AI Magazine Volume 7 Number 5 (1986) (© AAAI)

Arvind Sathi, Thomas E. Morton, and Steven F. Roth

Callisto: An Intelligent Project Management System

Introduction

In the following two subsections, we present a brief discussion of the project management problem and how the Callisto project began.

The Project Management Problem

Innovation is important to the continued vitality of industry. New products and changes in existing products are occurring at an increasing rate, causing product lives to decrease. In order to maintain market share, companies are forced to reduce product development time and bring their products to the market as early as possible.

A major portion of development involves performing and managing many activities. For example, in high-technology industries such as the computer industry, thousands of activities must be performed to design and build the prototype of a new product. Poor performance or management of an activity can result in critical delays. If product development time is to be reduced, better management and technical support are crucial.

The Callisto project was started at the initiative of Digital Equipment Corporation (DEC) with the goal of studying and supporting the management of large projects. The focus has been on large system development *programs* (collections of several projects geared toward the design of a new computer). The following points illustrate the complexity of project management tasks in such programs:

- A large number of activities (possibly greater than 10,000) make it impossible for a manager to acquire current information about all activities.
- A number of departments are involved with different foci, attitudes, and goals.
- A program requires significant cooperation. The engineering department cannot use components that are short in supply and has to interact with the purchasing department to ascertain the supply position. The engineering department also has to interact with the manufacturing department for prototype development. Any changes made by any of these departments have an impact on the entire program.
- The developmental and technological nature of these programs makes it difficult to plan accurately Changes are frequent and need to be approved by a large number of managerial personnel.

Abstract Large engineering projects, such as the engineering development of computers, involve a large number of activities and require cooperation across a number of departments Due to technological and market uncertainties, these projects involve the management of a large number of changes The Callisto[1] project was born out of the realization that the classical approaches to project management do not provide sufficient functionality to manage large engineering projects Callisto was initiated as a research effort to explore project scheduling, control and configuration problems during the engineering prototype development of large computer systems and to devise intelligent project management tools that facilitate the documentation of project management expertise and its reuse from one project to another In the first phase of the project, rule-based prototypes were used to build quick prototypes of project management expertise and the project management knowledge required to support expert project managers In the second phase, the understanding of point solutions was used to capture the underlying models of project management in distributed project negotiations and comparative analysis This article provides an overview of the problems, experiments, and the resulting models of project knowledge and constraint-directed negotiation

Arvind Sathi is currently working for Carnegie Group, Inc , 650 Commerce Court, 6th Floor, Pittsburgh, Pennsylvania 15219-1119, as the director of production management systems Thomas E Morton is a professor of industrial administration at the Graduate School of Industrial Administration, Carnegie-Mellon University (CMU), Pittsburgh, Pennsylvania 15213, and Steven F Roth is a research associate at the Intelligent Systems Laboratory, The Robotics Institute, CMU

[1]Callisto, only slightly smaller than Ganymede, has the lowest density of all the Galilean satellites, implying that it has large amounts of water in its bulk composition Its surface is darker than the other Galilean satellites, although it is still twice as bright as our Moon Callisto is the most heavily cratered body in the Solar System and, therefore, has the oldest surface of the Galilean satellites, probably dating back to the period of heavy meteoritic bombardment that ended about four billion years ago (NASA 1979)

Related project management tasks can be decomposed into three areas: (1) activity management, (2) product-configuration management, and (3) resource management. Each of these areas can, in turn, be further delineated. **Activity management** involves four elements: (1) *planning*, which involves definition of activities, specification of precedence, resource requirements, durations, due dates, milestones, and responsibilities; (2) *scheduling*, which is the selection of activities to be performed (if more than one way exists) and the assignment of actual times and resources; (3) *chronicling*, which is the monitoring of project performance, detection of deviations from the schedule, and analysis of deviations for changes to plan (possibly resulting in renewed planning and scheduling); and (4) *analysis*, which is the evaluation of plans, schedules, and chronicled activities for normal reporting and extraordinary situations and involves the study of durations, budgets, and risk projections.

Product configuration management involves two elements: (1) *product management*, which is the management of various versions and variations of the product being designed, and (2) *change management*, which is the management of change proposals and impact evaluations, assignment of personnel for making changes, and installation of product versions.

Finally, **resource management** involves three elements: (1) the projection and acquisition of resources for project needs; (2) the assignment of responsibilities to ensure proper utilization of resources; and (3) the storage, maintenance, and repair of critical resources to minimize bottlenecks.

Deficiencies in past approaches can be attributed to inadequate modeling techniques, poor scheduling algorithms, and limited analytical tools. Our first and foremost research effort concentrated on modeling how good managers deal with the size, complexity, and changes in large projects and how they foster cooperation given the organizational diversity and loose coupling. The first leg dealt with using rule-based models to build quick prototypes of project expertise. This understanding of point solutions was then used to capture the underlying models of project expertise in the areas of project negotiations and computer-generated explanation of change using comparative analysis. This article describes our exploration into project management needs and the evolution of the resulting models of project management.

The Callisto Project

An initial investigation was encouraged by the vice-president of engineering at Digital Equipment Corporation during fall 1981. It was observed that in many ways the problems encountered in managing large development projects were similar to those associated with managing job shop activities, which was the focus of the Intelligent Scheduling and Information System (ISIS) project at CMU. The focus of the initial investigation was to determine the feasibility of developing an expert system to aid in the management of large system development projects.

It was concluded that a significant improvement could be realized in project scheduling, monitoring, and control through the inclusion of resources and other project management constraints in the project-scheduling algorithms. It was also expected that a knowledge-based project management tool would facilitate the documentation of project management expertise and its reuse from one project to another. The engineering prototype development for large systems was selected as the representative application. No tools existed to monitor and control these projects. Their nature—engineering oriented, volatile, ill structured—was ideal for a test case.

Research goals were established in the following four areas for the Callisto project: In the area of activity modeling, the goal was to generate a model of the activities and the constraints related to these activities. It was hoped that the model would facilitate the manager's ability to create activities and identify problems at creation time. In the area of configuration management, the goals were to generate a hierarchical product representation for various versions and prototypes and a way of representing the changes of these products and to develop a system to support the management of change. In the area of activity scheduling, the goal was to schedule with various hard and soft constraints and goals, which involve dynamic rescheduling and what-if simulation during project monitoring and heuristics to guard against "bad" schedules. Finally, in the area of project control, the goal was to study and model the status updating and activity-tracking procedures and the use of managerial heuristics for reporting, focusing, and diagnosing problems.

A number of factors make engineering prototype development a difficult domain for experimentation with intelligent project management systems. First, building rule-based prototypes for large and dynamic environments is a resource-intensive activity and cannot be justified on its own. Second, such projects involve a mix of economic, engineering, and manufacturing considerations. It is not possible to appreciate the problem-solving process without an understanding of all these areas. Finally, such projects involve a large number of important, yet specialized project management problems. It is difficult to understand all of these problems, isolate important ones, or develop systems that solve everyone's problems. Figure 1 traces the phases of the Callisto project.

The first phase of the project lasted about two years. Its purpose was to develop the first Callisto prototype, which consisted of a model of project knowledge and a rule-based prototype of project expertise to formulate hard-wired solutions to specific problems using production rules. Although the prototype was found to be operationally usable for configuration tracking, a useful subset of the program management problem, its usability was restricted. The information

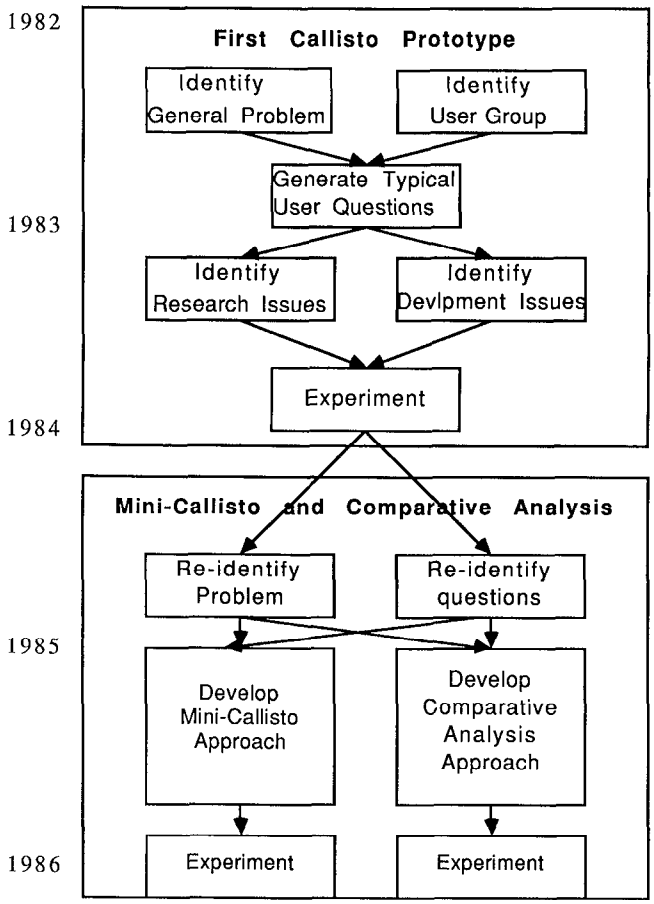


Figure 1 Callisto Development Path.

was to be funneled through a group whose responsibility was to maintain the project model. The existence of a committee inevitably causes a reduction in the information recorded, delays in the incorporation of information in the model, and the cleansing of information (for example, project reports might be too optimistic). Another problem was the level of analysis provided by Callisto. Although the model used a set of hard-wired procedures for pattern matching of typical comparisons, it could not intelligently configure the procedures together to decipher real problems.

A toy example project that engineered a fictitious computer named Micro-84 was fabricated for demonstration purposes. All the examples and scenarios in this article refer to this fictitious computer. The following test cases were used for the experiments: the developmental plans (40 activities) and configuration of Micro 84 (9 parts), the project network for Callisto itself (80 activities), a portion of the activity network for an ongoing system development project (125 activities), a random set of activity networks (776 networks with a range of 10 to 200 activities), and the system configuration and changes for a system under development (17 layers of hierarchy with 5000 parts).

The development of the configuration-tracking system (currently under way) uses as a test case a configuration with seven layers of hierarchy and about 10,000 schemata (Lynch, Marshall, and O'Connor 1986). The earlier versions of Callisto used the schema representation language (SRL) (Wright and Fox 1983; Wright, Fox, and Adam 1984), and the current work is being done using Knowledge Craft™ (Carnegie Group 1986).

