Int. J. Production Economics 68 (2000) 185-197

Application of Multiagent Systems in Project Management

Yuhong Yan, Torsten Kuphal, Jürgen Bode

Institute of Production Management and Industrial Information Management

Department of Economics and Management

University of Leipzig

Maschnerstr. 31, 04109 Leipzig, Germany

yuhong.yan@utoronto.ca

Abstract

Global markets often require the coordination of firms across the world to gain maximum competence. The organization structure and the management of a firm tend to be project-oriented, and members of a project team are distributed throughout a network. In this paper we use multi-agent systems as a technique to support project management in a distributed environment. Activity agents and resource agents represent activities and resources in a project. Service agents are automated experts for special project management tasks. This paper presents methods to schedule activities and resolve resource conflict by message exchanging and negotiation among agents. Finally, a prototype of project management tool is implemented, which uses a network of agents to provide services to team members distributed throughout Internet.

1. Introduction

An increasing number of products and services are generated by project organizations. The reasons are: firstly, shorter product life cycles, greater variety of products, and increased customer responsiveness reduce the production volumes of identical products. Typically, projects deal with singular, one-of-a-kind products. Secondly, companies, or units within companies, concentrate their activities to their core business. This concentration consequently requires the formation of highly cooperative teams in order to bring about complex products. Advanced project management techniques must be developed to meet the requirements of distribution and high complexity.

As project team members can be located at different places, *resources* and *operations* of a project are distributed by nature. In contrast, *planning* and *control* of projects (henceforth called *project management*) is often centralized. All information relevant to the project as a whole should be passed to the project manager. Information of interest for other team members is often transferred via the project manager as well, even if it is not crucial from the project's point of view. Centralized project management creates long communication paths and makes the exchange of information cumbersome, redundant and unreliable.

In a large engineering project, for instance, unforeseen problems occurred when testing an electronic circuit. Testing activities had therefore to be extended for one week. This, in turn, freed a resource temporarily which was only needed at the end of testing. In conventional project management, these two facts would have to be reported to the project management.

The project manager would insert the extended duration into his project management database, and he would check if other activities could use the resource while it is idle.

The conventional method, however, is prone to errors. The testing crew might not report to the project manager; the project manager might insert mistaken information into the database; he might delay the update of the database; he might not be able to communicate all relevant information of the free resource to all parties interested. Such errors could be avoided if all information would be kept locally in a distributed manner, and if managers and team members would have appropriate access to all relevant remotely held information. Mistakes are less likely if information is stored only once, and at the location where it is generated.

We will present a multi-agent system that is an implementation of a distributed project management tool. Activities, resources, and important functions are represented as agents in a network. No single computer contains all project-relevant data. An MPM chart, for example, is implemented as a distributed collection of activity agents and pointers from predecessors to successors in the net. In an extreme case, each activity, and each resource, resides as an agent on a separate computer. Thus, the manager of an activity, or a resource, needs only to update information locally. Local information then is accessible remotely from all sites and will be used automatically by all agents in the system.

Service agents incorporate expert knowledge on specific project management tasks. The buffer time agent is a simple example: it communicates with activity agents in the network in order to determine the buffer time of a specific activity. All main functions of a project management tool are realized as service agents. In this paper we will present a prototypical service agent that determines the total project duration.

Service agents need only be implemented once in the whole network. If standards are open, anyone can program a service agent and make it accessible to anyone else, much as is the case today with WebPages, or Applets. Thus, a multi-agent project management tool consists of two parts: a small core program that is mainly able to run remote service agents, and a dynamic set of agents residing anywhere in the network. The functions of such a tool would grow as the number of service agents offered throughout the net would be increased.

A prototype system using Java and the Internet as system environment will be described in the final section.

2. Relevant Work

2.1 Multiagent Systems in Industry

Multiagent systems (MAS) are a branch in Distributed Artificial Intelligence (DAI). The term *agent* represents a hardware or (more usually) software-based computer system that has the properties of autonomy, social ability, reactivity, and pro-activeness. A stronger notion of agent adopts mentalistic notions, such as knowledge, belief, intention, and obligation [Wooldridge and Jennings, 1995].

Agent technologies are applied into the areas in industry. It is motivated by the observation that computer-based industrial applications are currently testing the limits of system engineering in terms of their sheer size and amount of knowledge they need to embody [Jennings and Wittig, 1992]. The conventional systems for manufacturing are centralized. A central database provides a consistent global view if the state of the enterprise, which is hard to be maintained in current distributed manufacturing environment. All the production activities are carefully planning in advance, which are hard to be changed when uncertainty

occurs. Agent approach replaces the central database and the central control computer with a network of agents, each endowed with a local view of its environment and the ability and authority to respond locally to that environment. The overall system performance is not globally planned, but emerges through the dynamic interaction of agents in real time. Knowledge is integrated into agent to deal with uncertainty and computational complexity.

Planning and control in manufacturing is the most common manufacturing application [Parunak, 1996]. Design process [Syscara, 1991, Klein, 1991], and information integration [Sandia, 1997] are the other two main applications. Some other particular applications include power distribution system [Wittig, 1992], logistics [Fordyse et al., 1992].

MAS system architecture influences information exchanging patterns and relationships between individual agents. Hence it is the most common research topic. Most cases define physical entities, such as a company, a job-shop, a workstation, and a person, as agents. Their architectures correspond to the organizations in real world. For example, in the agent architecture developed in Sandia National Laboratories, each machine (or robot) and each design workstation are represented by an agent respectively. An agent inherits all the functions in a physical entity. Therefore each agent has different functions and the functions should be defined when an agent is developed.

