7. DESIGN AND DEVELOPMENT OF PROTOTYPE

7.1 INTRODUCTION

Designing a multi-agent system to manage project scheduling involves the systematic transformation of description of that project scheduling into a number of tasks and coordinating software agents. The focus of this work is on the interconnected problems of the resource allocation and decision-making about the dynamics of the environments the agents operates in. The allocation of resources is done in a single step: the resources are distributed among the agents and no re-allocation of resources is allowed during the plan execution phase. When viewed primarily from the resource allocation perspective, this work can be characterised as development of mechanisms for resource allocation where the utility values of the agents to resources are defined by the agents’ planning problems. Alternatively, if the planning problem is placed at the forefront, the work can be described as resource allocation between competitive agents where the interactions among agents are defined by the resources that are being distributed among them. Detailed coordination mechanism, steps for handling scheduling and coordination among agents are provided based on the resource objects coordination formalisms.

A multi-agent system approach, which is presented, is an implementation of project scheduling where activities, resources and important functions are represented as agents. This is implemented as a distributed collection of agents and the agents are allowed to execute tasks based on the event-control-action rule. Instead of the traditional use of schedules, control policies have been developed in the form of
planned resource allocation to tasks that capture the dependencies associated with task execution and the impact of resource allocation on those executions.

7.2 PROTOTYPE ARCHITECTURE

Multi-agent system is adopted as an infrastructure to support project management and the system is a distributed decision-making support system for coordinating the activities. The primary phase of this coordination involves gathering the agents into appropriate groups to carry out the tasks to be completed in the resource constrained environment. These agents interact with each other in a changing hierarchical order and the developed objects have life cycles that vary according to their importance in the coordination system and the task requirements. Each activity and resources needed in a project are represented as agents and the resource agents and task agents are responsible for the resources and activities respectively. The constraint management agent receives information from resource require/release agent and this information is used by the resource allocation agent. Based on the information, the partial order is produced by the schedule generation agent and the functions of project scheduling are taken by the scheduling agent.

Scheduling agent, an important module of the developed prototype realises most of the project management functions, schedules the activities through computation, optimises the project makespan, and determines the total project duration. Scheduling agent is also a computation agent which schedules the activities and minimises the project makespan. The prototype system includes a core agent system made up of multiple agents and a Graphical User Interface (GUI) which
provides the possibility to control and supervise the system. Figure 7.1 shows the access hierarchy of the prototype system.

![Figure 7.1 The access hierarchy in the prototype system]

The prototype integrates the functions of project management and administration of task agents and resource agents. The scheduling agent creates instances of task agents and the resource management agent has the similar functions such as resource dependency identification, resource release and resource requirement. The scheduling agent sends out task agents to carry out tasks. From the user interface, one can get an overview of the project and gather the status of the project. Each agent is defined as a set of well-defined software objects, and the communication between them is allowed by well-defined rules. Furthermore, each agent is internally designed as a multi-layer service, where each layer provides a refined service to its superior layer. The system is defined by the following rules: System = Objects + Coordination, Object = Attributes + Operations. These features
are gained through the integration of object orientation technique and task design. The information object coordination formalisms are employed which have a high expressive power, as they possess the coordination capabilities of both the resource information and the task information. Thus, these information coordination objects allow users to use this system as a collection of high level objects, which encapsulate their multi-threaded control structures and coordinate through a request / reply mechanism.

The objects are equipped with attributes and a function for processing these attributes and it is also equipped with a control structure that defines all its potential behaviours. This control structure is described by event-control-action rule which allows an object to have several concurrently ongoing activities; transitions occur one by one, but these occurrences belong to different activities so that these activities progress in turn. As a result, each object is able to concurrently: ① process requests, ② wait for the reply of pending requests and ③ run its own activity, as an autonomous entity. Classes are organized in accordance with several inheritance relationships that authorise the polymorphism and sharing of items. The definition of classes requires a sequential object oriented programming language such as Java to deal with their attributes processing aspect. Java is used to write the code of objects that appear in various parts of the control structure and it is used to write the additional code, which includes the definition of types, functions and constants that are shared by the objects.

The focus, in the area of the resource allocation, is on developing mechanisms that distribute the resources among the agents in desirable ways, by having the agents’ preferences, over set of resources. In such problems, the characteristics of the agents’ utility functions often have a significant role on the properties of the resource
allocation problem. Rules based on the resource allocation systems are quite popular in cooperative problem-solving environment, which is introduced for the resource allocation. In this method, the utility value is based on the time period of the resource allocated to the agent in comparison with other tasks and the agents are allocated with utility values based on time period, which is used for having access to resources. The priority of each agent depends on the resource demand and the priority value that other agents place on the resource at the time of usage.