Two models of project expertise emerged from the experiments. The first model attempts to capture the expertise used by good project managers in developing cooperation among project participants. In distributed project management situations, project participants often carry divergent and possibly conflicting goals and constraints. The plan specification and revision involves considerable negotiation around the constraints in order to formulate contracts to ensure cooperation. The constraint-directed negotiation model captures the expertise used by project managers in specification and revision of plans that satisfy individual constraints and foster cooperation on project goals (see Mini-Callisto for a description of the theory). These negotiations occur iteratively during plan generation, scheduling, monitoring, and repair. This model resulted in explorations with Mini-Callisto[2], a distributed problem-solving approach. Constraint-directed negotiations are examined in this article by exploring a number of negotiation situations.

A second model captures the expertise in comparative analysis of project knowledge This analysis includes understanding the quantitative, qualitative, and causal relations among activities, people, and resources (for example, the impact of a delay in resource procurement on the risk of meeting a follow-on milestone); how these properties change; how they can be classified, aggregated, abstracted (for example, a common project member responsible for all the delayed activities); and how the result of analysis can be explained using verbal and pictorial means. The purpose of this model is to support automated explanation by providing search and comparison, computation, significance testing, and verbal descriptions of change in project models. The work in this area is still in the formative stage and is the primary focus of current Callisto research.

Section 2 provides a summary of past approaches. Section 3 describes the first Callisto prototype. The experiments and related observations are described in section 4. Section 5 describes the distributed problem-solving architecture and its application to resource, activity, and configuration management. Section 6 summarizes the ongoing experiments with the distributed problem-solving architecture. Section 7 describes the current work on comparative analysis. Finally, section 8 summarizes the achievements and unexplored areas for research.

[2]The word Mini-Callisto was coined to signify that the new Callisto system locally contained a "mini" knowledge base and "mini" problem-solving capabilities which were owned by a suborganization or project member

Past Approaches

The origin of computer-based network analysis for project management dates back to 1959 when two separate but essentially similar procedures were developed: Program Evaluation and Review Technique (PERT) (Malcolm, Rosenboom, and Clark 1959) and Critical Path Method (CPM) (Kelley and Walker 1959; Kelley 1961). PERT involved the use of three separate time estimates for each activity and statistical procedures to produce probability estimates of project completion. CPM used a one-time estimate. Today, both terms are interchangeably used to refer to the common approach of (1) representing the project in the form of a network diagram and (2) performing the necessary calculations on the diagram to determine the "critical path" and start and finish times for each activity.

PERT/CPM was limited to precedence constraints and single projects A number of researchers in management science were drawn toward the project-scheduling problem. Notable among them are Turban (1976), Pritsker et al (1966), Crowston (1970), Weist (1967), Lambourn (1963), Davis (1973), and Talbot (1982). Their main agenda for research was the inclusion of resource considerations in the scheduling of project activities. Work has also been conducted in the project measurement area, where the emphasis has been on measuring, forecasting, and reporting project information, for example, cost (DeCoster 1964; Saitow 1969) For a detailed review of project management techniques, refer to Davis (1973, 1976) and Elmaghraby (1977). Despite their versatility, most of these techniques have gained little popularity. In a study of research and development (R&D) projects, Liberatore and Titus (1983) found that managers used very few sophisticated techniques to manage their projects. Gantt charts and project network diagrams were the only notable exceptions. Clearly, real-world project management problems were either different or too complex.

The human planning process has often been scrutinized by researchers in artificial intelligence (AI); their findings are applicable to project management problems. There are three major streams of research efforts that apply to project management: (1) plan representation; (2) plan generation and scheduling; and (3) plan measurement, diagnostics, and explanation

Research in plan representation explores the semantics of various concepts associated with human planning, such as time (Smith 1983; Allen 1984; Allen and Hayes 1985), process or activity (Hayes 1979; Georgeff, Lansky, and Bessiere 1985; Sathi, Fox, and Greenberg 1985), causality (Rieger and Grinberg 1977), and possession (Fox 1983). The research in this area has led to the development of semantic models of projects that can be used for intelligent reasoning and problem solving.

Research in plan generation and scheduling uses the knowledge about activities and goals to generate a sequence of steps for a plan (Tate 1977; Sacerdoti 1974; Fox 1983). A number of planning techniques have evolved, such as hierarchical planning (Sacerdoti 1974), least commitment (wait and see) approaches (Sacerdoti 1977), script-based planning (Stefik 1981; Wilensky 1983), blackboard architecture (Hayes-Roth 1985), constraint-directed search (Fox 1983), and distributed planning (Corkill 1983).

Research in plan measurement, diagnostics, and explanation interprets the project progress and diagnoses the delays to find problem areas. Many types of research touch this area including model explanation (Kosy and Wise 1984; Wise and Kosy 1985; Weiner 1980), plan recognition (Schmidt 1978), reactive scheduling (Fox and Smith 1984), vehicle monitoring (Lesser and Corkill 1983), speech interpretation (Erman et al. 1980), and simulation analysis (Reddy 1985).

Hierarchical descriptions of products are common to computer-aided design (CAD) (Freeman and Newell 1971; Latombe 1976, Preiss 1976, Stallman and Sussman 1977; Barbuceanu 1984) and software management systems (Tichy 1980) and draw upon the hierarchical modeling of objects (Winston 1975; Brachman 1979; Hendrix 1979). Refinement and change processes, however, are found less frequently (Tichy 1980; Zdonik 1984).

Tichy designed a software development and maintenance environment with three aspects: representation, interface control, and version control. His model supports multiple versions and configurations. A module family can have three kinds of members: parallel versions, revisions or sequential versions, and derived versions. A system family includes compositions or configurations and derived compositions (Tichy 1980).

The distributed constraint-directed negotiation approach is based on work in three research areas: (1) economic literature, (2) organizational behavior literature, and (3) distributed AI literature. Economic literature contains modeled agents in n-player game situations. Each player makes a choice whose outcomes (gains) are dependent upon the actions taken by the other agents, and the joint benefits depend upon the level of cooperation (Nash 1950; Luce and Raiffa 1957). The extensions to Nash's model include syndicate theory (Wilson 1968; Demski and Swieringa 1974; Demski 1976), team theory (Marschak and Radner 1972), the demand revelation model (Loeb 1975; Groves 1975; Groves and Loeb 1979), and agency theory (Fama 1980; Harris and Townsend 1981, Baiman 1982).

The work in agency theory deals with a principal and an agent. The principal forms a contract with the agent. Any returns from this contract are shared so as to maximize the returns to the principal, while subject to the constraints imposed by the agent. Agency theory model has been used by economists to study the optimal contract formulation (which would maximize cooperation between the principal and the agent subject to self-interests), admissible action rules for

the agents, and the information asymmetry between principal and agent (Baiman 1982).

Organizational behavior literature provides studies in human organizations on the human negotiation process (Pruitt 1981) and on the formation of matrix management in project organization (Galbraith 1973). Although this research is closely linked with project management, we have not yet encountered its impact on project management techniques.

In distributed AI literature, the distributed problem-solving approach conceptualizes a network of intelligent agents or actors (Greif and Hewitt 1975) capable of generating and executing plans and negotiating with other agents (Davis and Smith 1981). These problem-solving agents can be organized in an organizational hierarchy (Fox 1979; Fox 1981a; Fox 1981b) and can dynamically refine their roles (Durfee, Lesser, and Corkill 1985). This problem-solving approach decentralizes the problem solving with a limited communication sufficient for functionally accurate cooperation (Lesser and Corkill 1981) and makes solution generation feasible for large problems (Fox 1979). The approach also provides models of the contract formation process (Smith 1978) and organization designs for optimal flexibility and efficiency (Malone and Smith 1984). It theorizes distributed problem solvers with different beliefs (Fagin and Halpern 1985) negotiating on contracts (Smith 1980) and proposes a calculus of resource ownership (Lee 1980; McCarty and Sridharan 1981).

First Callisto Prototype

This section describes the various components of the first Callisto prototype which was developed to experiment with the emerging semantic model of project management.

Introduction

Consider the following scenario: *The engineering development activity for a central processing unit (CPU) typically involves the development of specifications, design on a CAD tool, and verification of the board on test cases. A committee of hardware engineers develops the specifications and assigns an engineer to design and verify the board specifications. Hence, specification is followed by design and verification. If verification is successful, the CPU is released for prototype development. Otherwise, the bug is located, the board is revised, and the design is performed again.*

Mr. Jones, a project manager in the engineering department, has been assigned the responsibility of designing the Micro-84 CPU board. Because it is not possible to cover all design aspects together, two milestones have been set for developing versions 1 and 2 of the board, respectively, and it is expected that the second version of the board will conform to project goals.

The expected duration of the design activity depends heavily on whether a new technology is used for the design. Because the decision on whether to go with the new technology has not yet been made, two schedules need to be developed, one with the assumption that the design durations will be reduced using the new technology and the other without the new technology.

Although this scenario sounds simplistic from a project management viewpoint, it raises a number of project management system engineering issues that are nontrivial. For an intelligent system that supports the management of such projects, we need to identify the critical components of the project management expertise and project knowledge representation. We need to define how this expertise and knowledge is acquired, maintained, and extended from one domain to another and how it supports the project managers.

The project representation should be complete. That is, the project knowledge should span the application domain and include all the relevant project elements used by expert project managers. For example, it should include activities (such as CPU specification), the durations of the activities; logical and temporal precedence; aggregation and abstraction (for example, how engineering development of the CPU is linked to the three activities of specification, design, and verification); individuation of schedules for the two versions of the board; representation of the two alternate schedules, one with and the other without the new technology; representation of Micro-84 and its component hierarchy, versions, and variations; representation of changes in the product; changes in the start or end dates; and resources required for each of these activities (for example, engineers, CAD tools, simulation software, and test examples). For each resource, one needs to define their availability, capabilities, and ownership. Finally, the project representation should include the representation of constraints that restrict the usage of the resources, for example, the maintenance schedule and previous reservations by other users on the CAD machine and the use of engineers for the next project.

An arbitrary set of data structures can not be used to capture this knowledge, especially if completeness implies extensibility to include new concepts. The knowledge architecture should have clarity. That is, one and only one representation exists for a given situation. For example, if a new situation involves a new type of resource, say suppliers, there should be a semantic rationale for how this new element is represented in the project knowledge base.

In addition, the knowledge representation should be precise. That is, the project descriptions should be at the appropriate granularity of knowledge. For example, depending upon the type of retrieval, the system should be able to either state that CPU verification is the next activity of CPU design or to specify all the conditions under which one activity can follow another.

The architecture for the first Callisto prototype was comprised of two major components: the knowledge architecture and the interface architecture.

