implementation has been significantly enhanced by redesigning the whole system. This new system has been implemented in the GoldWorks environment. Two versions currently exist: (1) 80 percent GoldWorks and 20 percent Common LISP and (2) 20 percent GoldWorks and 80 percent Common LISP. The parameters in contention between these versions are: (1) execution speed, (2) readability of code, and (3) overhead of tool features. The complete set of ART components, GoldWorks forms, and Common LISP functions is available on a 5-1/4-in. floppy disk (ASCII format) and may be obtained from

Department of Civil Engineering
Virginia Polytechnic Institute
and State University
Blacksburg, Virginia 24061-0105
Attn: Prof. Jesus M. De La Garza

ART Data Structures and Procedures for Scheduling

Schemata are selected to represent activities and schedules because their inheritance properties are useful for representing concepts that can be aggregated at many levels of abstraction. For example, this CPM-Kernel allows the representation of hammock activities.

An activity to be scheduled is created as an instance of the schema "task" and this activity's subtasks hierarchies, descriptions, durations, precedence specifications, and resource requirements are input for scheduling. This input results in schema facts that are asserted into the data base. These assertions activate rules that together enable procedures that compute the schedule.

The rule "propagate-forward-pass-duration-constraint" propagates a duration within an activity whose earliest finish time is not consistent with its earliest start and duration. Other rules exist to propagate logic precedence relationships.

The precedence relationships implemented in ART are (1) finish to start (FS), (2) start to start (SS), (3) start to finish (SF), and (4) finish to finish (FF). The form of the precedence specification and notation used is

(Predecessor, Relation, Minimum Delay, Maximum Delay) .

The last three elements are concatenated into a symbol that represents the instance name of a precedence relation. Maximum delays are not currently implemented. NIL means not applicable. FS-5-NIL, for example, means that the preceding activity must finish at least five units of time before the succeeding activity can start.

```
(defschema task
        "complete data about activities according to name"
  (is-a activity)
  (task-number)              ; internal control number
  (task-level)               ; depth in the hierarchy, this level is "0"
```

```
  (task-reference-number)   ; for example, Work Breakdown Structure
  (points-to)
  (points-from)
  (followed-by)
  (preceded-by)
  (duration)
  (earliest-start)
  (earliest-finish)
  (latest-start)
  (latest-finish)
  (resource-start)
      ; start due to resource allocation
  (resource-finish)
      ; finish due to resource allocation
  (slack-time)
      ; float within time window for activity execution
  (synchronized-with)
      ; activity that it is synchronized with
  (must-share-a-resource)
      ; activity with which it must share a resource
  (common-unshared-resource)
      ; activity with which it must not share a resource
  (precedence-specs)
      ; this is a slot with an ART sequence value
      ;  with arguments: (?parent ?rel-name ?min-delay ?max-delay)
  (requires-resource))
      ; this is a slot with an ART sequence value
      ; with arguments: (?resource ?proficiency ?how-many)

(defrule propagate-forward-pass-duration-constraint
      "given a new earliest start for an activity,
          propagate the earliest possible finish for the activity"
  (phase cpm-forward-pass)
  (earliest-start ?task ?time1)
  (duration ?task ?duration)
  (earliest-finish ?task ?time2&:(< ?time2 (+ ?time1 ?duration)))
  =>
  (modify
    (earliest-finish ?task =(+ ?time1 ?duration))))
```

Precedence relations are also represented as schemata. The schema "CPM-relation" has the FS, SS, SF, and FF types. The latter have instances for each precedence relation defined by user input. This approach allows future additions of other kinds of relations, based on resource usage. It also permits experimentation with reasoning based on the kind of relation.

```
      (defschema cpm-relation
              "class of all CPM relation types"
        (instance-of relation)
        (inverse cpm-relation)
        (relation-name)
        (points-to)
        (points-from)
        (min-delay)
        (max-delay))
```

The CPM control information is stored in a LISP array. This array is accessed for information about the computation state. The diagonal is initialized with parent and child counts in the forward and backward pass, respectively. These counts are decremented as processing is continued. When the count has been decremented to zero for an activity, then that activity is ready to be computed and a fact is asserted which enables computation of the activity to proceed. Other rules compute float time and identify activities on the critical path.

ART Data Structures and Procedures for Resource Allocation

[Jackson 86] describes a technique by which resource allocation and resource contention can be solved using the ART's viewpoint mechanism. This section presents that technique.