In the resource management, the information on the state of availability of the resources and utilisation are collected and this information is needed to make scheduling decisions. When all the control and resource flow dependencies of a task are satisfied, a task becomes ready to be started. It is at this point that the scheduler should act and decide the scheduling of a task to execute. The basic scheduling mechanism is complicated because of the priorities for different tasks being incorporated by the scheduling agent. ECA rule is employed based on resource requirement and it is used as the description of task execution sequence. The form of an ECA rule is shown in Figure 7.2.

\begin{verbatim}
| ON | Event | e |
| WITH | Con | c |
| DO | Action | a |
\end{verbatim}

\textbf{Figure 7.2 ECA form}

When event $e$ occurs and the condition $c$ is satisfied, the action $a$ is taken. In addition to this, a new event type called SAT (T$_i$) is added in the ECA form. SAT (T$_i$) event means all the constraints (precedence and resource) imposed on task T$_i$ is satisfied according to the time estimation and the task is started after this SAT (T$_i$)
event occurs. Activity priority is the priority of the activities specified and it is used to give importance to the activities of the project. Further, the activities on the critical path are automatically given a relative priority increase in order to guarantee that the scheduling agent attempts to reach the optimal shortest completion time of the project.

Having the ability to describe resource classes and their membership also facilitates improvements in the execution of activities. Each resource is represented as a set of attribute value pairs and there is a small set of predefined attributes. The required predefined attributes of a resource are: a name, a textual description of the resource and a set of criteria assertions. Resource instances have availability attributes and these attributes track use of resources indicating the time period of a resource that is available for use. Once a resource is allocated, it is necessary to maintain to which specific task or entity it has been allocated to and this information allows easy manipulation of allocated resources when the need arises. This information is captured as attributes of resource instances. Here, on allocation of a resource, two attributes must be set: 1 allocated to-this holds a reference to the specific task to which this resource is allocated and 2 time allocated-the time period of this resource allocated for the task. A resource class is defined as a set of criteria assertions and the criteria assertion is a boolean expression over the attribute values of the resource. These assertions serve as membership criteria for all the resource classes or instances of this class. If a resource is allocated, it can be made available again for other agents to use by releasing it. A resource is released on the completion of an activity for which it is allocated and a resource is also released if the duration for which it is allocated expires. The resource specification attached enable the identification; allocation and release of resource instances that meet the needs. The task agents can communicate directly with the resource agent when required. An agent might be in
the best position to select which specific resource to allocate or might want to release resource rather than waiting for the entire process to complete. The entire functionality of the resource management is thus available to agents to refine the use of resources within tasks. The resource taken for the process is indivisible resource and it can be assigned to only one task at a time. To realise the system, the time period and time points (when the tasks requires / releases resources) are given in Table 7.1 for the example discussed in section 3.5.

Table 7.1 Tasks and their attributes for the project described in Figure 3.3

<table>
<thead>
<tr>
<th>Task Name (T)</th>
<th>Duration</th>
<th>Predecessor (i)</th>
<th>Relationship Type with Predecessor</th>
<th>Successor (i)</th>
<th>Time Period of Resource Usage (tus)</th>
<th>Time point of Resource Required after starting</th>
<th>Time point of Resource Release before completion</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>T₂</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>R₁</td>
</tr>
<tr>
<td>T₂</td>
<td>6</td>
<td>T₁</td>
<td>SS</td>
<td>T₃,T₄,T₅</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>R₁</td>
</tr>
<tr>
<td>T₃</td>
<td>3</td>
<td>T₂</td>
<td>SF</td>
<td>T₆,T₁₀</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>R₁</td>
</tr>
<tr>
<td>T₄</td>
<td>2</td>
<td>T₂</td>
<td>FS</td>
<td>T₆</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>R₁</td>
</tr>
<tr>
<td>T₅</td>
<td>3</td>
<td>T₂</td>
<td>SF</td>
<td>T₆</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>R₁</td>
</tr>
<tr>
<td>T₆</td>
<td>3</td>
<td>T₃,T₄,T₅</td>
<td>SF,FS,FF</td>
<td>-</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>R₁</td>
</tr>
<tr>
<td>T₇</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>T₅,T₆,T₁₀</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>R₂</td>
</tr>
<tr>
<td>T₈</td>
<td>4</td>
<td>T₇</td>
<td>SS</td>
<td>T₁₁</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>R₂</td>
</tr>
<tr>
<td>T₉</td>
<td>4</td>
<td>T₇</td>
<td>SF</td>
<td>T₁₁</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>R₂</td>
</tr>
<tr>
<td>T₁₀</td>
<td>3</td>
<td>T₃,T₇</td>
<td>FS,SF</td>
<td>T₁₁</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>R₂</td>
</tr>
<tr>
<td>T₁₁</td>
<td>3</td>
<td>T₅,T₆,T₁₀</td>
<td>SF,FS,SS</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>R₂</td>
</tr>
</tbody>
</table>