Knowledge Architecture The project knowledge is organized into layers of representation. For details of the layers and their rationale, refer to Sathi, Fox, and Greenberg (1985). The *domain layer* provides concepts, words, and expressions specific to a domain of application. The *semantic layer* is composed of models of the common primitives, such as the concepts of time, activity, state, possession, agent, ownership, and so on. These concepts are common across domains and can, therefore, be used as building blocks for modeling the domain-specific concepts. The *epistemological layer* provides a way of regulating the flow of information through inheritance. It includes the concepts of set, prototype, and individuals as well as the structural relationships such as classification and aggregation. The *logical layer* defines the blocks or chunks of knowledge, such as concepts, assertions, and relations. Finally, the *implementation layer* provides primitives for machine interpretation of knowledge, such as schema, slot, relation, value, metaschema, and so on. Their specification depends on the knowledge engineering tool used.

Interface Architecture Callisto was interfaced as a single-writer, multiple-reader system with user-directed commands for activity management, configuration management, and resource management (inventory only) and supported a common (centralized) knowledge base. Through a hierarchical menu, it provided the user with the capability of interactively generating plans, scheduling the project in a simulated world to analyze the project progress under several what-if scenarios (one or more of these schedules could be stored and compared to actual progress), posting project progress, and reviewing project progress. Product configurations could be developed or changed. The user could post inventory transactions or seek status reports. Various expert critics could be activated to analyze plans, schedules, inventory status, and configuration changes. Some functional details of this system are described in the following subsections.

Resource Management

Resource management is concerned with the specification and allocation of resources to support activities. Resources in this context include personnel, work centers, tools, parts, and so on. Semantic and domain layers include the following concepts that relate to resource representation: At the domain layer exist calendars and shifts of work, stocks, vendors, stockrooms, kits, work centers, supervisors, and managers with responsibility for various resources. At the semantic layer exist resources, the time line, temporal relations, possession of resources, agents, objects and their transactions from one agent to another, aggregation of objects, resources, and associated inheritance of ownership and status.

The project-scheduling system included considerations of resource availabilities and capacities. Our initial resource-management system had three components: an inventory management component tracking resource consumption; a resource adjudication component dealing with decision making in resource allocation; and a resource critic documenting managerial heuristics for isolating problems in the utilization of resources.

The inventory management component was a discrete event-based inventory transaction system that was developed to support the activity scheduling and chronicling tasks. For example, "arrival event" increases the quantity of a given part. Events were defined for the loading and unloading of resources and for changes in the inventory. All machines and personnel were treated as resources that were possessed for the execution of activities.

The second component, resource adjudication, was an automated manager that could operate under either of three modes: the mail notification mode, the interactive mode, and the heuristic mode. In the mail notification mode, concerned responsibility centers were informed of conflicts by electronic mail, which were, in turn, resolved manually. In interactive mode, the user was given the conflicts on the screen and resolved them interactively by initiating the important activities. In heuristic mode, conflicts were resolved using a set of predefined managerial rules A large number (56) of scheduling heuristics were collected from the management science literature, and experiments were conducted to determine their comparative performance (Lawrence 1984).

The third component, the resource critic, was a rule base constructed by acquiring and encoding a number of managerial heuristics related to resources and suppliers. These rules criticized schedules and monitored performance. The rule base was assembled for concept demonstration. No experiments were performed to measure the adequacy or the impact of the criticisms.

Activity Management

Activity management deals with the generation, scheduling, and chronicling of project activities. The three phases are considered distinctly. The knowledge architecture supports these three phases using specific project knowledge at the domain and semantic levels. At the domain level exist knowledge of project activities, associated durations, risks, milestones, average (default) durations, aggregate activities for specification, design, verification, project members, mailing addresses, and responsibilities. At the semantic level exist representation of activities, states, the time line, causal and temporal relations, possession of resources, and agents and their relation to activities (see figure 2) (Sathi, Fox, and Greenberg 1985).

The three phases activity management is comprised of, are: (1)plan generation, (2)scheduling, and (3) chronicling. In the plan generation phase, an activity editor was developed to create and edit hierarchical activity networks. Interaction with the editor was through an English-like interface based on dynamic parser (DYPAR) (Carbonell et al. 1983).

Concept	Definition	Illustration
State	Fact which holds as of some point in time	*Cpu-specification is complete* *Possess CAD tool during cpu-design*
Activity	Basic unit of action Transforms states	*Specification of cpu*
Aggregation	Combine parts to make a whole	*Cpu-engg-network has three activities, spec, design and verification*
Abstraction	Process of reducing specific information	*Cpu-engg abstracts cpu-engg-network as a single activity*
Instances	Development of individual from universal	*Micro-84-design is an instance of cpu-design activity*
Manifestation	State specific description of individual	*M-cpu-design-1 specifies the schedule for Micro-84-design*
Temporal Relations	Relations to describe relative time	*Cpu-design is after cpu-specification*
Causality	Specifies an order of occurance	*Start-cpu-design (an aggregate state) enables cpu-design*
Relational Abstraction	Abstract relations to summarize complex relationships	cpu-design is the next activity-of cpu-verification (abstract) It occurs when verification fails (detail)

Figure 2. Activity Representation Definitions.

A rule-based activity critic was used to criticize the plans generated by the manager using structural (for example, missing precedence constraints) and heuristic (for example, duration estimates) project rules.

We perceived a major difference between the project scheduling of large engineering projects and the job shop scheduling being attempted in ISIS. Such projects involve a large degree of uncertainty and many changes. The accuracy or optimality of scheduling in such an environment is not as important as the development of a rough schedule that can be used for assessing and managing risk. In addition, the plan is typically too big to be developed or scheduled by one individual. Various organizational techniques, such as mutual agreements, internal pricing, and slacks, are used to distribute the scheduling problem and to solve it independently at multiple responsibility centers. A typical schedule might consist of a detailed three-month plan. Every week, or as often as needed, the project managers create a revised plan, implementing only its first week. This type of a procedure is widely used in industry and is termed a *rolling horizon procedure*; it involves scheduling far ahead of time to be sure what to do this week.

There are basically two ways of scheduling: forward or "dispatch" and backward or "reservation". The *dispatch approach* basically simulates the activities working forward in time. At each time point in the simulation, activities whose preconditions have been met are considered for scheduling. When two or more activities require the same resource, priority rules are applied to resolve the conflicts. The strength of the dispatch method is that it gives good control over the schedule in the near future. Compact schedules that make full use of resources are produced. The disadvantage is that there is less control over what happens in the distant future, for example, meeting due dates.

In contrast, the *reservation approach* works backward from the due date, reserving starting and ending times for each activity. First, a reservation is made for the last subactivity of the most important activity, then the next to the last, and so forth. Then the second most important activity is scheduled from its due date, and so forth. The strength of the reservation method is that the distant future is well controlled. Due dates for the most important projects are considered first. The disadvantage is that it provides poor control of scheduling in the near future. Gaps are often left in the schedule, and all activities of a project might not be reservable. These problems must be resolved in a second pass working forward. The reservation approach is typically useful in situations where the environment is stable, and there is a need to push work close to the deadline (for example, to reduce the cost of work-in-process inventory).

Given the level of risk and unforeseen changes, we decided to use a version of the dispatch approach. We simulated forward in time and forecast which projects create the most difficulties. These forecasts were then used to correct the current priorities so that the project with the maximum difficulties was given current priority over other projects.

Now, our principal problem was that slacks and lead times are unknown but must be included in forecasting priority. We recognized four distinct methods: (1) estimate the slacks using simple PERT/CPM; (2) use a historically estimated lead time for each resource to augment the duration of an activity; (3) repeatedly use the actual lead-time results instead of the historical estimates, and input these actual results for a second run, repeating the process until convergence is obtained; and (4) use regression to estimate the lead time for each resource from a set of critical factors, which might include shop dynamic load factor, load composition, and so on.

For multiple projects and multiple resource constraints, the algorithm can be extended by weighting the lead time on each resource by its price. The proper way to calculate prices is an interesting and complex subject. In general, one would have to solve a simpler, aggregate version of the problem with a method that would produce dual prices. These prices could be used in the detailed scheduling procedure. We found the computation of dual prices to be a cumbersome and unstable process and used the following approximation (which is similar to the way expert project managers would schedule): If a resource is, on an average, less than fully utilized, the price is 0. If a large number of activities demand the resource in the near future, the price is the ratio of resource demand to the supply available. For example, if given two engineers, one is required for two full-time activities and the other for two half-time activities, their respective prices

are 2.0 and 1.0. Expensive resources are avoided during resource conflict resolution, resulting in less bottlenecking of those higher in demand.

Project chronicling is comprised of three components: progress recording, reporting, and repairing. The first Callisto prototype was restricted to the first two. It provided capabilities of propagating the progress along the activity hierarchy and for generating rate charts, which combines progress from different levels to compute scheduled or actual performance. The structured reports and query system are used for passive reporting, and monitoring rules are used for proactive reporting, such as reporting all the pending activities at least once a week.

Configuration Management

Product configuration management involves maintenance of the configuration status, impact analysis, and installation of changes. The primary emphasis in the first Callisto prototype was on representing the configuration and change knowledge (including their relationship to the resources and the activities).

At the domain layer exist components, versions, variations, basic parts, systems and prototypes, a configuration editor, configuration reports, and the change management system. Micro-84 is composed of hardware and software. The hardware contains two circuit boards: board-h1, which consists of the CPU and the I/O, and board-h2, which contains the memory. There are two versions of board-h1. The first version is composed of a workable CPU with all the proper interfaces; the second version is a speed enhancement of version 1. Variations consist of different specifications for power supply in the U.S. and European models, requiring 60 Hz and 50 Hz power supply, respectively. Basic parts represents generic parts. Each part is defined conceptually; the parts are not connected hierarchically, for example, Micro-84 hardware. Finally, systems and prototypes represent actual physical products as instances of versions. For example, prototype 1 is an instance of Micro-84 version 1.

At the semantic layer exist the physical description, the behavioral description, and the links to activities. The physical description is derived from the work by Hayes (1979): Objects have a number of physical properties, such as mass, volume, momentum, and so on. In addition to the physical description, objects carry a description of how they behave under given conditions. The behavioral description is also the functional view of the object description.