Resources are defined in schemata as are tasks. The schema "resource" defines resources as a class of objects, each of which has attributes like resource-description, units-existing, and units-of-measure.

```
(defschema resource
        "taxonomy of resources and complete
            data about them according to name"
  (resource-description)
  (units-existing)
        ; number of units that exist on-board
  (units-of-measure)
  (resource-number)
        ; internal number assigned by ART
  (resource-shortage (none none))
        ; this is a slot with an ART sequence value
        ;  with arguments (?shortage ?resource)
  (users                ; who and what uses the resource
    (slot-how-many multiple-values))
  (locations            ; where are they located
    (slot-how-many multiple-values)))
```

Since each activity has been assigned a default usage time window, the next step is to determine whether any resource shortages exist in the "root" viewpoint. A LISP array is used as the method for representing the total resources required by all activities at each point in time. This array is utilized to identify shortages. Its dimensions are the number of resource pools and time points. Each entry in the array represents the number of units still available in a resource pool at a point in time.

Rules assign an internal number to each required resource, count the total number of different resources, and create the resource availability array. This availability array represents one context in which resources have been allocated in some manner. The ART viewpoint-based solution generates multiple resource allocation contexts, each showing the current state of resource availability at any given point in the computation state where availability status is being recomputed. Multiple availability arrays are required. These arrays are identified by the name of the viewpoint in which they are created. They are accessed by hashing from a space of ART viewpoint names. A rule creates such a hash table of arrays and places the initial array into the hash table by hashing to the "root" viewpoint name. This rule also sets up a usage array which has as its dimensions the total number of activities and resources. It contains the resource units required by each activity.

Another rule initializes the "root" viewpoint's availability array by calling a LISP function. It hashes to the "root" viewpoint's array, and stores the size of the resource pool, at every time point in

the array for that resource. This rule fires multiple times, once for each resource pool, so that the array contains the number of units of all resources that are available for each point in time.

The allocation process is started by the rule "imagine-task-gets-resource." This rule fires for very activity and resource combination, and asserts a usage schema with defined-usage start and finish times for each of them.

This rule also calls a LISP function that hashes to the viewpoint array and subtracts the activity's claimed number of resources from the pool over the activity's usage window. When this rule has fired for all activities, any shortages are identified by negative entries in that viewpoint's availability array. The negative entry positions correspond to the resource number and the time points when shortages exist. The concept of shortage is extended from a single point to a series of contiguous points. That is, shifting of activities occurs only past the full extent of the shortage. This practice sacrifices completeness for efficiency.

```
(defrule imagine-task-gets-resource
      "if an activity needs a resource, imagine that it gets
       that resource for the activity's duration. This takes
       place at the "root" node, building a viewpoint in
       which all activities have all their resources, if possible.
       A usage schema is instantiated to describe this
       assignment, and the usage and availability arrays
       are updated."
  (declare (salience 10000))
  (viewpoint @?root                    (phase resource-allocation)
        (requires-resource ?task (?res ?kind ?num))
        (resource-instance-of ?kind ?res)
        (units-existing ?kind ?units)
        (task-number ?task ?tnum)
        (resource-number ?kind ?rnum)
        (u-start ?task ?ustart)
        (u-finish ?task ?ufinish)
        (not (needs ?task ?rnum ?)))
  =>
  (bind ?usage (task-resource-concat ?task ?res ?num)) ; LISP function
  (assert
    (resource-instance-of ?usage ?usage)
    (usage-quantity ?usage ?num)
    (needs ?task ?rnum ?usage)
    (usage-start ?usage ?ustart)
    (usage-finish ?usage ?ufinish)
    (usage-duration ?usage =(- ?ufinish ?ustart)))
  (setf (aref *usage-array* ?tnum ?rnum) ?num)
              ; assumes only one usage per
              ; task/resource pair.
  (revise-available-resource
       ?current-alloc ?rnum ?num ?ustart ?ufinish)) ; LISP function
```

The rule "shift-a-task" matches against shortage facts and sprouts viewpoints which consider either of two alternatives: (1) shifting the activity usage-start time past the extent of the shortage or (2) leaving the activity usage-start where it is. Both are required since either may lead to a solution in combination with the shifting of other activities that are involved in the same shortage.

These alternative viewpoints are next merged by another rule in various combinations representing the combined actions to eliminate a shortage.

For each newly created viewpoint, the rule "perform-shift-composition" updates the viewpoint's availability array to reflect the results of performing the alternative activity shifting contained in that viewpoint.