MAS are high devoted into shop-floor control and scheduling. Although what we are interested is project management in general meaning, research results about how to deal with precedence constraints and resource constraints are valuable. GE Job Shop [Baker 1991] and CASCADE [Parunak, 1988] are this kind of MAS prototype systems. In the former case study, a job shop that includes hundred machine tools produces parts for steam turbines. An agent presents each workstation. Based on Contract Net Model agents are organized. Orders are delivered to workstations by biding and contract mechanism. In the later case, work-in-process inventory is the control aim. Agents represent martial containers and workstations. Scheduling strategies, such as “opportunistic scheduling”[Sadeh, 1994], are used as tools to satisfy constraints. In traditional AI, temporal logic algorithms [Bell, 1989] and constructive and repair-based algorithms based on constraint propagation are developed to satisfy constraints. These algorithms sometimes are used in MAS system.

These algorithms or strategies are based on centralized control. They are executed by the project manager alone and all necessary data are required to obtain before execution. No local interaction is considered in problem solving. On the viewpoint of whole general project, our research focus on general features in common projects and how to provide support to achieve management in decentralized environment.

Our system is based on the viewpoint of project management. We define three types of agents, activity agent, resource agent, and service agent. A physical entity may be represented by several different kinds of agents. When the functions of entity are changed, new agents can be created and old agents can be eliminated, which is easier than changing the internal structure of an agent as in existing systems.

One of the advantages of MAS comes from the cooperation among agents, which enables better solutions to problems that cannot easily be solved by centralized method. Conflicts are inevitable in cooperation process because of different opinions from individual agents. Negotiation is believed a promising method to resolve conflicts in MAS environment. Former research provides many choices of negotiation protocols, which depend on different approaches to express mental states [Müller, 1996]. Negotiation strategies are a hard issue in research. Utility function based on game theory gives out decision criteria in negotiation

process [Zlotkin and Rosenschein, 1996]. But real applications are too complex to give a definition of utility in real application. Rule-based reasoning [Doran, Frankling, and Jennings, 1997] and Case-based reasoning [Sycara, 1991] are other solutions. Although some researchers want to obtain general strategies suitable to all applications [Pruitt, 1981], it is still a far goal from current research level. More feasible way is to establish complete strategies in special application domain. It is also a proper method to set up industrial applications. In our paper, we study the conflicts in resource allocation and try to solve it using this method.

2.2 Software for Project Management

In this section we summary characteristics of commercial software in project management. MS Project, Project Scheduler, CA-SupperProject, and Milestones are some commonly used project management software [Gliedman, 1998].

These commercial software tools consist of three key elements: a database that holds all the information about the project, a spreadsheet that calculates the costs and resources required for each task, and a scheduling application that charts work schedules in Gantt or PERT. Reporting and multi-project supporting are common functions. Some of them integrate hands-on management tools. For example, "snapshots" of the plan are used for comparison. "What-if" modeling capability enables investigation of the impact upon costs and deadlines of the best-case, worst-case, and median estimates of the duration of each task. What is worth attention is that group communication features are integrated in some of them. Email reception is supported in order to collect information from other team member. Data and results can also be published in Web pages. MS Project 98 even encompasses a small web server in its package. Although a lot mathematical algorithms appear in literature, they are not integrated in any of commercial software.

These kinds of software are mainly centralized. That is they need central database and central decision-making. Human errors in information processing bring mistakes to project management. From our research, we want to provide a distributed software scheme that supports local decision power and autonomous information collection in order to avoid the likeliness of human errors and increase flexibility.

3. Multiagent Systems Applied in Project Management

Projects normally have distributed team members. Team members establish cooperation relationships via electronic links. In this section we present a novel MAS structure supporting project management in a highly distributed environment and describe how the tasks of management can be solved by interaction among agents.

3.1 System Overview

We adopt multiagent system as information infrastructure to support project management in a highly distributed environment. Each activity and resource needed in a project are represented by an agent. The resource agents and activity agents reside at the site of the project team members who own the resource or implement the activities. The functions of project management are taken by service agents.