Links to activities refers to objects in the project world that are produced, consumed, and transformed by project activities. Thus, the Micro-84 CPU behavior is defined through specifications, although its structure is defined by the design activity, and is released to the rest of the project organization by the verification activity. The specific activities, such as *revise part definitions,* are called *change orders.*

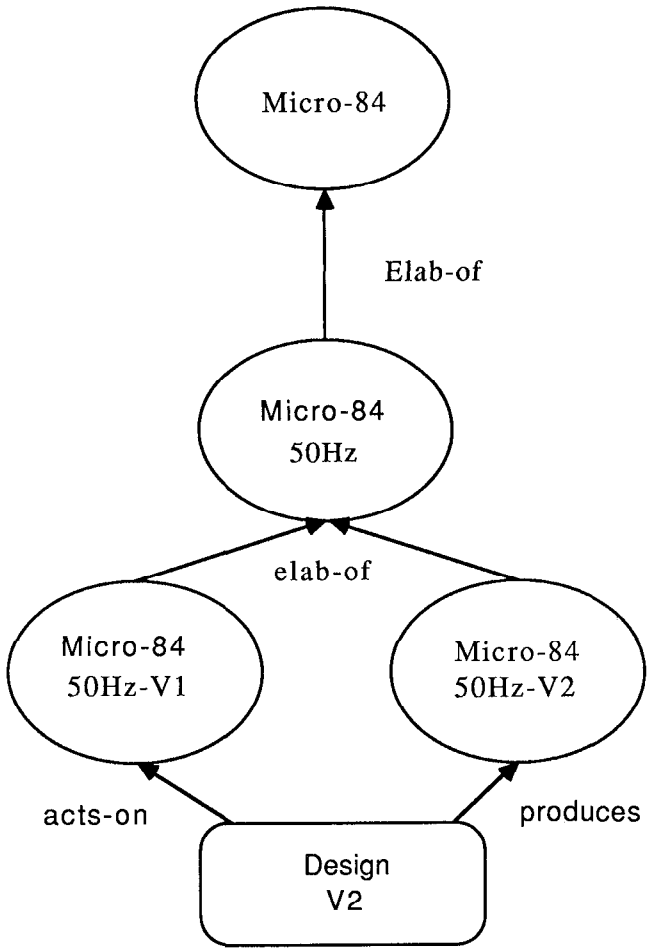


Figure 3. The Micro-84 and Changes

The configuration description in Callisto is hierarchical (that is, defined at multiple levels of detail). At its highest level of abstraction, Micro-84 is composed of hardware, software, and peripherals. At a detailed level, the hardware can be expanded into board-h1 and board-h2, and so on. Figure 3 is an illustration of the product representation. It shows the relationship between the two versions of Micro-84 and how one is generated from the other. The scope of the activity that acts on Micro-84-50Hz-v1 to produce Micro-84-50Hz-v2 can be reduced or increased by redefining the two versions.

The configuration editor is a semantic editor for specification and revision of product configuration. It can be used for adding or deleting basic parts, revisions, variations, and engineering change orders. The configuration reports can be used for reporting the configuration status or for comparatively analyzing at arbitrary levels of detail. Comparative analysis reports the changes from one version or revision to another and the associated causes for changes.

Finally the change management system can be used for entering a change. The changes can be grouped and used for creating a new version of the configuration. At any level, the

system interactively chooses three optional procedures for installing a change: destructive change (the new option supercedes the old), disjunctive (both new and old coexist) or release version (a new version is created for the part, and the procedure is repeated at the next higher level). A rule base is then used to analyze the impact of the change on the existing and scheduled prototypes.

Conclusion

The primary objective of the first Callisto prototype was to explore the knowledge architecture and the rule base prototypes of project management expertise. Although the knowledge architecture addressed some of the issues listed at the beginning of this section, the rule base prototype was inadequate for documenting or utilizing project management expertise. Although the rules matched the point solutions to the problems, they could not be extended to other problems. The project-scheduling approach required tolerance to incompleteness of project knowledge, appreciation for distribution of the plans and resources to various project managers, and the use of change orders in schedule revisions. The next section describes in detail the experiments with the first Callisto prototype and the associated observations.

Observations

The single user system was developed to the proof-of-concept level (that is, having enough functionality to demonstrate concepts but not really usable) and extended for the purpose of experimentation into the areas of activity scheduling and product-configuration management. The Callisto prototype was used in test cases for activity networks with up to 776 activities. The system was augmented with a subroutine written in C for faster number crunching. Our observations developed from the following activities:

- Interviewing key scheduling personnel: Interviews help to determine tasks and information-acquisition strategies.
- Observing review meetings to augment descriptions given by scheduling managers: The observations provided snapshots of actual project progress and problem-solving strategies used by program management personnel.
- Ascertaining commonly asked activity management questions from the project managers: These questions were the primary data points in understanding the difference between needs and available tools.
- Modeling of activities for two test cases using Callisto: These networks involved 80 and 125 activities, respectively, at multiple levels of detail. The modeling experiments were used to refine the expressiveness of the initial model.
- Interactively developing of the next set of ideas. Continuous discussions, presentations, and concept demonstrations were used to foster idea development.
- Experimenting with scheduling: The purpose of the experimentation was to study and compare project-scheduling heuristics based on knowledge used, time taken for scheduling, and the quality of schedule generated. A random set of activity networks (776 networks with a range of 10 to 200 activities) was created. Fifty-six scheduling heuristics were used individually to schedule the networks using the dispatch approach and the knowledge available for the scheduling (Lawrence 1984).

- Experimenting with product configurations: Callisto was used to model configurations and associated changes for a product under development. The system was used experimentally to assess its interface and problem-solving abilities compared to existing systems being used. The system contained two versions of the configuration, with 17 layers of hierarchy and about 5000 parts. Nearly 15,000 schemata were needed to store the above configuration.

Numerous observations were made in regard to the various aspects of system development In relation to planning, it was noted that plans evolve through negotiations and are not prespecified. For example, negotiations are often used to allocate slack time. The project support system should model and support negotiations on slack time and associated revisions in the plan rather than assume fixed durations and generate slack time as in PERT/CPM-based models.

Obvious organizational distances in communications were observed. Typically, the activities are not executed in the same department, office, or plant. The lack of face-to-face contact makes it difficult to maintain or analyze activity information in networks of the order of 10,000 activities. The critical paths generated by the PERT/CPM models lose their meaning in such situations because they do not carry or support the justifications and assumptions made during negotiations on the allocation of slack time.

Incomplete plans must be allowed for. Because networks of the order of 10,000 activities cannot be fully specified or maintained, we need tools for planning, scheduling, and monitoring that tolerate incompleteness in specification and trigger revisions when more knowledge is made available.

Project knowledge should be used for scheduling. Although a number of scheduling heuristics have been developed, they take a narrow view of constraints as applied to activity scheduling. As the knowledge is increased, the scheduling results improve; however, the time needed for scheduling increases (Lawrence 1984).

In regard to organizational ownership, it was noted that program management brings together a number of plants, divisions, and departments. The program manager seldom has overall control over the resources used by these organizations, and each of these organizations is a fairly autono-

mous unit with goals (that might not be identical to the program goals), management structure, and resources owned.

In regard to resource commitments, it was observed that these organizations commit some portion of their resources to the product development program. The commitments might have to coexist with other commitments made elsewhere. Very often, the commitments might be relaxed or renegotiated due to unforeseen changes, such as breakdowns, emergency needs, or changes in organizational structures.

Product descriptions need to be diverse. Various project personnel need different descriptions of the product. The perspectives of the supply department person counting the number of chips needed and the CAD simulation expert looking into behavioral modeling are very different. It should be possible for each one of them to maintain local definitions that are not necessarily tightly coupled with others' definitions.

Change needs to be managed. A large number of changes are made to the product at various stages of development. These changes are circulated to the other engineers, the field service staff supporting the product at Beta test sites, the manufacturing personnel building the prototypes, the supply department involved in the purchase of parts, and so on. One needs to identify whether a given change, such as the formation of the two subversions, needs to be communicated outside the negotiating entities and, if so, to whom and when. For example, for a change introduced by the design engineers, the supply department might need to be informed that some of the integrated circuits would be needed earlier than originally specified.

Product configurations are generated for diverse needs. As the needs change, they result in new definitions. The project management system should track the negotiation on definitions and the resulting product configuration.

Mini-Callisto

In this section, we describe the Mini-Callisto approach and illustrate it with resource activity and configuration management applications.

Problem-Solving Architecture

Consider the following scenario: *A new integrated circuits technology was introduced in the engineering design of Micro-84, a new supermicro. While the CAD engineers started designing the CPU using the new technology, the materials department was asked to procure the chips. The materials department informed the program manager that they could not come up with a definite plan unless they knew which chips would be included in the bill of materials. When the program manager asked the CAD group about the exact specifications of the chips, he was informed that they would be ready in about one year when the design was fully finalized. A detailed negotiation mediated by the program manager resulted in a revised plan with a predesign activity: The CAD team would develop rough specifications using high-level designs, and the materials department would purchase the chips using the rough specifications. From the first communication, the entire negotiation took about five months. Project management tools were then used to specify the resulting plan.*

In situations such as this, organizations or individuals initiate negotiations when faced with inconsistent or incomplete project knowledge. Constraints are shared, relaxed, and strengthened among all individuals involved in the negotiations. The negotiations result in plans which satisfy everyone and which, possibly, are much more detailed than the original plans. Existing support systems help the managers in storing the plans but only after the negotiations are complete.

Many project management systems, including the first Callisto prototype, are based on a fundamental assumption. They focus their attention on a plan, which is generated, stored, scheduled, and monitored, for proper project management. Such support systems are likely to fair well in situations involving stable or small project models. In contrast, engineering program management involves a large number of changes that are initiated and cooperatively agreed upon by a large number of participating project members or organizations. The support system in such an environment requires additional emphasis on the specification and revision processes and the need for cooperation, which leads to the concept of Mini-Callisto.

Mini-Callisto is a system capable of supporting the specialized needs of an organizational unit and of communicating with other such systems in a network during specification and revision processes. This system draws upon past research in distributed problem solving. A description of the design of the Mini-Callisto architecture follows.

It is assumed that the project is distributed across a number of organizational entities, such as the program manager, the CAD group, and the materials department, with overlapping, but not necessarily common, goals and associated specialized knowledge. These organizational entities hold agreements with one another to facilitate cooperation on the project. Each organizational entity has a Mini-Callisto that provides a portion of the project management capabilities. Thus, a Mini-Callisto attached to the program manager is capable of working as a scheduling assistant with procedures for critical-path and risk analyses. Each Mini-Callisto has its own local knowledge base that reflects the beliefs of the organizational entity it supports.

These Mini-Callistos are connected together. They use messages to communicate with each other. Each message has an associated action. Messages are used for generating proposals, communicating constraints, proposing constraint relaxations, committing to plans, and querying others' knowledge. The messages are grouped into protocols that describe how a specification or change can be made.

A *constraint* expresses an impediment to plan variables. It can be interpreted in three ways: (1) an elimination rule from the perspective of object selection; (2) a partial description and commitment from the perspective of plan refinement; and (3) a communications medium for expressing interactions among organizational entities, each of which solves a subproblem. A constraint is not only a restriction but also the aggregation of a variety of knowledge used in the reasoning process. In particular, it includes the relative importance of multiple constraints; the possible relaxations and their relative utilities; the obligation to satisfy constraints according to time, context, and source; the interactions among constraints; and the dynamic generation of constraints (Stefik 1981; Fox 1983).