Finally, a rule recognizes a solution. In the event that a solution is not found and all alternatives are exhausted, the process terminates with a message to that effect.

```
(defrule shift-a-task
      "if an activity is part of a resource shortage, shift its allocation
         to try to eliminate the contention problem"
  (declare (salience 5000))
  (viewpoint @?vp-alloc
          (shortage ?sfname ?res ?start-point ?end-point ?vp-alloc)
          (phase resource-allocation)
          (u-finish ?task ?uf&:(< ?start-point ?uf))
                  ; no resources at time = finish
          (u-start ?task ?us&:(<= ?us ?end-point))
                  ; shortage overlaps with window usage
          (needs ?task ?res ?usage)          (not (task-shifted ?sfname ?task ?res ?)))
  =>
  (at ?vp-alloc
      (assert (task-shifted ?sfname ?task ?res ?vp-alloc))
      (sprout
     (assert    ; shift the activity just past this shortage
       (shift-task-to ?sfname ?task ?res ?usage
                    =(1+ ?end-point) ?vp-alloc)
              ; =(1+ ?end-point) should be correct
                ; based on the convention that usage extends
                ; from the start of an activity through
                ; the activity but not including its finish time
       (allocation-child-of ?res ?vp-alloc ?current-alloc)))
      (sprout
     (assert    ; also, consider not shifting the activity at all,
                ; because this move combined with shifting some
                ; other activity, this may get rid of the shortage
       (shift-task-to ?sfname ?task ?res ?usage ?us ?vp-alloc)
       (allocation-child-of ?res ?vp-alloc ?current-alloc)))))

(defrule perform-shift-composition
      "within a merge node, perform all the shifts
         which have collected there to try to eliminate
         a particular shortage."
  (declare (salience 1000))
      ; Find an activity to shift and shift it within this merge node.
      ; Note that to avoid an infinite loop, a "shifted-task-to"
      ; fact is asserted. The alternative, retracting the
      ; "shift-task-to" fact, can unfortunately cause ART to go
      ; into Debug trying to create weird new merges under the
      ; influence of the "merge-shifted-tasks" rule, above.
```

```
(viewpoint @?vp-alloc
        (all-tasks-shifted ?sfname ?res ?vp-alloc)
        (phase resource-allocation)
        (shift-task-to ?sfname ?task ?res ?usage ?new-start ?vp)
        ?uff <- (u-finish ?task ?uf)
        ?usf <- (u-start ?task ?us)
        (duration ?task ?duration)
        (task-number ?task ?tnum)
        (not (shifted-task-to ?sfname ?task ?res
                              ?usage ?new-start ?vp)))
=>
(at ?vp-alloc
    (assert     ; assert the activity is shifted,
                ; whether anything else is done or not.
   (shifted-task-to ?sfname ?task ?res ?usage ?new-start ?vp))
    (bind       ; if the activity has not been shifted,
                ; then don't do anything else.
   ?dt (- ?new-start ?us))
    (if (< 0 ?dt)
     then       ; the activity has been shifted
     (at ?vp-alloc
         (retract ?uff ?usf)
         (assert                        (u-start ?task ?new-start)
          (u-finish ?task =(+ ?new-start ?duration)))
         (modify        ; added to update usage schema
          (usage-start ?usage ?new-start)
          (usage-finish ?usage =(+ ?new-start ?duration)))
                    ; release resources from old-start to, but not
                   ; including, new-start
         (release-task-resources ?vp-alloc ?tnum ?res ?us ?new-start)
                     ; LISP function
                    ; claim new resources from old-finish to, but not
                    ; including, new-finish
         (claim-task-resources ?vp-alloc ?tnum ?res ?uf (+ ?uf ?dt))
                     ; LISP function
         )))
```

GLOSSARY

activation: An entry in the agenda. It is a notation that a rule has matched a specific set of facts and therefore should fire. A rule that matches various sets of facts may generate several activations at the same time.

active values: Methods or functions which are triggered by certain events in the knowledge base. Also know as "demons." Triggering events can be the following:

- When there is a change in the slot value to which the active value unit is attached (but when there is a reference to the slot value to which the active value unit is attached).
- When the active value unit is attached to a slot.
- When the active value unit is removed from a slot.

agenda: The list of current activations competing for an opportunity to fire. In any particular agenda only one rule will actually fire. After the rule fires, a new agenda is generated.

artificial intelligence: A field of computer science in which a program solves a problem using heuristics in a way analogous to human reasoning.

