IEEE Region 10 Conference, Tencon 92
11th - 13th November, 1992
Melbourne, Australia

A Scheduler of Daily Personal Tasks on the Basis of the Object-oriented Model

Toyohide WATANABE
Department of Information Engineering,
Faculty of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-01, JAPAN

and Teruo FUKUMURA
School of Computer and Cognitive Sciences,
Chukyo University
101 Tokotate, Kaizu-cho, Toyota 470-03, JAPAN

===== Abstract =====

The scheduling facility of daily personal tasks is one of important application facilities. Some tasks often conflict with other different tasks, while some must be performed cooperatively with other persons. It is troublesome to arrange several conflicted/related tasks into one time axis with other tasks/persons. Our scheduling facility is designed on the basis of the object-oriented paradigm. One of the characteristics is the task inheritance mechanism. This makes it flexible to schedule conflicted tasks mutually. Another is the message passing mechanism. This is successful in the cooperation of shared tasks and negotiation of conflicted tasks.

1. INTRODUCTION

The scheduling of daily personal activities is a very intelligent job. Available information systems must be designed, more or less, so as to achieve the effective scheduling mechanisms, without depending on whether the subject of scheduling is the direct or indirect issue[1-3]. Our objective is to schedule various kinds of tasks, which are attended to each person, effectually with respect to the relationships among persons. The tasks of one person are always related to tasks of the others because individual persons keep the different partnerships among their fellows of organizations: the task to be allocated to one person is not necessarily independent of the intentions of other persons. Our task scheduling problem must be attached on the basis of the process scheduling mechanism[4] though it is functionally looked upon as a kind of reservation systems.

We introduce a modeling approach based on the object-oriented paradigm: persons are specified as active entities. Of course, our task scheduling problem can not always be solved completely by the basic mechanism of the object-oriented paradigm. This is because the object-oriented paradigm does not provide the ability to control the relationship, like the grouping concept, dynamically though it is effectual to analyze the entities and relationships in the real world from a static point of view. In our framework, everything is an entity and also is represented as the object instance: person and organization.

2. TASK SCHEDULING PROBLEM

We define our task scheduling problem, here. The goal is to allocate all tasks, which are assigned to a person, into his daily time table consistently, according to the task properties. In some cases, the allocation strategy is not always easy to satisfy the properties of all tasks completely. This is because tasks are generated independently from different persons/organizations.

【Definition-1 (agent)】
The agents are entities, as pseudo persons, to be manipulated in our task scheduling system. Each agent has his own daily time table, and must schedule his tasks with other agents cooperatively. The agents as autonomous objects have their own peculiar knowledge for handling the task allocation, in addition to their daily time tables and operational procedures.□

【Definition-2 (environment)】
Environments are interrelations among entities, as pseudo organizations. Although each environment also accompanies its own daily time table, it is a heteronomous object.□

【Definition-3 (hierarchical structure of environments)】
The environments construct the hierarchical structure. The link between upper and lower environments represents a kind of aggregation: PartOf relationship. In this case, the tasks scheduled in the upper environment are inherited to the daily time tables in the lower environments.□

This inheritance mechanism of scheduled tasks is conceptually similar to the property inheritance mechanism in the object-oriented paradigm.

【Definition-4 (relationship between agents and environments)】
Individual agents can participate to appropriate environments dynamically, and construct a kind of aggregated relationship: MemberOf. The parts of their scheduled tasks can also be inherited from dynamically linked environments.□

【Definition-5 (task)】
The task is an executable event either to be performed by agents or to be held in environments. The tasks are usually characterized with various attributes: task name, task type, task priority, starting time of task, available period of task, owners (agents or environments) generated the task, members (other agents) related to the task, alternate starting time, conditions and so on.□

It is unnecessary that all attributes are always specified when the task is generated.