Negotiation is a form of decision making in which two or more parties communicate with one another in an effort to resolve nonoverlapping interests (Pruitt 1981). In the context of project management, the nonoverlapping interests are in the form of constraints faced by each party. A negotiation is initiated if and when an organizational entity faces an inconsistency or an incompleteness in the project knowledge base that can only be resolved with the help of other organizational entities. Negotiation is the process of isolating the constraints, communicating them to the other organizational entities, and jointly relaxing or strengthening them in order to resolve the inconsistency or incompleteness. A variety of negotiation operators, such as cost cutting, trading, arbitration, and mediation, are used for selecting the direction and magnitude of the strengthening or relaxing (see figure 4).

In a centralized algorithm, the constraint-directed search uses global importance for each constraint and global utility for each relaxation. In the distributed case, such global measures are extremely expensive (if not impossible) to compute. Instead, the negotiation needs to accommodate multiple agents, each with a set of constraints and their evaluations. A number of techniques are used by expert negotiators to reach an agreement (Pruitt 1981). The techniques that can be chosen to manipulate constraints in an automated negotiation situation are cost cutting; trade, substitution, and compensation, log rolling; bridging; unlinking; mediation; and arbitration.

Cost cutting involves reducing the cost of a relaxation for the other party by manipulating other parameters, such as the context. For example, an overutilization of a resource for a short duration of time can be made to look like normal utilization by extending the time horizon.

Trade, substitution, and compensation involves the exchange of project objects, such as resource reservations, to reduce the cost of relaxation for all the negotiating parties. The sharing of the losses, if any, should be according to goals shared.

Log rolling involves an exchange of concessions through selective violation of a portion of the constraints. For example two agents, each contributing three constraints to the negotiation, might decide to violate one constraint

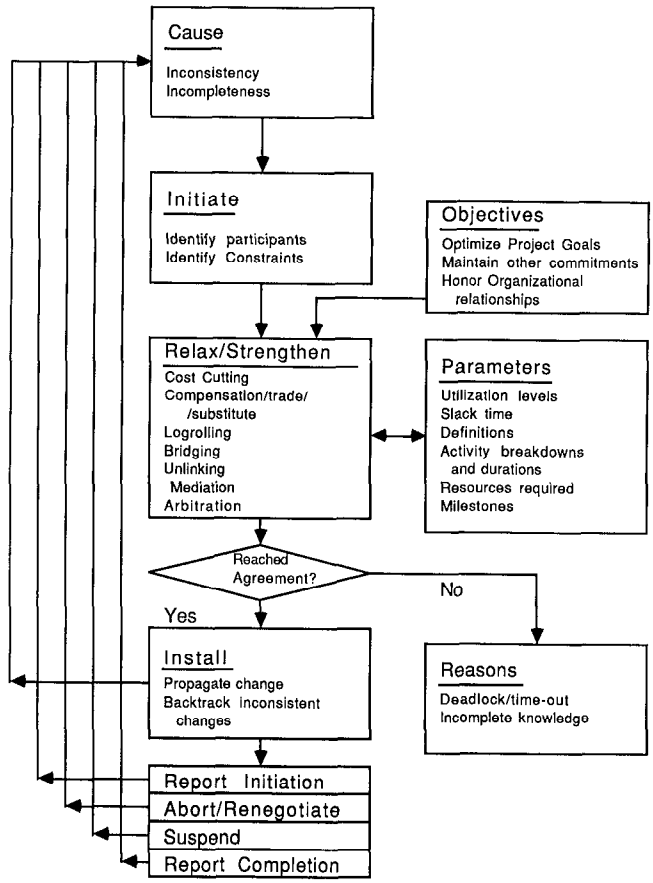


Figure 4. Constraint Directed Negotiation

each, in order to reach an agreement. Log rolling involves prioritizing constraints by their importance and selectively violating those which are low in priority. Although trading involves bargaining on relative losses, log rolling focuses on the relative importance of the competing constraints.

Bridging occurs when a new option is developed that satisfies both parties' most significant constraints. Such a relaxation is selected even though it violates all the other constraints.

Unlinking occurs when selective concessions are necessary to reach a solution and bridging or log rolling has not worked. It involves removing weak interactions among constraints for the purpose of negotiation (for example, relaxation of a due date that affects organizational stability).

Mediation occurs when the negotiating parties reach a deadlock (that is, are unable to relax any more constraints to reach an agreement). Mediation involves reassessment of the relaxations and the important constraints by a third party with, hopefully, a global perspective.

Arbitration occurs when the negotiation reaches a deadlock and even a third party can not mediate. In such cases, if the need is critical, a third party assigns losses and forces relaxations to the constraints posed by each of the negotiating parties.

Each Mini-Callisto is managed by a program module called an R-object (R stands for responsible). *R-objects* maintain the organizational structure of the project and have mechanisms for generating, filtering, and archiving messages for communicating with other modules as well as mechanisms for generating procedures for solving problems. The R-objects are comprised of ports, views, plans, and local and shared knowledge.

Ports provide the mechanism for communication across R-objects. Messages can be generated by an R-object and directed to the out port where they are preprocessed before the communication. Then the messages arrive at the in port of the receiving R-object where auxiliary communication can take place to resolve any inconsistency and incompleteness before their assimilation into the rest of the local knowledge.

Each R-object carries *views* specifying the other R-objects and their organizational relationships. The views use the organizational information to process a message for inconsistency and incompleteness. For example, in a given incomplete message, a Mini-Callisto can automatically fill in the name of the project if this is the only project common to the Mini-Callisto sending the message.

R-objects also carry *plans* associated with each protocol or message. Given a message in the in port, an R-object executes the associated plan. For example, the plan associated with a query message involves checking whether the query is understandable and whether the source has the authority to make the query and then responding to it (Kedzierski 1983).

Each Mini-Callisto carries two types of objects: those objects which are local to the Mini-Callisto and are thereby owned by the local R-object (*local knowledge*) and those objects which have been moving from one Mini-Callisto to another and are possibly owned by another R-object residing in another Mini-Callisto (*shared knowledge*). Any revisions to an object owned by another Mini-Callisto use *revision protocol*, which involves negotiation and approval before the revision is finalized.

A test bed of the Mini-Callisto network has been created using knowledge craft and its context (alternate world) mechanism (Carnegie Group 1986) in a simulated distributed environment. This network facilitates modeling of distributed knowledge and problem solving on local knowledge. The choice of simulating rather than actually using a distributed environment was made entirely for convenience (a real distributed version was also created but was found too difficult to experiment with). The test bed has provisions for creating several Mini-Callisto nodes; switching from one node to another; sending messages from one node to another; negotiating on project management problems; and gathering associated statistics on the number of messages, the processing load, and the associated changes for each negotiation. The test bed is currently being used for the experiments described under Mini-Callisto experiments. The next three subsections discuss the evolution of constraint-directed negotiation for resource, activity, and configuration management, respectively.

Resource Management

Consider the following scenario: *The printed circuit lab to be used for the design of the Micro-84 CPU belongs to the manufacturing department. The manufacturing department has agreed to the engineering department's use of 40% of the throughput. Jack, the supervisor of the printed circuit lab, schedules its use for the engineering department along with the preventive maintenance requirements and ongoing manufacturing department needs. The engineering department requested from Jack specific reservations for lab use for the next month. A round of negotiation was conducted, ending with a mediation by the plant manager, to provide the needed throughput.*

Although project-scheduling systems have been extended to include considerations of resource availability and capacity (Talbot 1982; Project/2 1981), we have not yet come across any approaches that include resource negotiations based on ownership and commitments in project scheduling. Large projects involve resource sharing between resource owners and project-activity owners (for example, sharing of the printed circuit lab between engineering and manufacturing departments). The sharing is finalized through negotiations, which involve complex agreements resulting from trading, log rolling, or arbitration.

The Mini-Callisto model explicitly brings resource ownership and commitment into the resource-allocation process. Each resource is owned by an agent. Resource sharing needs to be negotiated with the agent owning the resource. Agents are interdependent through organizational links. These organizational links are used for delegating of resource ownership from one agent to another and for adjudicating conflicts at lower levels of the organization. Contracts are formed across two or more agents for the use of a resource. The contracts specify the resource, the contracting parties, and the duration of use. No changes can be made to a contract without the approval of the contracting agents.

The steps for constraint-directed resource negotiation begin with the cause: A Mini-Callisto locates an incompleteness or an inconsistency in the project knowledge that needs to be resolved. As an example, a Mini-Callisto supporting the plan generation for the engineering team recognizes the absence of a contract for the use of the printed circuit lab in the design and verification activities. (Every resource reservation that requires a resource outside the engineering department should be in the form of a contract).

Next, it is necessary for the initiating Mini-Callisto to identify the negotiation participants (that is, those agents whose input or approval is necessary for resolving the inconsistency or incompleteness). Thus, our Mini-Callisto supporting the engineering department locates Jack as the owner of the printed circuit lab.

The third step is to identify the constraints. The agent requiring the resource shares the constraints with the agent owning the resource. The Mini-Callisto that supports our engineering department communicates to Jack's Mini-Callisto of the need to use 40% of the printed circuit lab over the next month.

Jack's Mini-Callisto then searches through the existing reservations for the lab and recognizes that the available capacity is less than 40%. It communicates the constraint to the Mini-Callisto supporting the engineering department. The Mini-Callisto supporting the engineering department responds back, informing Jack's Mini-Callisto that the reservations can not be made in any other time period (because of a due-date constraint).

Any of the negotiation operators can be used to relax the constraints. This negotiation takes into account the importance of the agents, their organizational relationship, and the past contracts. If conflicts cannot be resolved, the negotiation is passed to a higher level for mediation. If mediation at a higher level fails, the proposal is aborted or revised.

Thus, in our example, Jack's Mini-Callisto searches through the existing reservations to find that the contracts are with the manufacturing department and that Jack is unable to assess their importance. In the absence of any other local ways of relaxation, Jack's Mini-Callisto informs the engineering department that the PC lab is not available. The engineering department communicates the negotiation situation to the plant manager (request for mediation). The Mini-Callisto supporting the plant manager searches and finds a contract specifying an overall 40% throughput for the engineering department and decides to trade the reservations (by moving or bumping manufacturing reservations into the future).

Finally, if no conflicts remain, the contract is formalized. The formalization can result in auxiliary proposals for associated changes in contracts with other organizations. For example, the Mini-Callisto supporting the plant manager communicates the changes in exisitng reservations to the Mini-Callisto supporting Jack. Jack's Mini-Callisto reevaluates the constraints associated with the utilizations and recognizes that the proposed reservations can be granted to the engineering department. It informs the Mini-Callisto supporting the engineering department of the agreement to use the printed circuit lab. The Mini-Callisto supporting the engineering department responds with an acknowledgment, thereby signaling the contract formation. New negotiations are initiated within the manufacturing department to decide when the manufacturing reservations should be scheduled.