atom: An element of a list that is indivisible into further elements. For instance, in the list (A (BC)) the element A is an atom, but the element (BC) is not. In practice, an atom is either a symbol or a number.

backward chaining: Reasoning from an unproven hypothesis backwards in search of the facts that would prove it. In an expert system, backward chaining is used to supply needed facts to the data base, either by finding the information in another form or by interrogating the user.

bug: Flaw in a program.

car: (Also see **cdr**.) Acronym for "contents of address register." A very primitive LISP function that steals the first element out of a list. The car of the list (A B C) is A.

cdr: Pronounced "could-er," stands for "contents of decrement register." A LISP primitive similar to car, except cdr acts to throw away the first element of a list. The cdr of the list (A B C) is (B C).

chaining: See **forward chaining** and **backward chaining**.

control pattern: A pattern in the left-hand side of a rule that matches a unique fact. The fact is used to control the activity of the rule by turning it on and off. If the fact is present in the data base, the rule can fire. If the fact is not available, the rule cannot fire.

constraint rule: Specifies that its pattern is never allowed to occur in a valid solution.

data driven: Symbolic reasoning processes which are triggered by changes in data and which then cause other processes to occur. Forward chaining is sometimes referred to as data driven because assertions made by a rule can trigger other rules whose premises match the assertion.

declarative information: (Also see **procedural information**.) Factual data.

domain: Application area served by an expert system. For instance, an expert system might serve the domain of mathematical problem solving or the domain of mineral exploration or the domain of financial planning.

domain expert: Endowing an expert system with knowledge about a particular domain sufficient for the system to model expert decision-making processes requires the participation of someone with expertise in the domain being modeled. This person is the domain expert. He or she often works with a knowledge engineer familiar with building expert systems to express domain-specific information for that expert system.

expert system: Computer software which models the decision-making processes of a person with expertise in a particular domain or field. Expert systems generally are constructed of several components: a knowledge base, containing domain-specific procedural, declarative, and structural information; a set of rules for reasoning over the knowledge; several reasoning mechanisms for generating new knowledge; and a specialized user interface. Expert systems also generally have built-in support for explaining the reasoning processes and conclusions.

fact: A fundamental unit of knowledge.

forward chaining: Reasoning from facts to conclusions. In an expert system, a forward-chaining rule detects certain facts in the data base and takes an action because of them.

frame: See **schema**.

function: Subroutine that performs some action on its arguments and returns some value as a result. For instance, in the example (Product 2 3) the function "Product" multiplies its arguments "2 and 3" and returns a value of "6" as a result.

goal-driven: Typically refers to backward-chaining reasoning. Goal-driven processing starts with an explicit goal, matches the goal to rules or frames, and then attempts to prove or achieve the goal by proving or instantiating the rule or frame.

heuristic: Loosely, a rule of thumb. A heuristic rule guides in a direction which promises success but does not guarantee it. Success of a heuristic is not guaranteed, which is why a problem that can be solved by one algorithm requires many heuristics. Human expertise, which we seek to capture in expert systems, is much more likely to take the form of heuristics than algorithms.

hierarchy: (Also see **Taxonomy**.) Classification of objects according to assumed relationships such that the relationships imply directional links indicating incrementally inclusive levels within the classification. For example, a taxonomic hierarchy of automobiles might be composed of foreign cars and domestic cars. Foreign cars might be composed of Toyota, Volvo, and Nissan; domestic cars might be composed of Chrysler, GM, and Ford. GM might then be composed of Chevrolet, Pontiac and Cadillac. In this way, detail can be more specific in the wider base of the hierarchy; and automobiles, as a parent of every object in this hierarchy, must maintain a more general level of detail.

inference: A step in building a logical chain, usually expressed as "IF this is true, THEN that must be true also."

inheritance: Mechanism by which hierarchically higher level units in a knowledge base can share information with lower level units in the knowledge base. The relationship is analogous to a parent-child relationship; the parent units have information in their slots which the child units

inherit. For example, a higher level unit describing the class of objects Transistor might have a slot for NUMBER OF LEADS. That slot could be given the value 3 at that higher level, and lower units representing individual transistors could automatically inherit that value in their NUMBER OF LEADS slot. However, if a particular transistor class or member has some other NUMBER OF LEADS, the value can be changed explicitly in the unit representing that object.

instantiation: Giving something an instance or example. A pattern becomes instantiated when it is matched by a fact. There is a different instantiation of that pattern for each fact that matches it. When all the patterns in a rule become instantiated, we say that the rule is activated.