【Definition-6 (task type)】
Individual tasks are classified into appropriate classes. The task type indicates the categorized class of tasks.□

【Definition-7 (priority)】
The priority is fundamentally assigned to 2 different task types. The priority among tasks is the partial order. However, another priority can be specified directly to each task in order to manage acute tasks effectually.□

The concept of our priority is not always powerful to solve the conflictions among various kinds of tasks. A practical solution strategy among conflicted tasks will be performed, using several parameters, in addition to the task priority.

【Definition-8 (communication)】
Each agent can communicate with another one by the message passing mechanism in order to allocate mutually related tasks intelligently. When an agent also communicates with all other agents related to an environment, the broadcasting function supported in the environment is effective. □

These definitions make it clear that the framework in our task scheduling system is characterized suitably on the basis of the object-oriented paradigm. In particular, the concept of environment is very successful with a view to modeling the mutual relations among agents, in addition to the concept of agents. We can represent the basic framework of our task scheduling system as illustrated in Fig.1. With these definitions, we define our task scheduling problem.

【Definition-9 (task scheduling problem)】
Our task scheduling problem is to allocate tasks into the daily time tables of the related agents effectively so as to be matchable with the task properties. Of course, the scheduling may be not always performed successfully in point of satisfying with all properties because some properties among tasks often conflict. The cooperation and negotiation processes are necessary to solve such conflictions.□

3. A FRAMEWORK BASED ON OBJECT-ORIENTED PARADIGM

A global framework in our task scheduling system is very similar to that in the object-oriented paradigm, though they are different in point of the detail concepts. All objects in our framework are entities: agents and environments. The properties of object instances in the object-oriented paradigm are destined almost statically since the relationship ISA is

0-7803-0849-2/92 $3.00 © 1992 IEEE

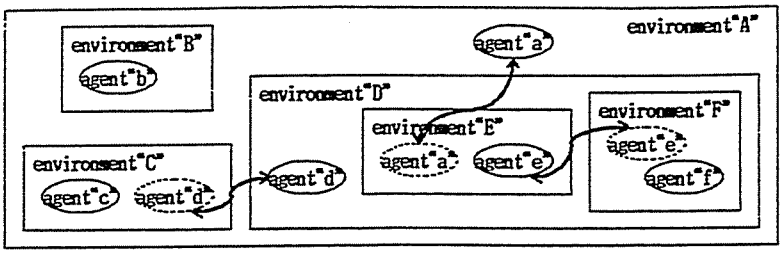


Fig.1 Agents and environments

defined among the classes in advance. While, our autonomous agents have their own peculiar properties in themselves and also attach to appropriate environments dynamically. For example, an agent "b" can move from an environment "B" to another environment "C" in Fig.1. The relationship MemberOf of "b" for "B" is deleted, and a new relationship MemberOf is established for "C". Of course, "b" can also belong to "C" simultaneously in addition to "B". In this case, a part of scheduled tasks in "b" is necessarily inherited from 2 different environments "B" and "C" at once. Namely, the tasks for agents may be inherited from multiple environments.

We can summarize mutual relationships among environments and between agents and environments in Fig.2. The relationships PartOf and MemberOf have the inheritance mechanism of scheduled tasks from the upper to the lower. The relationship PartOf is defined statically, while MemberOf is created dynamically, according to the activities of agents.

4. STRUCTURE OF AGENT

Basically, each agent is composed of daily time tables, procedures, actions, its own knowledge and so on. The knowledge is personal information about the property of agent. Individual agents can select their own desirable tasks from some concurrently conflicted requests/appointments. In this case, the utilization of knowledge supports such a selection process. We show the conceptual structure of agent in Fig.3.