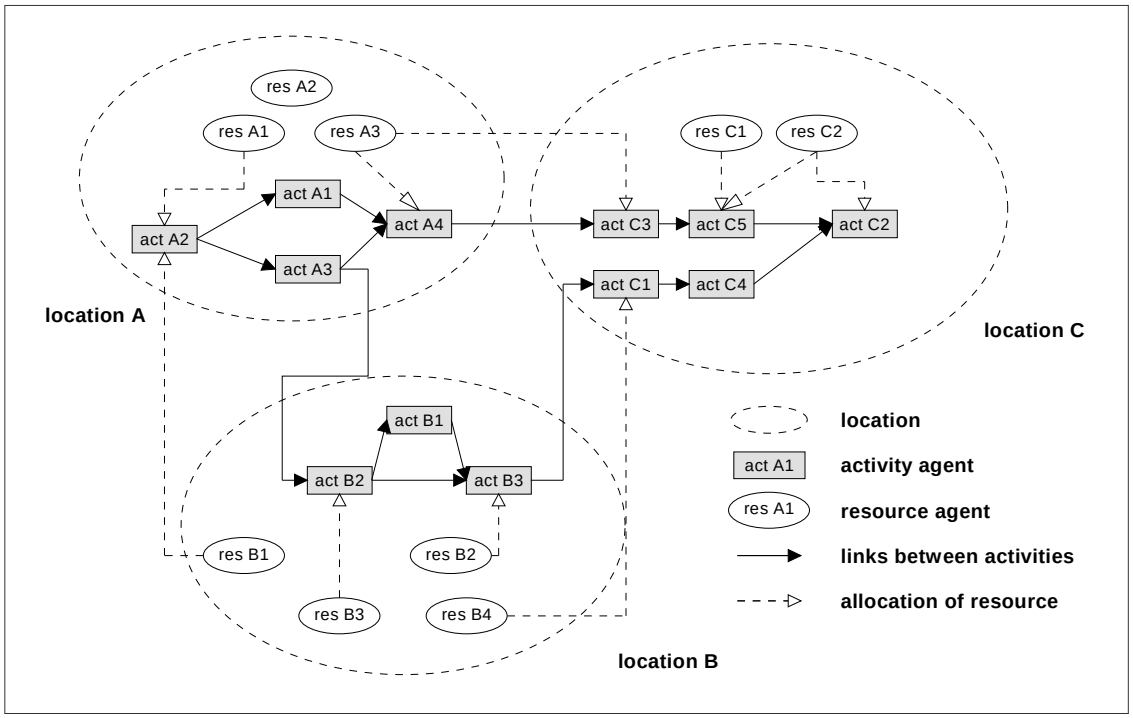

Figure 1: System Structure

Figure 1 shows the resident agents in the system. Several resource agents and activity agents reside in one physical location. The dotted lines show resources used by activities. The solid lines show the temporal relationships of activities in a project. For instance, the activity act B3 has two predecessors act B1, act B2. Its successor act C1 resides at a different location C. Thus, different from conventional project management software, the project network spans more than one computer. No single computer stores the total network (e.g. as MPM chart). In contrast, it is represented in a distributed manner.

Service agents take the functions of project management. Service agents usually are mobile agents, which can reduce the cost of communication and increase speed. For example, the project manager at location A sends out a mobile agent to calculate the duration of the project. First the mobile agent visits the agent act A2 in location A to gather its duration data. Behind activity act A2 the project follows two paths, represented by two arrows following act A2. Therefore, the mobile agent clones itself. The two clones trace down different paths and visit act A1 and act A3 respectively. The two mobile agents collect the duration data, add it the duration computed so far, and go on traveling. Whenever two or more paths in the project network combine (e.g. before act A4) only the clone storing the longest path duration continues travelling while the others are being deleted. In the end, after act C2, the remaining agent returns to the project manager's site and reports the total project duration.

3.2 Agents

3.2.1 Activity Agent

An activity represents an ability of a potential team member. It resides in the location where team member is. For example, a company offers the service to test electronic circuits. Therefore an activity of testing electronic circuit is created at the site of this company and its address can be searched through network. Normally an activity agent is expressed by the

inputs and outputs of the activity. If the input conditions are satisfied, this activity agent is enabled to take on its activity and produces the outputs after its action.

$$(y_1, y_2, y_3, \ldots, y_k) = AAgent(x_1, x_2, x_3, \ldots, x_n)$$

Where x_1, x_2, ..., x_n are the inputs of the activity and y_1, y_2,..., y_k are the outputs of the activity. AAgent() represents the transfer from inputs to outputs.

Inputs and outputs can be machines, material, or data. In the example, testing electronic broad needs testing machines, labors, the circuit, and testing requirements. If these conditions are matched, this activity is enabled. The output can be a testing report.

Inputs and outputs are useful in determining sequences of activity. When the requirements of a project is broadcasted by project manager, the activity agents whose outputs match or partially match the requirements register themselves at the project manager if their inputs can be satisfied. In order to satisfy its inputs, the activity agent search for the agents whose outputs provide the whole or partial of its inputs. In this way, the sequences of activities are determined by interaction among agents.

In the left side of Figure 2, the three activities can be described as

$$Y1 = F_1(X1)$$

$$Y2 = F_2(X2)$$

$$Y3 = F_3(X3)$$

Where Y1,Y2, Y3 are the output set and X1, X2, X3 are the input set.

If

$$Y1 \cap X3 \neq \phi$$

$$Y2 \cap X3 \neq \phi$$

$$X3 \subset (Y1 \cup Y2)$$

Then activity Y1 and Y2 are the predecessor of activity 3.

By interactive determination of dependencies, the schedule of the project can be determined. This method provides more flexible way of scheduling. The activity agent can choose the most beneficial predecessor to provide its input. Sometimes conflicts arise in scheduling process. For example, two agents compete to get a resource, or the outputs and inputs can not be completely matched. Then agents negotiate with each other to solve the problem. In section 3.3, the negotiation procedure is described.

By keeping a record of all inputs and outputs, the activities are possibly paralleled in time. In above of the right part in Figure 2, the activity 3 can only begin after activity 2 and activity 1 are finished in conventional way because no outputs of the upstream activities can be gotten before the end of them. Considering that some resources are used by the upstream only in a short period, and some output data can be released in the middle of the duration, the downstream activity can be started earlier when the input conditions are satisfied. In the lower of the right part in Figure 2, the activity 3 starts ahead of time after it gets all the inputs from activity 1 and activity 2. One can see that the total duration of the three activities is reduced by such parallelization.

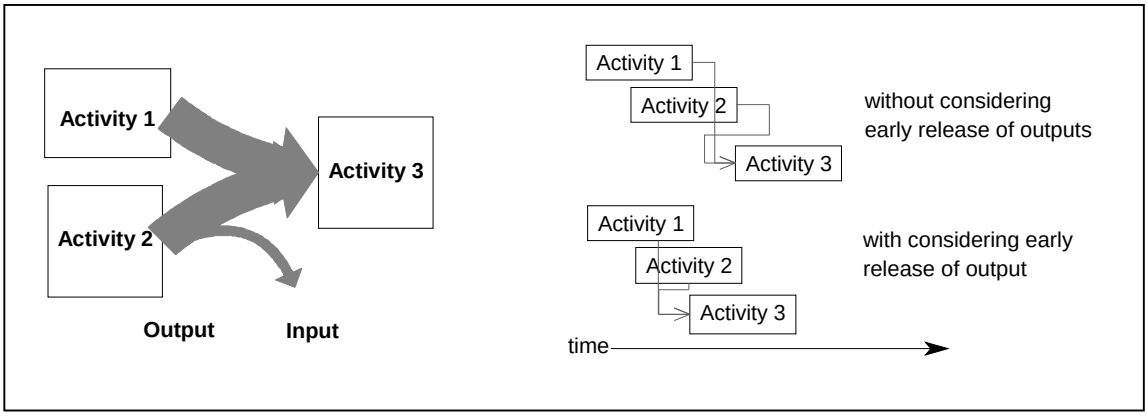

Figure 2: Activity Agents

3.2.2 Resource Agent

Resources involved in manufacturing industry are machine, material, and manpower. Different resources have different properties. Materials can be transported, manipulated and exhausted. Machines and manpower are reusable and shareable.