The test bed is being used for experiments with the Mini-Callisto model of resource management in various allocation specification and revision situations. In the next two subsections, we explore other types of negotiations and how they are used if and when the resource negotiations fail (or take too long).

Activity Management

Consider the following scenario: *The design engineers were falling behind in their work because of the unavailability of CAD machines. The negotiations between the CAD machine owners and the design engineers resulted in the realization that verification of the first version of the CPU was competing (in the CAD machine utilization) with the design of version 2. It was decided to trade some slack time available in the verification of version 2 to version 1, thereby pushing the design of version 2 into the future.*

Resource management dealt with the specification of resource schedules through negotiations based on ownership and commitment levels. Activity management tasks extend the negotiation to include the constraints related to the activities, such as activity criticality and available slack time. These negotiations often repeat, with the focus changing from resource to activity constraints and back again. In the scenario just described, each of the two versions of the Micro-84 CPU design carry slack time meant to be used in unforeseen situations. The contention for resources, which could not be adequately addressed through resource-commitment negotiations, was retried and resolved through negotiations on the activity slack, thereby "substituting" or "trading" some time available for version 2 to version 1.

The activity negotiation support system involves three activities. The first is the distribution of problem and activity knowledge, which assumes that the project organization is divided into a large number of groups, each possessing the capability to execute a subtask and each having the knowledge of the prototypical activity networks for the subtask. The project goals are divided accordingly and distributed to each group. These goals are translated by each group into a set of activities to meet the goals.

The second activity is a negotiation on plan specification and revision. The groups begin to share a portion of their local plans to define the durations and the performance in accordance with the project objectives or milestones. Extending activities beyond the due dates (as assigned in the milestones) implies additional costs to be incurred. The most common way to eliminate these additional costs is to trade slack time available from one activity to the activity requiring more time. Another strategy, one that is often technically infeasible, is to compensate by providing more resources (that is, activity crashing). Sometimes, it is possible to reduce the costs of delaying the activity beyond the due date. Mini-Callisto supports the sharing of plans as well as the relaxation of constraints (for example, changes to activity durations and related slack times). It provides the support by computing constraint utilities (for example, using critical-path evaluation for the local network) or by using automated interactions for simpler negotiations.

The third activity is the completion of incomplete specifications. The local plans might not be complete for the nego-

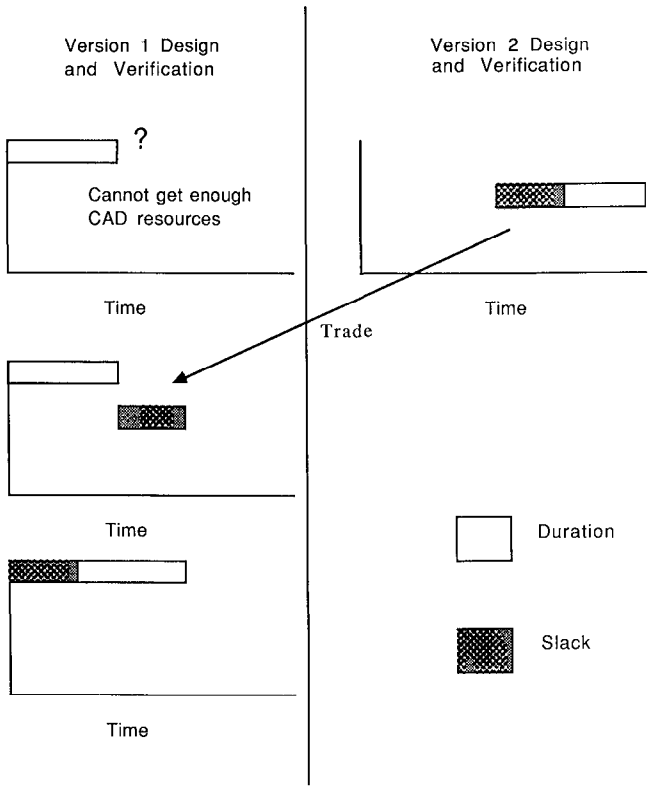


Figure 5 Activity Negotiation

tiation with others. Completion of the incomplete portions initiates new negotiations at a future time

Figure 5 illustrates the negotiation process for the CPU engineering activity. The test bed experiments with the use of activity negotiations to augment resource negotiations and to improve the quality of the final plans (that is, the level to which they satisfy all of the constraints). The experiments simulate a set of engineers negotiating on the completion of the overall project using commitments, resource substitutes, and activity slack times.

Often negotiations on resources or activities require consistent understanding of the plan or product definitions, which is especially true when the negotiations fail at the resource or activity level. The next subsection extends the negotiations to include the definitional constraints

Configuration Management

Consider the following scenario: *Each of the CPU design and verification activities for Micro-84 will take a year. The engineers were told that the engineering development needs to be completed in 1 1/2 years It became apparent to the engineers that some verification had to take place while design was still being done. Detailed negotiations on the design elements and technological precedence constraints resulted in a plan for verification of a portion of the instruction set while other portions were still being designed. The supply department was informed about the change so that the related purchases were made six months earlier.*

Such negotiations require a good understanding of the components of the CPU and their design, verification, and associated purchase activities. Any changes to the components of the CPU can affect or initiate such negotiations. Also, multiple versions of the CPU can exist, and negotiations can focus on one or more of these versions.

Definitional negotiations use product definitions and generate new definitions in order to resolve project conflicts. These changes have far-reaching consequences for previously negotiated resource allocations or activity networks. The biggest deficiency in conventional project management tools, for use in engineering program management is their exclusion of configuration and change management and their inability to propagate these changes to the rest of the project activities. For example, if a change made to the design activities is not communicated to the supply department, it nullifies the intended effect of finishing the design early. In order to model negotiations and related changes, one needs to model the diverse descriptions, their relationships, and the impact of one specification or revision on another.

The Mini-Callisto approach models product definition and change negotiations as an integral part of the negotiation process. It supports user-initiated change negotiations and identifies and initiates the associated activity management or resource-allocation negotiations. A typical scenario begins with the generation of a change requirement The design team detects the need for a change in the plan for Micro-84 CPU design A goal of reducing the engineering time by six months is established.

Change negotiation is the next step. A number of negotiations are attempted to meet the goal. Let us assume that the goal cannot be met through increased resource commitments because they are already 100%. Also, available slack time cannot be decreased any more because no slack time is left. Each of these negotiations involves sharing of knowledge, such as resource commitment, capacities, and slack time available for the entire activity network. Finally, the negotiation is turned toward product definitions. A possible relaxation is found in dividing the CPU into two parts, each complete in itself (that is, no design dependence). The cost of delaying a part of the design is thereby reduced.

Change installation is the final step. The changes lead to two subversions of Micro-84 A search among other activities related to the Micro-84 CPU reveals that the supply department needs to be informed. A subsequent negotiation is initiated with the supply department and results in changes to the project activities.

Figure 6 shows the activity and product knowledge before and after the negotiations. The part definition negotiations are the most difficult to implement and support. As can be seen from this scenario, the negotiations cannot be done unless the model includes design descriptions. A miniature design expert was developed to explore the design knowledge and its use in project negotiations (Glackemeyer 1984),

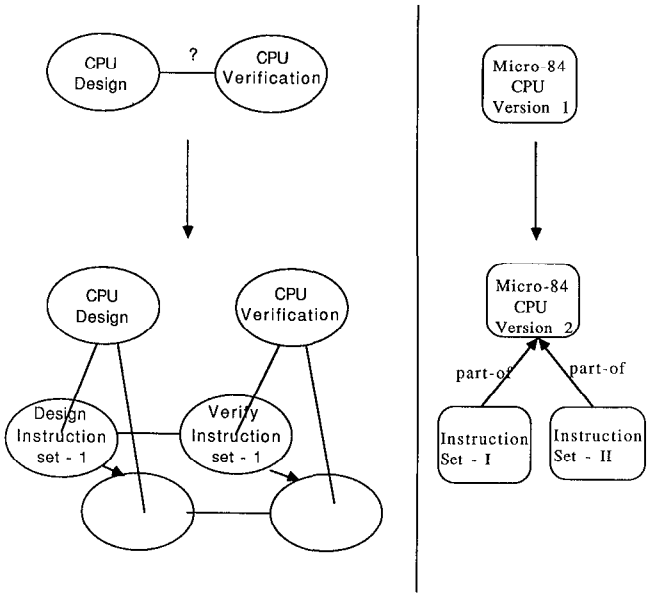


Figure 6 Product Definition Negotiations

although the related experiments are yet to be designed or conducted.

To summarize, the three components of project management—activity, resource, and configuration management—are interconnected both at the project knowledge and the negotiation levels. Negotiations begin with one type of constraint and are either resolved or continued to include relaxations of other constraints, with possible backtracking if the negotiations fail. The Mini-Callisto system can be used for modeling, supporting, or automating these negotiations.

Mini-Callisto Experiments

We need to validate the constraint-directed negotiation approach on real project situations. The Mini-Callisto experiments initiate the validation process in three ways.

First, Does it work? The constraint-directed negotiation should be able to model various project management situations within activity, resource, or configuration management. Also, the approach should converge to a solution wherever a solution exists. This article demonstrates the applicability of the approach. The experiments involve a number of random situations for each of the three types of negotiation situations. The solutions are tracked for quality (that is, the number of successful negotiations) and negotiation effort (that is, the number and size of communication messages).

Second, How good is it? A centralized project-scheduling or change installation algorithm can provide an ideal solution to a cooperative problem, that is, the performance of the systems when the goals are identical and when the information is completely available to all the project members. A series of experimental runs compare the distributed project negotiation-based solutions to the upper limit generated by the centralized solution under different distributions of goals to the project members.

Third, Does it use negotiation expertise? Choosing the right goals and constraints for negotiation is important. Choosing incorrect constraints involves additional sharing of information, expensive trade-offs, and additional time spent searching and backtracking.

The third set of experiments will probably be the most difficult and will take the longest time. At the same time, the comparisons and adequacy of the approach (as in the first two experiments) will be affected by the level of negotiation expertise, thereby requiring a reasonable level of negotiation expertise to be captured in the Mini-Callisto test bed. The details of experiments and their results will be the subject of a follow-on article.

The Mini-Callisto concept is being developed with the goal of fulfilling any of the following roles, depending on the objectives, the complexity of the situation, and the negotiaton expertise embedded in the system: a support system which documents human negotiations, such as that developed by Marca and Cashman (1985); an enhancement to existing tools, providing minimal negotiation and change management support; an expert system that automatically negotiates in simpler (yet large and dynamic) situations, especially those involving automation of the activities (for example, flexible manufacturing systems (FMS) scheduling); and a test bed for understanding and exploring project management practices. The extension to these areas will trigger a set of experiments to assess the adequacy of the Mini-Callisto approach to problem solving.