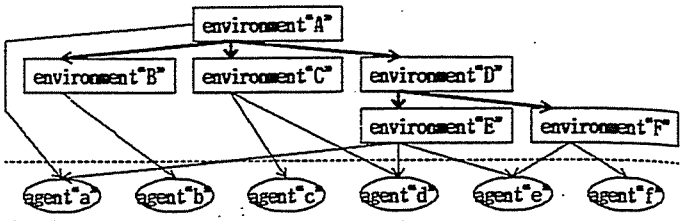
1) State variables: These represent the internal states of individual agents. The variables are refered only by actions and procedures, defined in the agent.
2) Daily time tables: These are the personal daily tables to schedule personal and public tasks consistently, day by day. The other components work to manage these tables effectively. Peculiar tasks in the table may be reedited through the inheritance mechanism of scheduled tasks, planned in some environments.
3) Actions: These interpret messages to/from the other agents, or manage each job.
4) Procedures: These are internal routines to manipulate the daily time tables directly. Therefore, these procedures are operated as slave routines of actions.
5) Message buffer: This buffer holds messages sent from other agents temporarily.
6) Message selector: This picks up an appropriate message from the message buffer one by one, and then transfers to the actions which can evaluate the message.
7) Regular task list: This keeps some regularly scheduled tasks for the agent.
8) Knowledge: This represents the personal information to schedule tasks.

The knowledge and regular task list must be specified when individual agents are instantiated. The agent is usually instantiated in the following description form;

(ex.)

```
agent <agent name> ;
  type { <task type>,-------} ;
  class { { <task>,-------: <task type> } ,-------} ;
  priority { <relationship of task types>,--------} ;
  regularity { <task>: <time attributes>,-------} ;
end;
```

In this description form, the section "type" declares particular task types to be useful in the agent, and the section "class" defines which task types individual tasks must be accommodated into. Of course, some tasks and task types are preassigned in the task scheduling system. Therefore, the sections "type" and "class" are optional. The section "priority" defines the private choice way among 2 different task types, using mathematical comparison operators. Finally, the section "regularity" declares tasks to be performed regularly. For example, we consider an example of agent:



(note) → : PartOf relationship among environments
→ : MemberOf relationship between agents and environments

Fig.2 Relationships

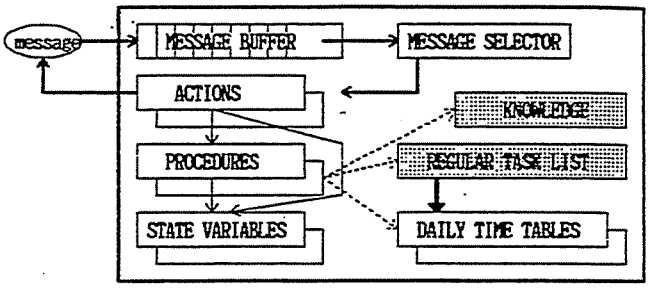


Fig.3 Structure of agent

(ex.)

```
agent A;
  type  meeting, play, study, trip, sleep,-----;
  class  meeting-1, meeting-2, meeting-3: meeting,
         play-1, play-2: play,
         study-1, study-2, study-3: study,
         trip-1, trip-2: trip,
         sleep-1, sleep-2: sleep,
          ..... ;
  priority  meeting>play, play>study, meeting>sleep,----;
  regularity  meeting-1: (Monday, 900//1100),
              sleep-1  : (everyday, 2400//700),
              meeting-2: (Thursday & Friday, 1000//1700),
              play-1   : (10, 1300//1700),
               ..... ;
end;
```

Although this description specifies a property concerned with an agent "A" initially, the attributes of the agent property may be changed successively. This example says that the task type "meeting" is prior to "play" and "play" is prior to "study". In this case, the relationship "meeting > study" is not always derived from both "meeting > play" and "play >study" transitively. If neither "meeting < study" nor "meeting = study" is specified in the succeeded lines, "meeting > study" is generated transitively. Otherwise, "meeting > study" is not acceptable. Additionally, in the section "regularity" regularly scheduled tasks are declared: "meeting-1" must be held from 9:00 to 11:00 every Monday, "meeting-2" is effective on Thursday and Friday from 10:00 to 17:00, and so on. These regularly specified tasks are, in advance, allocated into the daily time tables from the regular task list.

5. STRUCTURE OF ENVIRONMENT

Environments are passive and heteronomous objects, though agents are active and autonomous objects. Therefore, the structures of environments are almost similar to those of agents, except for the functionalities. Of course, the description "environment <environment name> within <environment name>" is used in place of "agent <agent name>".