Each resource is represented by an agent. It is assumed that any resource can be utilized by any activity that requires it. The properties of resources and the utilization status are recorded in the memory of agents.

Resource agent is controlled by sending it commands. For example, if the resource is transferred to another location, a dispatch command can order the agent to also move to the other. If the resource is exhausted, a terminal command deletes the agent. By keeping the record of utilization requirements, it is easy to find the implicit dependency of resources. Normally dependence of an activity on a resource is known. But the interdependence between two activity using a common resource must be inferred. Now resource agent can easily discover the conflict by checking the exploitation status. The involved activity agents negotiate with each other to solve the conflict.

3.2.3 Service Agent

The tasks of project management such as determination of the critical path, time calculation and activity scheduling are taken by service agents. These service agents are normally mobile agents, because mobile agent in our prototype system has a faster speed than massage passing. Mobile agent has the ability to travel to the relevant locations and to retrieve information. For example, duration calculation agent is sent out to calculate the project duration. It visits the activity agents following the topology of MPM and collects data of period of each activity. When it returns home, it reports the result it gets. A resource allocation agent can set out to find available resources in the network.

Service agents are small and single functioned. They are stand-alone packages and can be utilized both manually and other agents. If standards are open, anyone can program a service agent and provide a service to anyone else.

3.3 Problem of Resource Allocation

In this section we use the resource allocation problem to demonstrate how the agents solve problems by interactions among agents.

3.3.1 Resource Allocation Problem in Distributed Environment

Resource allocation is a typical project management problem. In traditional research resource conflicts are created whenever the total resource requirement of activities to be scheduled in a particular time period exceeds the resource available during that period. In mathematical formulation the bound of total resources is a constraint condition [Belhe and Kusiak, 1995].

But when we consider the potential team members come from the distributed space of the whole world, we can not give the bound of total resources. We define the resource conflict whenever one resource is required by several activities simultaneously. As shown in Figure 3, if activity 1 and 2 require resource 1 in time interval [st1,et1],[st2,et2], and st1<st2<et1<et2, then in time interval [st2,et1], a resource conflict happens. The conflict can be detected easily by comparing the upper bound and lower bound of different time interval, because resource agent keeps a record of each utilization requirement. The memory of resource agent is shown on the right side of Figure 3.

Another assumption in traditional project management is that the resource is occupied during the whole period of activity. It is more like a mathematical convenience than real wold situation. For example, a typical constructing activity, erect gantry, needs crane to elevate the gantry. It takes one week. The crane is used in the first half week, while in the latter half week gantry is bound to the columns. We eliminate this assumption. The resource can be released before the completion of activity.

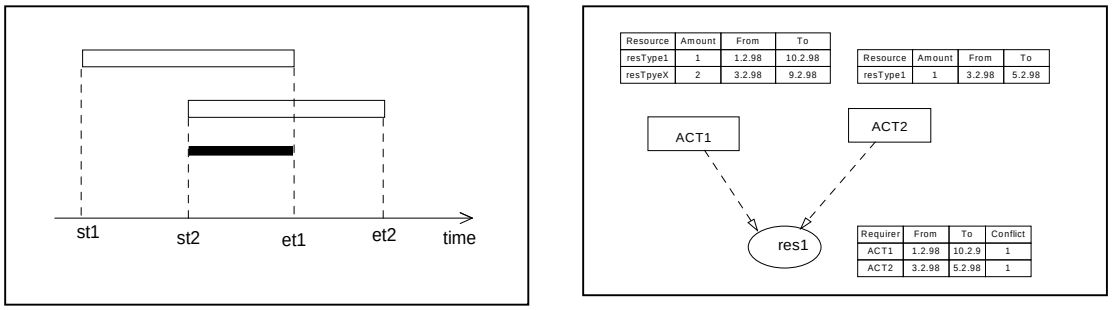

Figure 3: Resource Conflict

3.3.2 Strategies

The resource conflicts are solved via negotiation between agents. In this section we analyze the strategies in negotiation.

The objective is minimization of project duration. The activities on critical path has the priority to get resource for their delays surely extends the makespan of the whole project. Here only the case in which conflict is between a critical path activity and a non-critical path activity (CP-NCP) is analyzed.

In Figure 4.A a project is shown using PERT diagram. The critical path is represented by the links with darker arrow. Dotted links represent idle activities. Assume that activity 2-4 and activity 2-3 require the same resource to start work. Because they have the same beginning time, a resource conflict arises.

The strategies are expressed as a decision tree Figure 5. While activity 2-3 is on critical path, it takes “critical-path-strategy”. That is it asks the priority to get resource. “Critical-path-strategy” is the left branch of the tree. Non-critical path activity 2-4 follows the “non-critical-

path-strategy", which is the right branch. It chooses the strategy on the left leave because a left-advance searching method is used here. It checks if it can shift the start time during its float time to avoid conflict. It is the situation in chart 3.A. Activity 2-4 can begin during the process of activity 3-4 to avoid conflict.