The challenge is the allocation of resources to the tasks to be performed which are having the same priority. This means that resources are indivisible and are allocated to task once at a time and this has been achieved by the results received from the game and this allows only one task to use a resource at a time. An allocation policy defines how resources are allocated to tasks for execution and this is done using tricky game or by choosing a task based on resource allocation game. The utility value
is used in the game for resource allocation decision. If the two tasks are having the same utility value (same priority), then tie occurs and the problem of allocation is solved by breaking the tie using resource allocation game to obtain a feasible solution.

7.3 AGENTS IN THE SYSTEM

7.3.1 User interface agent

This agent offers an interface between a user and the agents. The user gives a high level request to what to do through the interface. Tasks are received into a set for scheduling and the tasks are called scheduling events. Each agent is an entity which acts as a coordination expert on a particular aspect of project scheduling and has the capabilities of problem solving and conflict resolution. The graphical interfaces with multiple independent software agents provide information to the user while performing a project related tasks.

The user interface screen for this agent is designed and developed and this agent receives all the information about the project as shown in Figure 7.3. The details such as tasks, durations, predecessors, relation type, successors, and the resources needed with time points and time periods are received and stored in separate project name. The save option is used to store the task details in a unique project name. The clear all and exit options are used to clear the displayed details and to switch to the scheduling agent screen respectively. The displayed information is removed or stored by selecting the corresponding options and the exit option is used to move the user control from this agent to the scheduling agent screen. The resources, durations, resource usage time periods to be used to execute the tasks are offered to the tasks as attributes. To support resource constraint driven execution semantics, the
resources and the time points that the tasks require or release them are also specified. The above said information such as set of tasks and set of resources are given to the task agent and resource agent through the update option.

Figure 7.3 User interface agent screen for the sample project

7.3.2 Task agent

A task agent is expressed by the inputs and outputs of the task. When the input conditions of task are satisfied, this task agent is enabled and produces outputs after its action or after its partial action. Task agent receives the resources as input and releases the resources as output. Input and output resources are used in determining the sequences of tasks with dependent_on_for( ) function. To enable resource transfers between tasks, each task has its own input and output parameter lists. When a task becomes ready for execution, resource is copied into its input parameters from other tasks' output parameters. When the tasks' execution is finished, resource is
copied out of its output parameters into the process. These mechanisms allow the specification of the wide variety of resource exchanges between tasks. A task is in the initial state when it is first created. Once the program start event has been acknowledged, execution of a task begins. A task becomes execution once the program start event has been acknowledged.

When the requirements of a project is broadcasted, the task agents whose output resources match or partially match the requirements register themselves at the scheduling if their input resources are satisfied. In order to satisfy the input resources, the task agents search for the agents whose output resources provide the whole of its input resources. In this way, the sequencing of activities is determined by coordination among agents. Figure 7.4 illustrates the way of scheduling tasks with and without considering the early release of resource. Task agent executes all the tasks by initiating one after the other or by activating the task earlier and it lists all the tasks which are executed with their time units (tus). Task agent screen displays the tasks along with the resources that are being consumed (Figure 7.5).

![Figure 7.4 Scheduling of tasks](image-url)
7.3.3 Resource agent

Each resource is represented by an agent and it is assumed that any resource can be utilised by any task that requires it and the properties of resource and the utilisation status are recorded. Resource agent is controlled by sending its information about the status of the resource such as initial, intermediate, final etc. By keeping the record of utilisation requirements, it is easy to find the implicit dependency of resources and the dependency of a task on a resource is known. But the interdependence between two tasks using a common resource must be informed. The responsibility of the resource agent is to provide information about the types of availability of resources and to track their usage. To identify all resources that meet a particular need, the task which requires the resource specifies a resource class name and the attributes of the class. The resource agent finds all instances from the named resource class and the identified resource is allocated to the requested task and then it
is locked for use by the given task and the resource status information is obtained from the resource agent.