knowledge acquisition: Process of gathering declarative, procedural, and structural information from a domain expert for building a knowledge base. This is often the most difficult process in building an expert system for two reasons: Experts sometimes find it difficult to explicitly define and articulate the knowledge they use and their own reasoning techniques, and the experts' time is usually at a premium.

knowledge domain: Area of interest for a particular application. Typically, the knowledge domain must be bounded to include only information useful in reasoning about that domain.

knowledge engineer: A person who implements an expert system. A knowledge engineer must possess two essential skills. He or she must be able to interview experts to obtain the raw knowledge from which to structure the knowledge base and formulate the rule base. He or she must also have the programming skills necessary to convert the raw knowledge into a form a computer can understand.

knowledge engineering: The software engineering discipline focusing on constructing expert systems.

LHS: Left-hand side.

LISP: Acronym for "list processor." A programming language based on a "list" construct. LISP provides flexible representation of data and algorithmic procedures as symbolic expressions; manipulation of these symbols is well-supported in LISP. Functions in LISP programs can be interpreted, in addition to being compiled, which facilitates rapid prototyping when constructing a knowledge-based system.

LISP-machine: Computer with specially-designed architecture optimized for handling constructs in the LISP programming language. LISP-machines usually have high-resolution, bit-mapped display screens and a mouse for pointing at objects displayed on the screen.

list: An ordered sequence of elements, usually found within a pair of matching parentheses. For instance, (A B C) is a list composed of the elements A, B, and C. The first element in the list is usually a function, while the other elements are arguments to be acted upon by that function.

matches: Set of facts that instantiates the first n patterns of a rule. There may be any number of matches through pattern n. When ART assembles a match containing all the facts needed to instantiate all the patterns of the rule, we say the rule is activated. When a rule fires, you may be sure a match was responsible.

message: Object-oriented programming involves associating behavioral information with each object. A specific behavior is evoked by sending a message to the object requesting that behavior or the result of that behavior.

method: LISP function which defines how to determine the value of the slot it is associated with or how to execute a behavioral action associated with that slot.

nil: An empty list in LISP, such as (). A LISP predicate returns fBT or some value when true, and fBnil when false.

object: (Also see **schema**.) Structure of information which describes a physical item, a concept, or an activity. Each object is represented as a frame, containing declarative, procedural, and structural information associated with the object.

object-oriented programming: Frame-based knowledge representations can support the association of behavioral information with object descriptions. The behavior descriptions are in the form of LISP procedures or classes of production rules. The behavior is evoked by sending a message to the object or by accessing or changing an attribute of the object. This style of behavior specification, called object-oriented programming, has many advantages over traditional function-oriented programming and is particularly well-suited for building simulation models. It allows much of the information in a program to be stored declaratively, in the frames, where it is easily accessible, understandable, and modifiable.

pattern: A description of a fact.

procedural information: (Also see **declarative information**.) Describes a series of steps to be taken to calculate, determine, or infer new data.

predicate: Function which returns a truth value. Predicates are used to select among conditional alternatives.

recursion: Process by which a function calls itself. "If I can solve the problem in a somewhat smaller instance than the one I am faced with, perhaps I can use such a solution in the larger instance." This is the notion of recursion--the inclusion in a procedure of the procedure itself.

RHS: Right-hand side. Rule A procedural response triggered by a pattern is a rule. From an intuitive standpoint this is an IF-THEN situation. IF pattern is matched, THEN schedule procedure for execution. In practice, a rule that has been activated by a pattern match may be in competition with other activated rules. The inference engine decides which of the activated rules should be executed, and in what order to execute them.

s-expression: In general, a symbolic expression. Strictly speaking, either a list or an atom.

salience: Importance. The salience of an activation is the measure of its importance when competing with other activations on the agenda. Within a set of competing activations, the one with the highest salience will fire first.

schema (pl. **schemata**): A structure within a knowledge base that relates objects or classes of objects that share certain properties. Schemata capture information in two ways: by storing attributes of the concept (Rover weighs 26 lbs) and by relating the concept to similar concepts (Rover is an instance of the general concept "dog").

semantic net: Model of human associative memory. Each semantic net consists of nodes and links between the nodes which represent relationships.

slot: A place to store information within a schema.

taxonomy: (Also see **inheritance**.) Most domains require the representation of the domain objects in a taxonomy, which is the classification of objects according to assumed relationships. Frame-based representations support taxonomies by allowing the objects' attributes to be inherited from the objects' classes to subclasses and members.

wild card: Special symbol that may be used in constructing a pattern to match against the knowledge base. Depending on the qualities of the specific wild card used, ART will match it against any of a wide variety of items in the knowledge base. For instance, the pattern ($? 0 $?) will match any list with a zero in it because the wild card $? matches any sequence of elements in a list. The pattern actually specifies (any list elements, zero, any list elements).