A complex situation is that the float time of the non-critical path activity is not enough for time shifting. It is the case in chart 4.B. In order to retain project duration, activity 2-4 uses "parallel-strategy". It asks activity 2-3 to release the competed resource before its completion, as shown in Figure 4.C. If activity 2-3 accepts the negotiation ends. If not "sequential-strategy" is adopted. Activity 2-4 begins when activity 2-3 is completed, as shown in chart 3.D. Because of the bottleneck of resource, both of the two activities are on the new critical path. The total duration is increased. In this case this solution should inform project manager.

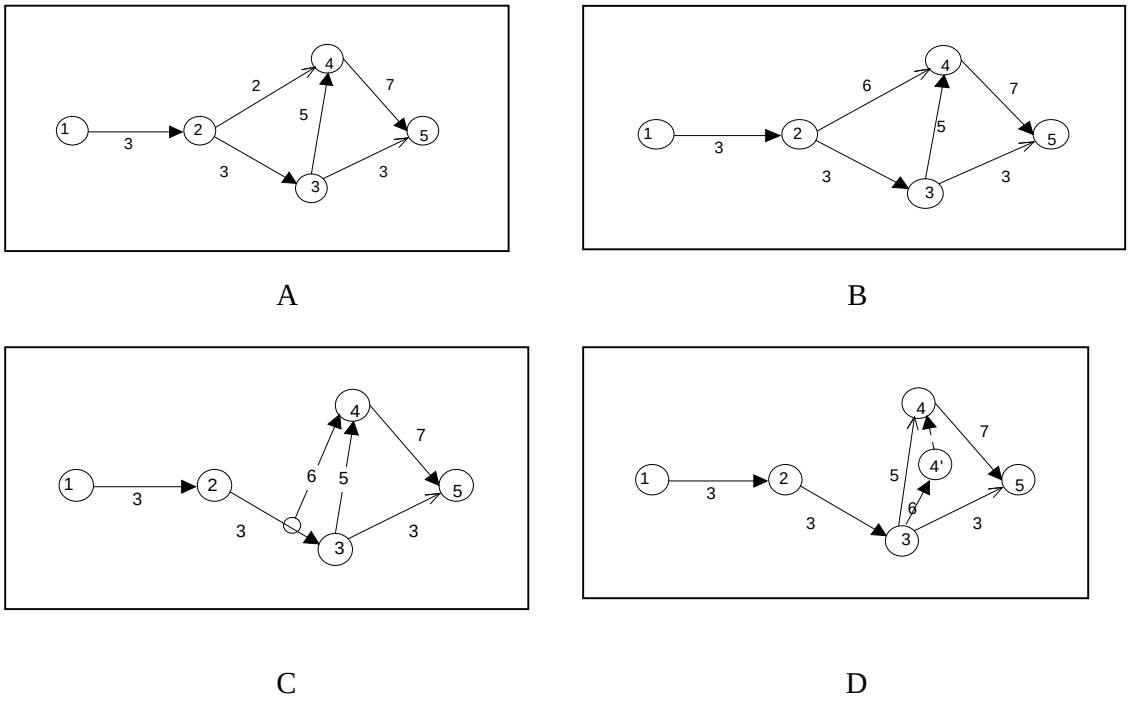

Figure 4: A Case to Resolve a Resource Conflict

These strategies are expressed into production rules. As shown in Figure 6 these rules use KQML performatives and notions in Belief-Desire-Intention theory (beginning with capital characters) [Haddadi and Sundermeyer, 1996]. The left side is the rules for critical path strategy and the right is the rules for critical path strategy. When situation matches the condition of a rule, the actions (predefined functions in small character) in the right side are enabled. When the negotiation process succeed, both sides take out begin_task_at() function to initiate the task.

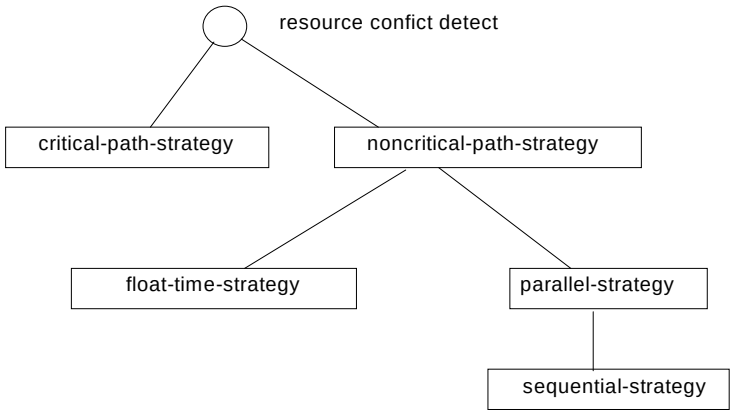

Figure 5: Strategies to Resolve Resource Conflict

```
CP rules
(r1) IF (?TF1=total_float())==0
     THEN Believe: myself(CP)
(r2) IF Believe: myself(CP) AND
          Believe: conflict_with(?ag2)
     THEN
          achieve( receiver: ?ag2,
                content: ask_ priority(#me));
(r3) IF receive: ask_duration(?ag2)
     THEN achieve(receiver: ?ag1, content:
                duration(#me, d1)
(r4) IF receive: ask_parallel(?t)
     THEN release_at(d1-?t)
(r5) IF Believe: release_at(?tx)
     THEN achieve(receiver: ?ag1, content:
                accept_parallel()
(r6) IF Believe: NOT release_at(?tx)
     THEN achieve(receiver: ?ag1, content:
                not_accept_parallel();
(r7) IF receive: accept_priority()
     THEN begin_task_at(ON_TIME)
```