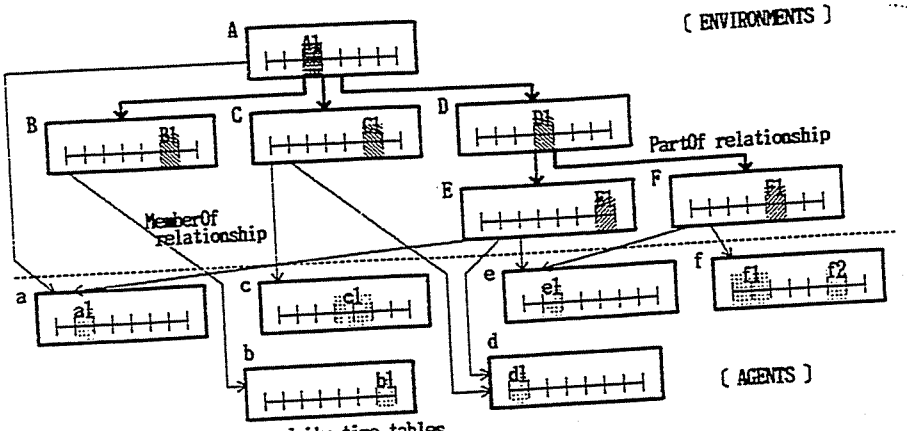


Fig.4 Relationships among daily time tables

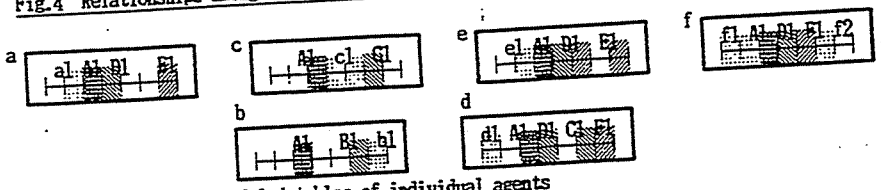
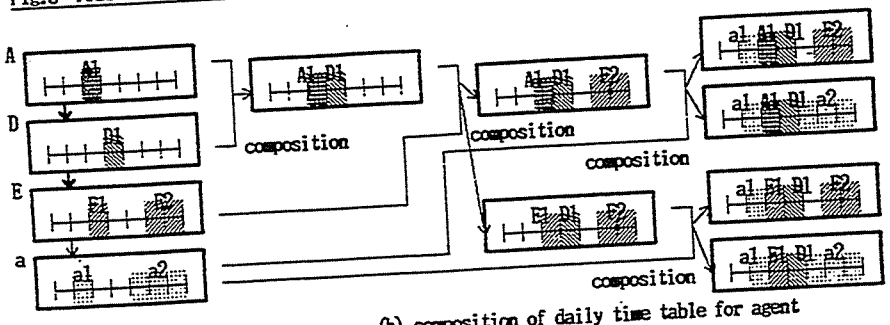


Fig.5 Practically scheduled tables of individual agents



(a) hierarchical structure (b) composition of daily time table for agent

Fig.6 Conflicted tasks among daily time tables

"within" defines a relationship PartOf for the existing environment, if necessary.

The main different facility between agent and environment is the communication method. The communication among agents is 1:1 for agents/environments, while the communication facility of an environment is 1:n for agents. Of course, the communication among environments is not meaningful because environments are passive objects.

6. DAILY TIME TABLE

Under our framework, we mention our task scheduling facility on the basis of the inheritance mechanism among daily time tables. Our relationships PartOf and MemberOf are very powerful in scheduling several kinds of tasks. In Fig.2, "A", "B", "C", "D", "E" and "F" are heteronomous environments, while "a", "b", "c", "d", "e" and "f" are autonomous agents. "B", "C" and "D" are linked to "A" by the relationship PartOf, and "E" and "F" are also related to "D". These relationships are statically defined in advance. "b" attaches to "B", dynamically. Similarly, "c" to "C", "a" to "A" and "E", "d" to "C" and "E", "e" to "E" and "F", and "f" to "F" are corresponded, respectively. These environments and agents have their own daily time tables. Fig.4 shows relationships among these daily time tables, attended inherently to individual objects.