CITED REFERENCES

Avots, I., "Application of Expert Systems Concepts to Schedule Control," *Project Management Journal*, Vol 16, No. 1 (March 1985), pp 51-55.

Buchanan, B. G., "Expert Systems: Working Systems and the Research Literature," *Expert Systems--The International Journal of Knowledge Engineering*, Vol 3, No. 1 (January 1986), pp 32-51.

Buday, R., "Carnegie: Schooled in Expert Systems," *Information Week*, Vol 63 (April 1986), pp 35-37.

Clayton, B. D., *ART--Programming Tutorial, Vols 1, 2, 3, and 4* (Inference Corporation Los Angeles, CA, 1987).

Clough, R. H., *Construction Contracting* (4th Ed) (John Wiley & Sons, 1981).

Cugini, J. V., "Programming Languages for Knowledge-Based Systems," National Bureau of Standards Special Publication 500-145, *Computer Science Technology* (February 1987).

Davis, R., "Diagnostic Reasoning Based on Structure and Behavior," *Journal of Artificial Intelligence*, Vol 24 (1984), pp 347-410.

Denning, P. J., "The Science of Computing--Expert Systems," *American Scientist*, Vol 74 (January-February 1986), pp 18-20.

Feigenbaum, E. A., and P. McCorduck, *The Fifth Generation: Artificial Intelligence and Japan's Challenge to the World* (Addison-Wesley, 1983).

Fenves, S. J., "Expert Systems in Civil Engineering, State of the Art," *Proceedings of the Fourth International Symposium on Robotics & Artificial Intelligence in Building Construction, Haifa, Israel* (June 1987).

Forgy, C. L., "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Journal of Artificial Intelligence*, Vol 19 (1982), pp 17-37.

Freiling, M., J. Alexander, S. Messick, S. Rehfuss, and S. Shulman, "Starting a Knowledge Engineering Project: A Step-by-Step Approach," *Artificial Intelligence Magazine*, Vol 6, No. 3 (1985), pp 150-164.

Gray, C., "Intelligent Construction Time and Cost Analysis," *Journal of Construction Management and Economics*, Vol 4 (1986), pp 135-150.

Hart, A., "Knowledge Elicitation: Issues and Methods," *Computer-Aided Design*, Vol 17, No. 9 (November 1985), pp 455-462.

Hayes-Roth, F., "Rule-Based Systems," *Communications of the ACM*, Vol 28, No. 9 (September 1985), pp 921-932.

Hayes-Roth, F., D. A. Waterman, and B. L. Douglas (Ed.), *Building Expert Systems* (Addison-Wesley, 1983).

Hendrickson, C., C. Zozaya-Gorostiza, D. Rehak, E. Baracco-Miller, and P. Lim, *An Expert System Architecture for Construction Planning*, Publication No. EDRC-12-07-87 (Department of Civil Engineering, Carnegie-Mellon University, December 1986).

Hoffman, R. R., "The Problem of Extracting the Knowledge of Experts from the Perspective of Experimental Psychology," *Artificial Intelligence Magazine*, Vol 8, No. 2 (1987), pp 53-67.

Ibbs, C. W. Jr., *Proceedings of a Workshop for the Development of New Research Directions in Computerized Applications to Construction Engineering and Management Studies* (University of Illinois, Construction Research Series Technical Report 19, 1985).

Ibbs, C. W., Jr., "Future Directions for Computerized Construction Research," *Journal of Construction Engineering and Management, ASCE*, Vol 112, No. 3 (September 1986), pp 326-345.

Intellicorp, *KEE 3.0--The Knowledge Engineering Environment Technical Summary* (IntelliCorp, Mountain View, CA, 1985).

Jackson, P. C., and M. C. Maletz, "Critical Path Resource Allocation Using ART Viewpoints," *Proceedings of the 6th International Workshop on Expert Systems and Their Applications, Avignon, France* (April 1986), pp 405-415.