a. Critical Path Strategies

```
NCP rules
(r1) IF (?TF2=total_float())<>0
     THEN Believe: myself(NCP)
(r2) IF Believe: myself(NCP) AND
          Receive: ask_priority(?ag1)
     THEN achieve(receiver: ?ag1, content:
                ask_duration(#me);
(r3) IF receive: duration(?ag1, ?d1)
     THEN SUCCESS
(r4) IF  ?TF2 > ?d1
     THEN achieve(receiver: ?ag1, content:
                accept_priority())
          begin_task_at(?d1)
(r5) IF ?TF2 < ?d1
      THEN achieve(receiver: ?ag1, content:
           ask_parallel(?d1-?TF2)
(r6) IF receive: accept_parallel()
     THEN begin_task_at(?d1-?TF2)
(r7) IF receive: not_accept_parallel()
     THEN begin_task_at(?d1)
          achieve(receiver: PM, content:
               delay(?d1-?TF2)
          achieve(receiver: ?ag1, content:
               accept_priority())
```

b. Non-Critical Path Strategies

Figure 6: Translate Strategies into Rules

4. Prototype System and Examples

4.1 System Implementation

We implemented a prototype system to illustrate our ideas. Java is chosen as programming language. Java has integrated network abilities, which is feasible for a distributed

environment. It also has the advantages of platform independence, good security mechanism and ease of programming.

A lot of Java based tools are developed by computer companies and universities, for example, IBM Aglets, JAMFAS, AGTLite. Some of them have the ability to develop mobile agents. Mobile agents are programs that can be dispatched from one computer and transported to a remote computer for execution. The advantage of mobile agent are that [Baumann, 1997]: 1) distribute on command, a program can be installed on the remote location by transporting code form local site; 2) reduction of communication cost, while communication takes place between a mobile agent and the programs on the visiting location; 3) asynchronous tasks, the connection to client can be stopped and resumed after the results are gotten; 4) scalability due to dynamic deployment of agent programs.

We use Voyager from ObjectSpace Company as our basic tool. Voyager provides Java classes to build resident and mobile agent. It allows using regular message syntax to construct remote objects, send them messages and move them between programs and locations. It supports TCP/IP protocol.

We need only to develop client programs for the Voyager server is included in the software package. Our prototype system includes the core agent system made up of multiply agents and a graphical user interface (GUI), which provides the possibility to control and supervise the agent system. Figure 7 shows the access hierarchy of client program.

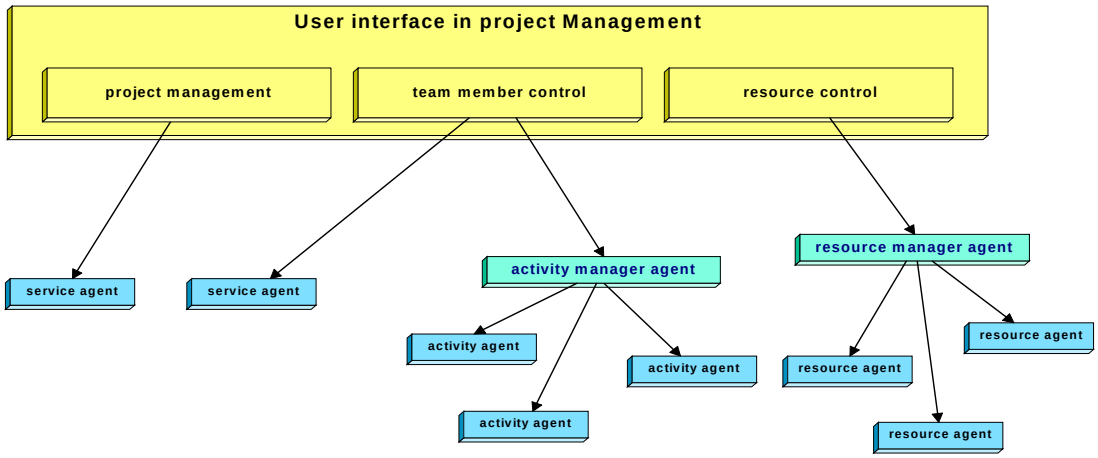

Figure 7: The Access Hierarchy in the Prototype System

The prototype integrates the functions of project management and administration of activity agents and resource agents. The activity manager agent can create instances of activity agents for its local site and detect other activity agents in remote locations. The resource manager agent has the similar functions. The project management module can send out service agents to carry out management tasks. The team member control module can also send out service agents to do tasks for activities, for example, detecting resources for each activity.

From the user interface, one can get a overview of the whole project and gather the status of the project. This interface can be started as a stand-alone JAVA program or embodied in a web page browsed by a internet browser.

4.2 An Example

A company wants to produce a special computer installation with own hardware and software for a customer. Three departments of the company are committed to the work. The customer service department works directly with the customer. The hardware department produces the required hardware and the software department develops the software. These three departments are distributed at different locations. Table 1 contains all activities in detail. The MPM network of the project is drawn in Figure 8. The brackets contain the duration. Activities owned by different departments have different colors. The bold links follow the critical path that has duration of 21. In this example, the duration of the project is calculated by a mobile agent.