The daily time tables hold only their own directly allocated tasks. However, individual agents can have compositely scheduled tasks derived from environments through the inheritance mechanism of the relationship MemberOf. Of course, the inheritance of scheduled tasks is applicable to the daily time tables among environments according to the relationship PartOf: the lower linked environments can inherit some scheduled tasks from the upper environments. Fig.5 shows the daily time tables of scheduled tasks, composed of both the existing scheduled tasks in environments and their own scheduled tasks of agents.

In such an inheritance mechanism, we must address the confliction issue for scheduled tasks among different daily time tables. Our scheduling strategy need not necessarily allocate tasks exclusively, which conflict or may conflict mutually among daily time tables of both agents and environments because each daily time table holds only its own related tasks. Of course, the consistent task allocation must be done in each daily time table. For example, consider the agent "a" in Fig.4. "a" must inherit the scheduled tasks simultaneously from "A" and from "A", "D" and "E" through multiple inheritance mechanism. As the inheritance path "A" -"a" duplicates partly with the path "A"-"D"-"E"-"a", only the path "A"-"D"-"E"-"a" is effective in practice. In this configuration, we assume that the daily time tables have individual scheduled tasks as shown in Fig.6(a). Such hierarchically inherited tasks generate several daily time tables as illustrated in Fig.6(b). Our scheduling strategy does not resolve the confliction problem on the task allocation, but constructs an effective daily time table on the table composition. In many cases, we observe that appointed plans/jobs may be changed alternately or canceled suddenly until the day that they are performed practically. Our framework is appropriate to manage alternative schedules.

7. SCHEDULING STRATEGY

Each daily time table does not include conflicted tasks as a whole. However, the scheduled tasks may conflict among several daily time tables. Therefore, we must focus on 2 issues: allocation of tasks; and composition of an effective daily time table. Here, we call the existing task in the daily time table as the scheduled task, and also call the task to be allocated now as the current task.

(1) allocation of tasks in each daily time table

The tasks are allocated into the appropriate daily time ranges according to the task properties. If the time position, which is assigned into the current task, is empty, the current task is set to the position. When the currently