Kim, S. S., M. L. Maher, R. E. Levitt, M. F. Rooney, T. J. Siller, and S. G. Richie, *Survey of the State-of-the-Art Expert/Knowledge-Based Systems in Civil Engineering*, Technical Report, P-87/01/ADA175257 (U.S. Army Corps of Engineers Construction Engineering Research Laboratory, October 1986).

Kitzmiller, C. T., and J. S. Kowalik, "Coupling Symbolic and Numeric Computing in Knowledge-Based Systems," *Artificial Intelligence Magazine*, Vol 8, No. 2 (1987), pp 85-90.

Kneale, D., "How Coopers & Lybrand Put Expertise Into its Computers," *Wall Street Journal* (November 14, 1986).

Kunz, J. C., T. Bonura, R. E. Levitt, and M. J. Stelzner, "Contingent Analysis for Project Management Using Multiple Worlds," *First International Conference on Applications of Artificial Intelligence to Engineering Problems, Southampton, United Kingdom* (April 1986).

Levitt, R. E., and J. C. Kunz, "Using Knowledge of Construction and Project Management for Automated Scheduling Updating," *Project Management Quarterly*, Vol 16, No. 5 (December 1985), pp 57-76.

Levitt, R. E., "Expert Systems in Construction," in S. S. Kim et al. (Eds.), *Survey of the State-of-the-Art Expert/Knowledge-Based Systems in Civil Engineering*, Technical Report, P-87/01/ADA175257 (U.S. Army Corps of Engineers Construction Engineering Research Laboratory, October 1986).

Levitt, R. E., and J. C. Kunz, "Using Artificial Intelligence Techniques to Support Project Management," *AI EDAM*, Vol 1, No. 1 (1987), pp 3-24.

Lopez, L. A., and S. L. Elam, "SICAD: A Prototype Expert System for Conformance Checking and Design," *Proceedings of the Third Conference on Computing in Civil Engineering, San Diego, CA* (April 1984), pp 84-94.

McGartland, M. R., and C. T. Hendrickson, "Expert Systems for Construction Project Monitoring," *Journal of Construction Engineering and Management, ASCE*, Vol 111, No. 3 (September 1985), pp 293-307.

Mettrey, W., "An Assessment of Tools for Building Large Knowledge-Based Systems," *Artificial Intelligence Magazine*, Vol 8, No. 4 (1987), pp 81-89.

Michie, D., "Machine Learning and Knowledge Acquisition," in D. Michie and I. Bratko, Eds, *Expert Systems -Automating Knowledge Acquisition--AI Masters Handbook* (Addison-Wesley Publishers, 1986).

Mittal, S., and C. L. Dym, "Knowledge Acquisition from Multiple Experts," *Artificial Intelligence Magazine*, Vol 6, No. 2 (1985), pp 32-36.

Nay, L. B., and R. D. Logcher, "An Expert System Framework for Analyzing Construction Project Risks," Technical Report CCRE 85-2 (MIT Department of Civil Engineering, 1985).

Niwa, K., and M. Okuma, "Know-How Transfer Method and its Application to Risk Management for Large Construction Projects," *IEEE Transactions on Engineering Management*, Vol EM-29, No. 4 (November 1982), pp 146-153.

Niwa, K., and K. Sasaki, "A New Project Management System Approach: The Know-How Based Project Management System," *Project Management Quarterly*, Vol 14, No. 1 (March 1983), pp 65-72.

Niwa, K., K. Sasaki, and H. Ihara, "An Experimental Comparison of Knowledge Representation Schemes," *Artificial Intelligence Magazine*, Vol 5, No. 2 (1984), pp 29-36.

O'Connor, M. J., G. E. Colwell, and R. D. Reynolds, *MX Resident Engineer Networking Guide*, Technical Report P-126/ADA116730 (U.S. Army Corps of Engineers Construction Engineering Research Laboratory, April 1982).

O'Connor, M. J., and J. M. De La Garza (Eds.), *Workshop on Expert Systems for Construction Scheduling*, USACERL Conference Proceedings, P-87/13/ADA84642 (U.S. Army Corps of Engineers Construction Engineering Research Laboratory, August 1987).

Ponce de Leon, G., "Schedule Submittals: To Approve or Not to Approve," *Strategem*, Vol 2, No. 1 (Fall 1984).

Prerau, D. S., "Knowledge Acquisition in the Development of a Large Expert System," *Artificial Intelligence Magazine*, Vol 8, No. 2 (1987), pp 43-51.

Richer, M. H., "An Evaluation of Expert System Development Tools," *Expert Systems--The International Journal of Knowledge Engineering*, Vol 3, No. 3 (July 1986), pp 166-183.