Model of Explanation Using Comparative Analysis

We observed that it is inherently difficult to interpret and analyze complex, resource-constrained activity networks (even when graphically illustrated). For example, it requires tremendous effort to identify and understand the effects of changes to some areas of a schedule on other areas. Project managers regularly study project schedules and status information to analyze progress and its effect on related activities. Very often, such analysis is done using unfamiliar tools and graphic aids that offer virtually no assistance other than fancy documentation. Such utilities usually allow users to peruse a project schedule at a single phase of completion and never support comparison of multiple schedules.

By "explanation," we mean the analysis, interpretation, clarification, and report presentation of plans, schedule information, and conclusions produced by activity management systems. The motivation for our work is based on several observations about the task of project management and our attempts to build tools to support it. First, managers must be able to maintain, access, interpret, and act upon information from large and diverse project knowledge bases. Monitoring and managing change in project plans and the status of project activities requires frequent comparisons of lengthy

schedules and networks. Managers must know what information is needed, where to locate it, and how to interpret and use it. Equally important is that they be able to do so without great effort.

Our second observation is that with advances in project management research, this task has become more difficult rather than easier. AI and operations research (OR) techniques for project management are increasing the quantity and complexity of knowledge about projects that can be represented and maintained. Decision support systems are using this knowledge to assist or automate many aspects of management decision making. As a result, advances in knowledge representation and inference-making capabilities will greatly expand the need for managers to access and interpret project knowledge and procedures and justifications behind system decision making. Although the process of monitoring the evolution of project plans was complex enough when only activity precedences and durations were represented (for example, with critical path method (CPM) approaches), this process becomes an even greater bottleneck as plans become more knowledge intensive.

Our approach to explanation extends a technique called *comparative analysis* (Kosy and Roth 1986), that has been used successfully in the explanation of change in financial modeling (Kosy and Wise 1984). Although previous work was limited to explaining change in quantitative models, our approach involves identification and interpretation at many levels of understanding, depending on the depth required by the user or the knowledge that is available to the system. These levels include (1) understanding the quantitative relations among the temporal properties of activities and resources contained in PERT networks (for example, knowing statistically how the change in risk in a milestone was produced by changes in delays of prior activities); (2) understanding the qualitative properties of activities, resources, and other project entities and the ways they can be classified, aggregated, abstracted, and summarized in order to help managers find reasons for changes beyond quantitative relations (for example, recognizing that a set of activities contributing to the increased risk for the milestone of a large project is the responsibility of a single subproject); (3) understanding the methods by which changes to a plan are made, including who makes the changes (people or software agents), the types of changes made (for example, precedence, time estimates, detailed breakdowns, or addition of new activities), and (when possible) awareness of other information that might provide the rationale for changes (for example, recognizing the existence of new precedence links and realizing their creation is the result of a Mini-Callisto in response to new commitment information); and (4) understanding other events in a project environment which explain the causes of changes (for example, the process and results of commitment negotiation among people or software agents in a project).

A series of experiments were performed to implement components of comparative analysis within Callisto (Roth et al. 1986). CPM was chosen as the first application because it involves only a single dimension (time) and a single quantifiable constraint (technological precedence) and is therefore analogous to financial spread sheets (costs and algebraic equations) (Kosy and Wise 1984). Although this restriction limits explanation to the identification of change based on quantitative relations (the first level of understanding), it serves as a starting point for explaining complex models of projects.

Explanation is one of the important future areas of research in project management We intend to expand the necessary expertise in the following ways: (1) to increase the complexity of quantitative models that can be explained (for example, resource-constrained schedules and probabilistic time estimates), (2) to expand comparative analysis to deal with qualitative changes and integrate with quantitative changes, (3) to develop explanatory capabilities for the results of distributed processes for negotiation, (4) to develop an approach to knowledge-based graphics for creating and coordinating multiple text and graphic displays to satisfy the needs of explanation, and (5) to develop a discourse model of human explanations in project management both to support and test our approach to computer-generated explanation.

Conclusions and Future Plans

The Callisto project has made a number of significant research contributions. It has successfully modeled the specification and revision process of project management as a series of constraint-directed negotiations. This model enhances the capabilities of project management tools in dealing with large, complex, and dynamic projects. It has also contributed toward the development of knowledge-based models for project management and similar planning tasks. This model has subsequently been applied to a number of diverse situations ranging from software engineering to manufacturing planning. A configuration-tracking system has been developed out of the first Callisto prototype to be used by Digital Equipment Corporation for the tracking of product configurations. The system is currently being field tested.

The successful implementation of Mini-Callisto requires a set of user interface capabilities that facilitate the use of mixed-mode negotiations. The interfaces are critical for intelligent assistants because the users need to share assumptions, defaults, and decisions with the system. The capabilities include methods for displaying and reporting information contained in various Mini-Callistos as well as tools for their specification and revision. Our efforts in the comparative analysis area need to be extended to include other interface tools required to support mixed-mode project negotiations.

Callisto provides an excellent opportunity for studying distributed problem solving and validating the constraint-directed negotiation approach. We foresee a large number of experiments to validate and extend the Callisto model in project management and similar domains. The approach can be used to solve large real-world problems. The successful applications, though, require a maturing of technologies in the areas of communications, hardware for workstations, and system software for distributed problem solving.

Acknowledgments

This research was conducted at the Intelligent Systems Laboratory at CMU and was partially supported by Digital Equipment Corporation and Carnegie Group Inc. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Digital Equipment Corporation or Carnegie Group Inc.

We would like to acknowledge Mark Fox, Mike Greenberg, and LeRoy C. Smith who contributed to earlier versions of this article. Also, we would like to acknowledge the helpful suggestions of John McDermott, William Sears, Robert Murphy, Mark Olsen, Richard Glackemeyer, Ray Ross, Pam Gage, Jean Kauffmann, and two anonymous referees.

References

Allen, J F 1984 General Theory of Action and Time *Artificial Intelligence* 23(2): 123-159

Allen, J F., and Hayes, P J 1985 A Common-Sense Theory of Time In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 528-531

Baiman, S 1982 Agency Research in Managerial Accounting: A Survey *Journal of Accounting Literature* 1:154-213

Barbuceanu, M 1984 Object-Centered Representation and Reasoning: An Application to Computer-Aided Design In SIGART Newsletter, January, 33-39

Brachman, R J 1979 On the Epistemological Status of Semantic Networks In *Associative Networks Representation and Use of Knowledge by Computers,* ed N V Findler, 3-50 New York: Academic

Carbonell, J ; Boggs, M ; and Monarch, I 1984 DYPAR User's Manual, Computer Science Dept , Carnegie-Mellon Univ

Carnegie Group 1986 Knowledge Craft User's Manual, Version 3 1, Carnegie Group

Corkill, D D 1983 A Framework for Organizational Self-Design in Distributed Problem Solving Ph D diss., Computer and Information Science Dept , Univ of Massachusetts

Crowston, W B S 1970. Decision CPM: Network Reduction and Solution *OR Quarterly* 21(4): 435-452

Davis E. W 1976 *Project Management. Techniques, Applications, and Managerial Issues* Atlanta, Ga : American Institute of Industrial Engineers, Inc

Davis, E W 1973 Project Scheduling under Resource Constraints: Historical Review and Categorization of Procedures American Institute of Industrial Engineering Transactions 5: 297-313

Davis, R , and Smith R G 1981 Negotiation as a Metaphor for Distributed Problem Solving, Technical Report, AI Memo 624, Artificial Intelligence Laboratory, Massachusetts Institute of Technology

DeCoster, D. T 1964 PERT/Cost—The Challenge *Management Services,* May-June

Demski, J S 1976 Uncertainty and Evaluation Based on Controllable Performance *Accounting Research,* Autumn: 230-245

Demski, J S , and Swieringa, R J 1974 A Cooperative Formulation of the Audit Choice Problem *Accounting Review* 49(3): 506-513

Durfee, E H ; Lesser, V R ; and Corkill, D D 1985 Increasing Coherence in a Distributed Problem Solving Network In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 1025-1030

Elmaghraby, S E 1977 *Activity Networks Project Planning and Control by Network Models* New York: Wiley

Erman, L D; Hayes-Roth, F ; Lesser V R.; and Reddy, D R. 1980. The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty *Computing Surveys* 12(2): 213-253

Fagin, R , and Halpern, J Y 1985 Belief, Awareness, and Limited Reasoning: Preliminary Report In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 491-501

Fama, E F 1980 Agency Problems and the Theory of Firm *Journal of Political Economy* 88(2):288-307

Fox, M S 1983 Constraint-Directed Search: A Case Study of Job-Shop Scheduling. Ph D diss , Computer Science Dept , Carnegie-Mellon Univ

Fox, M S 1981a The Intelligent Management System: An Overview In *Processes and Tools for Decision Support,* ed H G Sol Amsterdam, The Netherlands: North Holland

Fox, M S 1981b An Organizational View of Distributed Systems In IEEE Transactions on Systems, Man, and Cybernetics 11(1): 70-80

Fox, M S 1979 Organization Structuring: Designing Large Complex Software, Technical Report, CMU-CS-79-155, Computer Science Dept , Carnegie-Mellon Univ

Fox, M S , and Smith, S 1984 The Role of Intelligent Reactive Processing in Production Management In Proceedings of the Thirteenth Annual Computer Aided Manufacturing International Incorporated Technical Conference, 6-13—6-17

Freeman, P , and Newell, A 1971 A Model for Functional Reasoning in Design In Proceedings of the First International Joint Conference on Artificial Intelligence, 621-640

Galbraith, J 1973 *Designing Complex Organizations* Reading, Mass : Addison-Wesley

Georgeff, M P ; Lansky, A L ; and Bessiere, P 1985 A Procedural Logic In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 516-523

Glackemeyer, R 1984 Behavioral Simulation of Electronic Design, Technical Report, Intelligent Systems Laboratory, The Robotics Institute, Carnegie-Mellon Univ

Greif, I , and Hewitt, C E 1975 Actor Semantics of PLANNER-73 In Proceedings of Association of Computing Machinery, Special Interest Group in Programming Languages-Special Interest Group in Automata and Computability Theory Conference, Palo Alto, Calif : ACM

Groves, T 1975 Information, Incentives, and the Internalization of Production Externalities In *Theory and Measurement of Economic Externalities,* ed S Lin New York: Academic

Groves, T , and Loeb, M 1979 Incentives in Divisionalized Firms *Management Science* 25: 221-230

Harris, M , and Townsend, R M 1981 Resource Allocation under Asymmetric Information *Econometrica* 49(1): 33-64