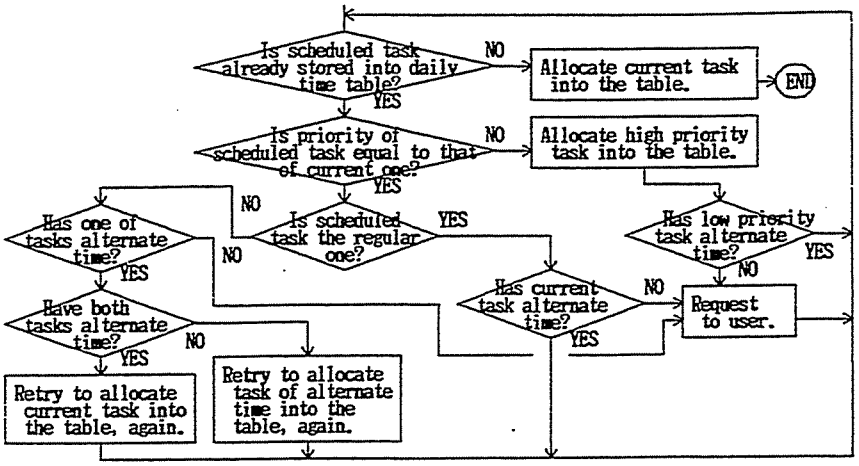
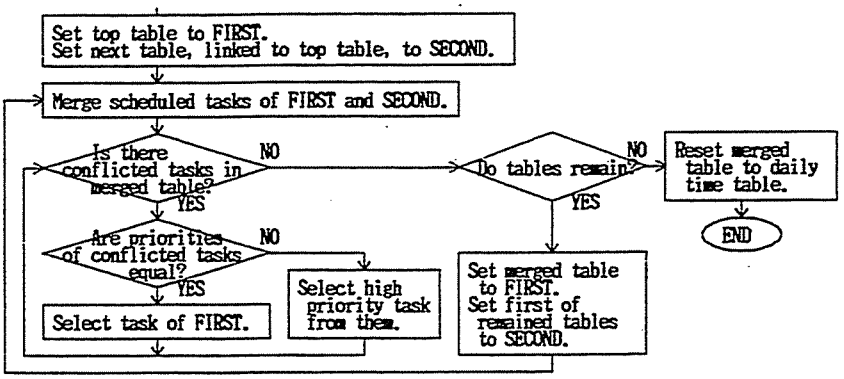
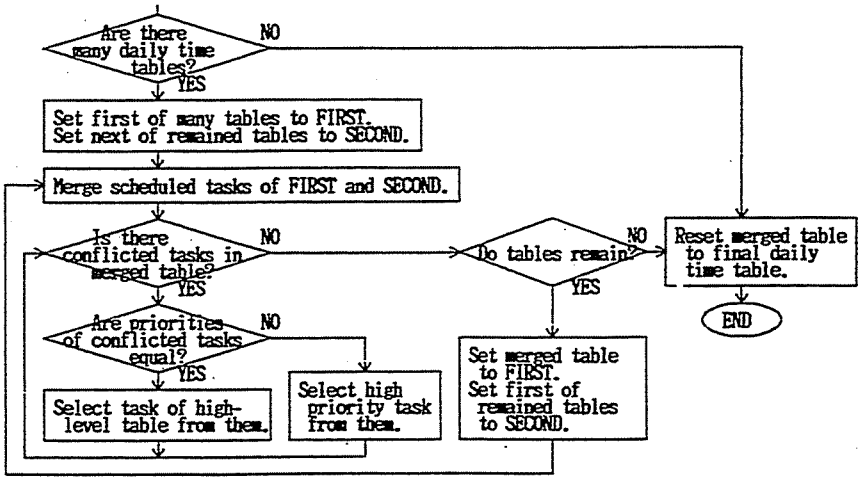


Fig.7 Allocation of tasks in each daily time table



(a) table reduction for each path



(b) table reduction for tables

Fig.8 Composition of an effective daily time table in the hierarchical structure

assigned time position has been already occupied by another existing task, the task confliction problem occurres. Fig.7 shows the resolution algorithm for the confliction problem between the current task and the scheduled task. Although our resolution strategy depends mainly on the relationships among 2 priorities and the assignments of alternative times, the direct manipulations of users are required when the resolution process is failure.

(2) composition of an effective daily time table in the hierarchical structure

The scheduled tasks are distributed over individual daily time tables in the hierarchical structure. Of course, there are many pathes from the top environment to the agent through the multiple inheritance mechanism. Therefore, the composition process of an effective daily time table becomes equally a graph reduction problem along multiple pathes. The reduction procedure is divided into 2 steps: the reduction along pathes; and the reduction for a table set, derived from the hierarchically reduced tables. In the first step, the resolution process for conflicted tasks is mainly dependent on the priority as the first parameter, and the level in the hierarchical structure as the second parameter. Namely, the scheduled task in the upper level is prior to that in the lower level. This is because in the hierarchical organization the global rules control generally the local rules. In the second step, the selection process for conflicted tasks is executable on the basis of the same parameters as the first step. These processes are shown in Fig.8.

8. CONCLUDING REMARKS

In this paper, we addressed an architectural framework of the scheduling system for daily personal tasks. In particular, our objective is to manage effectively the daily schedules about the activities of persons, who must interact to various kinds of organizations and/or have contacts with many groups/co-persons. Thus, we applied the object-oriented paradigm to our task scheduling system with a view to modeling the following issues successfully:

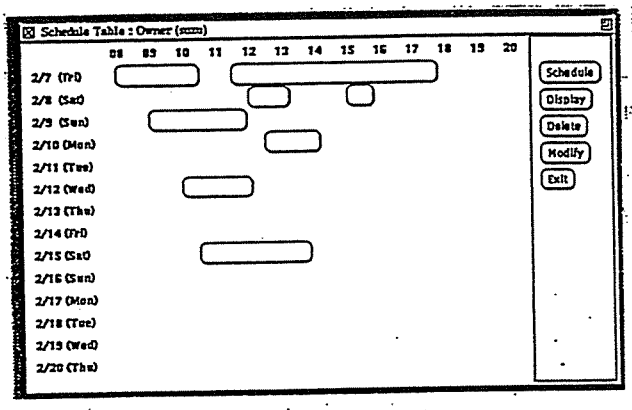


Fig.9 An example of scheduled time table

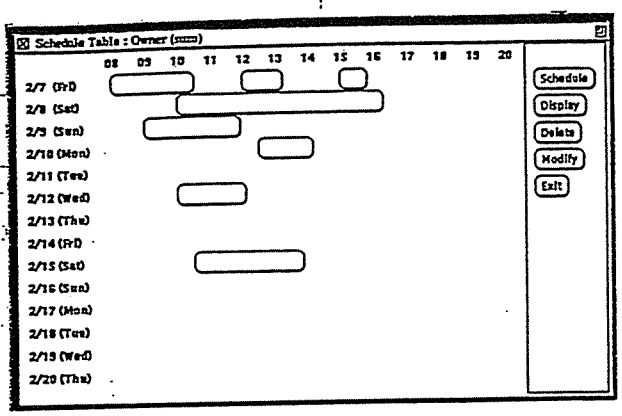


Fig.10 An example of rescheduled time table

(1) Individual entities have their own schedules when persons and their organizations are looked on as different entities;
(2) Persons can often perform the alternative task soon when the existing task was canceled;
(3) The activities of persons in some organizations are, more or less, restricted by the schedules of their organizations without depending on persons' schedules;
(4) Persons have their own selection abilities in case that several tasks conflicted simultaneously.

Our task scheduling system must manipulate active objects, by themselves, to be interactive concurrently. However, the relationship ISA is insufficient to control dynamic relationships between persons and organizations. Some organizations are temporarily generated by some persons. In our framework, every object is an instance as a person (agent) or an organization (environment). Thus, we introduced the relationship PartOf (for organizations) and MemberOf (for an organization and agents). Now, we have been developing a prototype of the task scheduling system on UNIX. For example, in Fig.9 the scheduled tasks about the daily time table is shown. Additionally, after having allocated a new task the status is illustrated in Fig.10.

We have some future work. The most important subject is an improvement of the selection process for conflicted tasks and a development of the cooperative message-exchange facility among related agents. The current version is designed too loosely to deal with such complex processings. However, it is provable that our framework is very intelligent to support a secretary facility in office information systems[2, 5].

Acknowledgements --- The authors are very grateful to thank Prof.N.SUGIE, Prof.Y.INAGAKI and Prof.J.TORIWAKI of Nagoya University for their continuous suggestions, and also would like to thank Mr.H.SUZUKI and our research members for their eager discussions.

References

1) M.Weiss, Z.Fang, C.R.Morgan & P.Belmont: "Effective Dynamic Scheduling and Memory Management on Parallel Processing Systems", Proc.of COMPSAC'89, pp.122-129(1989).
2) M.Chen, J.Chang & K.Lin: "Scheduling Algorithms for Coalesced Jobs in Real Time Systems", Proc.of COMPSAC'89, pp. 143-150(1989).
3) K.Sugihara, T.Kikuno & N.Yoshida: "A Meeting Scheduler for Office Automation", IEEE trans.on Software Engineering, Vol. 15, No.10, pp.1141-1146(1989).
4) E.G.Coffman, Jr. & P.J.Denning: "Operating Systems Theory", Prentice-Hall, Inc., NJ(1973).
5) I.Marui, M.Ichikawa & M.Tokoro: "An Organizational Model of Computation and its Application", trans.of IPSJ, Vol.31, No. 12, pp.1768-1779(1990) (in Japanese).