Trimble, G., A. Bryman, and J. Cullen, "Knowledge Acquisition for Expert Systems in Construction," *Proceedings of the 10th Triennial Congress of the International Council for Building Research, Studies and Documentation, CIB86 Advancing Building Technology*, Vol 2 (Washington, DC, September 1986) pp 770-777.

Waterman, D. A., "How Do Expert Systems Differ From Conventional Programs?" *Expert Systems - The International Journal of Knowledge Engineering*, Vol 3, No. 1 (January 1986), pp 16-19.

Williams, C., *ART--The Advanced Reasoning Tool: Conceptual Overview* (Inference Corporation, Los Angeles, CA, 1985).

Wilson, J. L., *Proceedings of a Workshop on Construction Automation: Computer-Integrated Construction* (Department of Civil Engineering, Lehigh University, April 1987).

Yoshimoto, G. M., "Applications of the Automated Reasoning Tool at Lockheed," *Conference on Circuits, Systems, and Computers, IEEE Computer Society, Pacific Grove, CA* (November 1985).

UNCITED REFERENCES

Davis, E. W., "CPM Use in Top 400 Construction Firms," *Journal of Construction Management, ASCE*, Vol 100, No. CO1 (March 1974), pp 39-49.

De La Garza, J. M., and C. W. Ibbs, Jr., "A Knowledge Engineering Approach to the Analysis and Evaluation of Vertical Construction Schedules," *Proceedings of the 10th Triennial Congress of the International Council for Building Research, Studies and Documentation, CIB86 Advancing Building Technology*, Vol 2 (September 1986), Washington, D.C., pp 683-691.

De La Garza, J. M., and C. W. Ibbs, Jr., "Issues in Construction Scheduling Knowledge Representation," *Proceedings of the CIB W-65 Symposium--Organisation and Management of Construction, London, U.K.* (September 1987).

De La Garza, J. M., C. W. Ibbs, Jr., and E. W. East, "Knowledge Elicitation Techniques for Construction Scheduling," *Proceedings, Microcomputer Knowledge-Based Expert Systems in Civil Engineering, ASCE Spring Convention, Nashville, TN* (May 1988), pp 140-153.

Diekmann, J. E., and T. A. Kruppenbacher, "Claims Analysis and Computer Reasoning," *Journal of Construction Engineering and Management, ASCE*, Vol 110, No. 4 (December 1984), pp 391-408.

Ibbs, C. W., Jr., and J. M. De La Garza, "Knowledge Engineering for a Construction Scheduling Analysis System," in H. Adeli, Ed, *Expert Systems in Construction and Structural Engineering* (Chapman and Hall Publishing Co., 1987).

O'Connor, M. J., J. M. De La Garza, and C. W. Ibbs, Jr., "An Expert System for Construction Schedule Analysis," *Proceedings, First Symposium on the Expert Systems in Civil Engineering, ASCE Spring Convention, Seattle, WA* (April 1986), pp 66-77.

Schindler, P., "At Teknowledge, Knowledge is First," *Information Week*, Vol 63 (April 1986), pp 26-27.

USACERL DISTRIBUTION

Chief of Engineers
ATTN: CEIM-SL (2)
ATTN: CECC-P
ATTN: CECC-P
ATTN: CECW
ATTN: CECW-O
ATTN: CECW-P
ATTN: CECW-RR
ATTN: CEMP
ATTN: CEMP-E
ATTN: CEMP-C
ATTN: CERD-L
ATTN: CERD-C
ATTN: CERD-M
ATTN: CERM
ATTN: DAEN-ZCE
ATTN: DAEN-ZCI
ATTN: DAEN-ZCM
ATTN: DAEN-ZCZ

US Army Engineer Districts
ATTN: C/Construction Division (41)
ATTN: Area & Resident Engrs (164)

US Army Engineer Divisions
ATTN: C/Construction Division (14)

Ft. Belvoir, VA 22060
ATTN: CECC-R

USA Japan (USARJ)
ATTN: Facilities Engineer 96343

416th Engineer Command 60643
ATTN: Facilities Engineer

US Military Academy 10996
ATTN: Dept of Geography &
Computer Science

FORSCOM
FORSCOM Engineer
ATTN: Facilities Engineer (28)

NAVFAC
ATTN: Division Officer (11)

Engineering Societies Library
New York, NY 10017

Defense Technical Info. Center 22314
ATTN: DDA (2)

282
02/90