Activity	Description of the activity	Predecessor	Successor	Department	Duration
A	Planning stage of the project	-	B, C, D	Service	10
B	Production of the central unit	A	G	Hardware	5
C	Provision of the periphery	A	G, H	Hardware	2
D	Development of basic software	A	E, F	Software	4
E	Development of testing software	D	G	Software	4
F	Development of customer software	D	K	Software	3
G	Test of the computer hardware	B, C, E	K	Hardware	2
H	Provision of additional devices	C	K	Hardware	5
K	Delivery and installation	F, G, H	-	Service	1

Table 1 : Activity list for the example

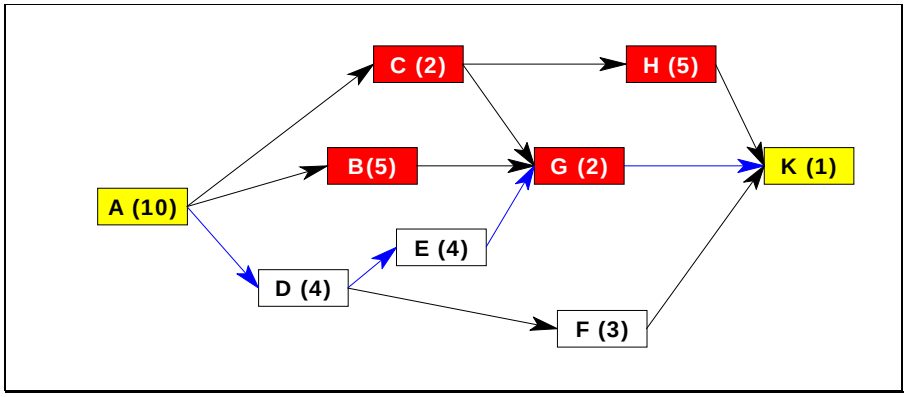

Figure 8: MPM Graph of the example

Figure 9 is a snapshot of the user interface in the hardware department. The activities that are implemented in this department, G, C, H, and B, are listed on the left side. The selected activity G has predecessors E, C and B that can be seen in the predecessor field. Notice that E is an external link to the software department. As you can see in the dropped down successors list all other implemented activities in the network are available. In current stage, the inputs and outputs are not used to determine predecessors and successors as described in section 3.2.1.

After definition of all activities and their links, it is possible to calculate the duration and the critical path. In Figure 10 we choose "forward calculation" method with the strategy of keeping "best agent". A mobile agent is sent out to trace down the MPM network. It collects the duration data of activity A, and then clones itself into three clones to follow the different paths. When paths combine (e.g. before activity G), only the clone storing the longest path duration continues travelling. After visiting activity K, the service agent come back to the project manager and reports the duration. From Figure 10, one can see that the service agent reports that the duration is 21.

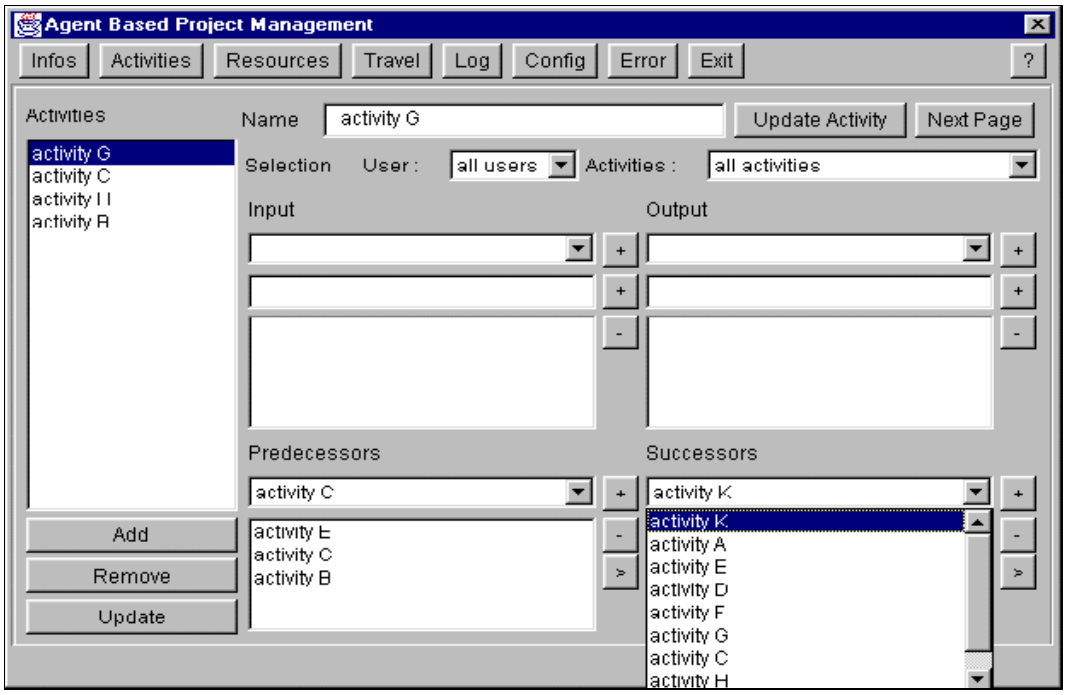

Figure 9: User Interface of the Hardware Department

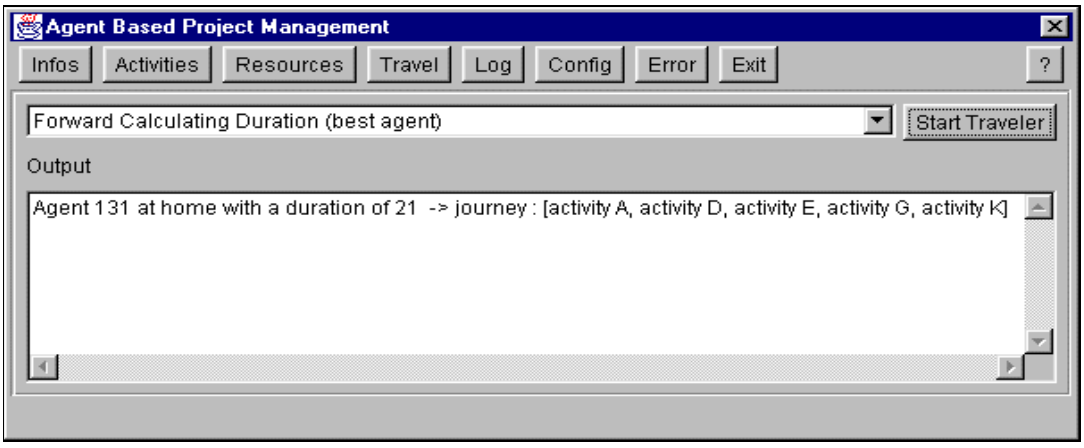

Figure 10: Report of the Service Agent with "Best Agent" Strategy

5. Conclusion and Future Work

In this paper we apply the techniques of intelligent agent to support the management of virtual project. The competence of our prototype system is described by an example.