Hayes, P J 1979 The Naive Physics Manifesto In *Expert Systems in the Micro Electronic Age,* ed D Michie, 243-270 Edinburgh, United Kingdom: Edinburgh

Hayes-Roth, B 1985 A Blackboard Architecture for Control *Artificial Intelligence* 26(3): 251-321

Hendrix, G G. 1979 Encoding Knowledge in Partial Networks In *Associative Networks, Representation and Use of Knowledge by Computers,* ed N V Findler New York: Academic

Kedzierski, B I 1983 Knowledge-Based Communication and

Management and Support in a System Development Environment Ph D. diss , Computer Science Dept , Univ of Southwestern Louisiana

Kelley, J E, Jr 1961 Critical-Path Planning and Scheduling: Mathematical Basis *Operations Research* 9(3): 296-320

Kelley, J E , Jr., and Walker, M R 1959 Critical-Path Planning and Scheduling In Proceedings of Eastern Joint Computer Conference, 160-173

Kosy, D W , and Roth, S F 1986 Applications of Explanation to the Analysis of Schedules and Budgets, Technical Report, Intelligent Systems Laboratory, The Robotics Institute, Carnegie-Mellon Univ

Kosy, D. W., and Wise, B P. 1984 Self-Explanatory Financial Planning Models In Proceedings of the Fourth National Conference on Artificial Intelligence, 176-181 Menlo Park, Calif : American Association of Artificial Intelligence

Lambourn, S 1963 Resource Allocation and Multi-Project Scheduling (RAMPS)—A New Tool in Planning and Control *Computer J* 6: 300-303

Latombe, J C 1976 Artificial Intelligence in Computer-Aided Design: The TROPIC System, Technical Report, Tech Note 125, Artificial Intelligence Center, Stanford Research Institute

Lawrence, S R. 1984 Resource-Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques, Technical Report, GSIA, Carnegie-Mellon Univ

Lee, R M 1980 CANDID: A Logical Calculus for Describing Financial Contracts. Ph.D diss , Dept of Decision Sciences, The Wharton School, Univ of Pennsylvania

Lesser, V. R , and Corkill, D D 1983 The Distributed Vehicle-Monitoring Testbed: A Tool for Investigating Distributed Problem-Solving Networks *AI Magazine* 4: 15-33

Lesser, V R , and Corkill, D D 1981. Functionally Accurate, Cooperative Distributed Systems In *IEEE Transactions on Systems, Man, and Cybernetics* 11(1): 81-96

Liberatore, M J , and Titus G J 1983 Management Science Practice in R&D Project Management. *Management Science* 29, 962-974

Loeb, M 1975 Coordination and Informational Incentive Problems in the Multidivisional Firm Ph D diss , Graduate School of Management, Northwestern Univ

Luce, R D , and Raiffa, H 1957 *Games and Decisions* New York: Wiley & Sons

Lynch, F ; Marshall, C ; and O'Connor, D 1986 AI in Manufacturing Start-Up Paper presented at Computer and Automated Systems Association, Society of Mechanical Engineers Ultratech Conference on AI in Manufacturing

McCarty, L T., and Sridharan, N S 1981. The Representation of an Evolving System of Legal Concepts. In Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 246-253

Malcolm, D G ; Rosenboom, J H ; Clark, C E ; and Fazar, W 1959 Application of a Technique for Research and Development Program Evaluation *Operations Research* 7(5)

Malone, T W , and Smith, S A 1984 Tradeoffs in Designing Organizations: Implications for New Forms of Human Organizations and Computer Systems, Technical Report, CISR WP #112, Sloan WP# 1541-84, Center for Information Systems Research, Sloan School of Management, Massachusetts Institute of Technology

Marca, D , and Cashman, P 1985 Toward Specifying Procedural Aspects of Cooperative Work In IEEE Proceedings of Third International Workshop on Software Specification and Design, 151-154

Marschak, J , and Radner, R 1972 Economic Theory of Games, Monograph 22 Cowles Foundation , Yale Univ

Nash, J F 1950 The Bargaining Problem *Econometrica* 18: 155-162

National Air and Space Administration 1979 Voyager Encounters Jupiter

Preiss, K 1976 Engineering Design Viewed as an Activity in Artificial Intelligence, Technical Report SRIN-167, Stanford Research Institute (SRI)

Pritsker, A A B ; Watters, L J.; Wolfe, P M ; and Happ, W 1966 GERT: Graphical Evaluation and Review Technique, Part I *The Journal of Industrial Engineering* 17(5): 267-274

Project/2 User's Manual, Sixth Edition 1981 Project Software & Development, Inc , Cambridge, MA

Pruitt, D G 1981. *Negotiation Behavior* New York: Academic

Reddy, Y V ; Fox, M S ; and Hussain, N 1985 Automating the Analysis of Simulations in KBS In Proceedings of Summer Computer Simulation Multiconference, 34-40

Rieger, C , and Grinberg, M. 1977 The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 250-255

Roth, S F.; Mesnard, X ; Mattis, J A ; Kosy, D W ; and Sathi, A 1986 Experiments with Explanation of Project Management Models, Technical Report, Intelligent Systems Laboratory, The Robotics Institute, Carnegie-Mellon Univ

Sacerdoti, E D 1977 *A Structure for Plans and Behavior.* New York: American Elsevier

Sacerdoti, E D 1974 Planning in a Hierarchy of Abstract Spaces *Artificial Intelligence* 5(2): 115-135

Saitow, A. R 1969 CSPC: Reporting Project Progress to the Top *Harvard Business Review* 47(1): 88-97.

Sathi, A ; Fox, M S ; and Greenberg, M 1985 Representation of Activity Knowledge for Project Management In IEEE Transactions on Pattern Analysis and Machine Intelligence 7(5): 531-552

Schmidt, C F ; Sridharan, N. S.; and Goodson, J L 1978 The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence *Artificial Intelligence* 11(1-2): 45-83

Smith, R G 1980 The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver In IEEE Transactions on Computers C-29(12): 1104-1113

Smith, R G 1978 A Framework for Problem Solving in a Distributed Processing Environment Ph D diss , Computer Science Dept , Stanford Univ

Smith, S F 1983 Exploiting Temporal Knowledge to Organize Constraints, Technical Report, CMU-RI-TR-83-12, Intelligent Systems Laboratory, The Robotics Institute, Carnegie-Mellon Univ

Stallman, R M , and Sussman, G J 1977 Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis *Artificial Intelligence* 9(2): 135-196

Stefik, M 1981 Planning with Constraints (MOLGEN: Part 1), 111-139; and Planning and Meta-Planning (MOLGEN: Part 2), 141-169 *Artificial Intelligence* 16(2)

Talbot, F B 1982. Resource-Constrained Project Scheduling with Time Resource Tradeoffs, The NonPreemptive Case *Management Science* 28(10): 1197-1210

Tate, A 1977 Generating Project Networks. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 888-893

Tichy, W F 1980 Software Development Control Based on System Structure Description Ph D diss , Computer Science Dept , Carnegie-Mellon Univ

Turban, E 1976 The Line of Balance—A Management by Exception Tool In *Project Management: Techniques, Applications, and Managerial Issues,* 39-47 Atlanta, Ga : American Institute of Industrial Engineers, Inc

Weiner, J L 1980 BLAH, A System That Explains Its Reasoning *Artificial Intelligence* 15(1-2): 19-48

Wiest, J D 1967 A Heuristic Model for Scheduling Large Projects with Limited Resources *Management Science,* 13(6): 359-377

Wilensky, R 1983 *Planning and Understanding* Reading, Mass : Addison-Wesley

Wilson, R B 1968 The Theory of Syndicates *Econometrica* 36(1): 119-132

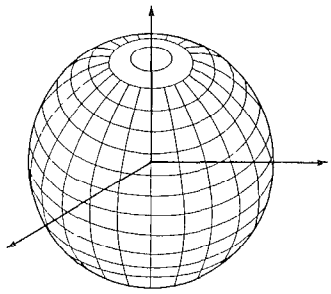
Winston, P H 1975 *The Psychology of Computer Vision* New York: McGraw-Hill

Wise, B P , and Kosy, D W 1985 Model-Based Evaluation of Long-Range Resource Allocation Plans, Technical Report, CMU-RI-TR-85-22, The Robotics Institute, Carnegie-Mellon Univ

Wright, J M , and Fox, M S 1983 SRL: Schema Representation Language, Technical Report, The Robotics Institute, Carnegie-Mellon Univ

Wright, J M ; Fox, M S ; and Adam, D 1984 SRL/2 User's Manual Technical Report, The Robotics Institute, Carnegie-Mellon Univ

Zdonik, S B 1984 Object Management System Concepts In Second Association of Computing Machinery-Special Interest Group in Office Automation Conference on Office Information Systems, 113-19 Toronto, Canada: ACM



call for participation

Workshop on Spatial Reasoning and Multi-Sensor Fusion

October 5-7, 1987

sponsored by AAAI

Spatial reasoning is central to the interaction of an intelligent robot with its environment. Although the problems are somewhat different for mobile and stationary robots, the basic need for correlating perceived information -- which due to viewpoint limitations in most cases constitutes only partial evidence about scene entities -- with the stored world knowledge remains the same. Also common to both cases are the problems of integrating incoming information through various sensors, such as photometric, range, tactile and force/torque. Such issues will form the focus of this workshop. In particular, the topics that will be highlighted at the meeting include

- Reasoning about shape from partial evidence
- Fusion of photometric and range data for mobile robots
- Fusion of 2D, 3D, tactile and F/T sensing for assembly robots
- Evidential reasoning for verification vision
- Reasoning architectures for spatial data
- Programming paradigms for spatial reasoning
- Formal theories of spatial reasoning
- Spatial planning and problem solving

Papers on these topics are invited for consideration. Three copies of an extended abstract or a full-length paper should be sent to either of the following two addresses prior to March 15, 1987.

Su-shing Chen
Dept of Computer Science
University of North Carolina
Charlotte, NC 28223

Avi Kak
Robot Vision Lab
EE Building, Box 121
Purdue University
W. Lafayette, IN 47907

This workshop will be held October 5-7, 1987 at the Pheasant Run Resort in St. Charles, Illinois, a distance of 25 miles from Chicago's O'Hare International Airport. The resort management will provide transportation between the airport and the workshop site. Pheasant Run encompasses 300 acres of the beautiful Fox River Valley. Resort facilities include championship golf courses, indoor and outdoor tennis and basketball courts, Fox River boat rides, etc. Pheasant Run Theatre features top name entertainment and critically acclaimed hit plays.

PROGRAM COMMITTEE

Su-shing Chen (Co-Chair)
Avi Kak (Co-Chair)

Jake Aggarwal
Ruzenna Bajscy
Tom Garvey
Tom Henderson
Tod Levitt
Linda Shapiro