From the viewpoint of system organization, we adopt a novel structure. Agents correspond not to physical objects. Only three types of agents are defined and they jointly or separately can represent any object. Therefore this structure is suitable to all applications.

These agents are small and share the most common characteristics. Because their functions are unitary, it is easier to define and program them than other huge and complex agents.

These agents are stand-alone packages and can be called interactively. They are like blocks that can be built into any system. The systems built up from them have strong flexibility. If something changed, a new agent can be created while the out-of-date agent is deleted. This process is very simple in our prototype system.

In this paper we discuss the resource allocation problem in distributed environment. The strategies in CP-NCP case are analyzed. It provides a new approach in solving this problem.

One of the future work focuses is on the problem of resource allocation. Other cases, such as two non-critical path activities will be analyzed. It concerns different criteria, e.g. less waiting time. Cost effect will be considered. In these cases, more complex strategies and mechanism are concerned.

Another direction is in the area of Concurrent Engineering. How to parallel the design activities and how to balance cost-time tradeoff in CE based design project are interesting topics.

References:

Baumann, J., Hohl, F., Straßer, M., Rothermel, K., 1997, "Mole: Concepts of a Mobile Agent System", http://www.informatik.uni-stuttgart.de

Baker, A., 1991, "Manufacturing Control with a Market-driven Contract Net", Ph.D. Dissertation, Dept. Of Elect. Engineering, Rensselaer Institute, Troy, NY.

Belhe, U. and Kusiak, A., 1995, "Resource Constrainted Scheduling of Hierarchically Structured Design Activity Networks", IEEE Trans. On Engineering Management, 42(2), pp150-158.

Bell, C., 1989, "Maintaining Project Networks In Automated Artificial Intelligence Planning", Management Science, 35(19), pp1192-1213.

Doran, J. E., Franklin, S., Jennings, N.R., and Norman T. J.. 1997. "On Cooperation in Multi-agent Systems", The Knowledge Engineering Review, 12(3), pp309-314.

Fordyse, K., Dunki-Jacobs, R., Gerard, B., Sell, R. and Sullivan, G., 1992, "Logistics Management System: an Advanced Decision Support System for the Fourth Decision Tier Dispatch or Short-interval Scheduling", Production Operations Management, 1(1), pp70-86.

Gliedman, J. (1998) "Mission Control", ZDNet Magazine,
http://www.zdnet.com/cshopper/content/9801/cshp0118.html.

Haddadi, A. and Sundermeyer, K., 1996, "Belief-Desire-Intention Agent Architectures", Foundations of Distributed Artificial Intelligence, Edited by G. M. P. O'Hare and N. R. Jennings, JOHN WILEY &SONS, New York, Inc, pp169-187.

Klein, M., 1991, "Supporting Conflict Resolution in Cooperative Design Systems", IEEE Trans. On System, Man, and Cybernetics, Vol. 21, No. 6, pp1379-1389.

Jennings, N.R., Wittig, T., 1992, “ARCHON: Theory and Practice”, Distributed Artificial Intelligence: Theory and Praxis, edited by N. A. Avouris and L. Gasser, KLUWER Academic Publishers, Dordrecht, New York, pp. 179-195.

Müller, J., 1996, “Negotiation Principles”, Foundations of Distributed Artificial Intelligence, Edited by G. M. P. O’Hare and N. R. Jennings, JOHN WILEY &SONS, New York, Inc, pp211-229.

Parunak, H. V. D., 1996, “Applications of Distributed Artificial Intelligence in Industry”, Foundations of Distributed Artificial Intelligence, Edited by G. M. P. O’Hare and N. R. Jennings, JOHN WILEY &SONS Inc, New York, pp139-164.

Parunak, H. V. D., 1988, “Distributed Artificial Intelligence Systems”, Artificial Intelligence Implications for CIM, edited by A. Kusiak, Springer-Verlag, Berlin, pp225-251.

Prasad, B. 1996, “Concurrent Engineering Fundamentals: Integrated Product and Process Organization”, Prentice Hall, New Jersey.

Pruitt, D. G., 1981, “Negotiation Behavior”, Academic Press, New York.

Sadeh, N., 1994, “Micro-Opportunistic Scheduling: The Micro-Boss Factory Scheduler”, Intelligent Scheduling, edited by M. Zweden, and M. S. Fox, MORGEN KAUFMANN PUBLISHERS, San Francisco, pp67-98.

Sycara, K.P., 1991, “Cooperative Negotiation in Concurrent Engineering Design”, Computer-Aided Cooperative Product Development, Edited by D. Sriram, R. Logcher, S. Fukuda, Springer-Verlag, Berlin, 1991, pp. 269-296

Sandia National Laboratories, 1997, “Sandia Intelligent Agents for Manufacturing (SIAM)”, http://nittany.ca.sandia.gov:8001/

Wittig, T., 1992, “ARCHON: An Architecture for Multiagent Systems ”, ELLIS HORWOOD, New York, 1992.

Wooldridge, M., and Jennings, N., 1995, “Intelligent Agents: Theory and Practice”, Knowledge Engineering Review, Vol. 10, No. 2, pp. 115-152.

Zlotkin, G., and Rosenschein, J. S., 1996, “Compromise in Negotiation: Exploiting Worth Functions Over States”, Artificial Intelligence, Vol. 84, pp150-178.