7.3.4 Constraint management agent

The constraint management has functions which are distinguished as determine about resource requirement, determine about resource release, determine priority and determine coordination. The function determine resource require identifies the resources that are required by the activity and stores the information on resources and this information is obtained from the dependency information. The function determine resource release stores the information about when to release the resource. The tasks of the function determine priority is to determine which of two agents may access the resource first by using the priority rule. The determine coordination function contains information about coordinated resource access to avoid conflicts. The constraint management agent extracts the resource constraints from the definition according to the task dependencies and the specified resource, especially the resource require and release setting and this constraint management agent manages these resource constraints.

To support resource constraint driven execution semantics, the resource information and the time points that the tasks require or release them are monitored by using the event-control-action rule. This agent extracts the resource constraints according to the task dependencies and the specified information, especially the resource require and release setting. The SAT (Tj) event is an event to initiate a task execution and it is viewed as a schedule event and the SAT (Tj) event is created and sent by the constraint management agent. When a SAT (Tj) event is created and sent, it means that some initial constraints have been already satisfied and others will be
satisfied according to the time. So, for the constraint management, one of its functionality is to find the appropriate time point to initiate the task $T_i$. When there exists a task resource constraint $(T_j, T_i, r, t_{req}, t_{rel})$ imposed on task $T_j$ where $t_{req} \in T_j$ and $t_{rel} \in T_i$, the ECA rule is used to start the task $T_i$ and the partial overlap is obtained. Another new event type called END ($T_i$) is added in the ECA form and the task agent uses the END ($T_i$) events as well as the SAT ($T_i$) events from the constraint management agent. The scheduling agent uses those events to interpret the ECA rules and initiate the scheduling of a task.

![Constraint management agent screen](image)

Figure 7.6 Constraint management agent screen

Constraint management agent screen for the sample project is given in Figure 7.6. During scheduling, a task is executed and the scheduling agent uses and updates (including release) the relevant resource (including sharing of resource) and the constraint management agent monitors this updated resource information. When some related resource information is released (the state of the resource information is
said to be RELEASED), this agent checks the resource constraints that are related to this resource information. Then, the related constraints are said to be satisfied if the required resource is available. This agent calculates all the constraints implied on related task and determines whether it is time to release a SAT (T_i) event and sets it to the scheduling agent to initiate a task.

### 7.3.5 Scheduling agent

The tasks of project management such as determination of the critical path and task scheduling are taken by scheduling agent. Using a serial schedule generating scheme, priority values are then used to construct an active schedule by scheduling each task one-at-a-time and as soon as possible within the precedence and resource constraints. The scheduling is done using the schedule generation schemes and it is a priority rule based method and a task is put on the schedule if its precedence and resource requirements are met. If the optimum rule is selected, the schedule generation agent produces the different possible schedules based on all the four priority rules and the optimum project makespan is selected. Otherwise, the schedule generation agent produces an activity list based on the selected priority rule and the list is stored in task-in-queue. Then the scheduling agent puts the task in the schedule if the task initiate event is received from the constraint management agent. The details of the scheduling process and the scheduling agent is shown in Figure 7.7.
7.3.6 Resource allocation agent

Resource allocation is a typical project management problem and in traditional research, resource conflicts are created whenever the total resource requirement of tasks to be scheduled in a particular time period exceeds the resource available during that period. Here, the resource conflict is defined whenever one resource is required by several branching tasks simultaneously. This conflict is detected by comparing the resource require / release time points because resource agents keep a record of each utilisation requirement. Otherwise, it is detected by the resource usage time periods.

A resource is defined as a collection of resource classes and resource instances and a resource instance represents a unique entity in the environment and a resource class represents a set of resources (or classes) that have common properties. The task priorities are obtained from the duration of the tasks already on the schedule.
Resource-level utility functions from multiple tasks are sent to a resource allocation agent, which computes allocation of resources across the tasks. This is done by obtaining the time period of resource usage from each task. When the priorities of two tasks are same, then the resource allocation agent plays resource allocation game. Based on the outcome of the game, the resource is allocated to the agent (task) who wins.

7.3.7 Service agent

Service agent screen displays the project duration for critical path and overlapped path after scheduling the activities on the critical path. The service agent screen for the sample project is given in Figure 7.8.

![Service agent screen](image)

**Figure 7.8 Service agent screen for project makespan**

This agent also displays the list of activities in the critical path before displaying the critical path and overlapped project durations. From this service agent
screen, one can get an overview of the whole project and gather the status of the project and this agent functions are normally viewed after the scheduling agent schedules all the activities in the project.

From the given schedule displayed in the service agent screen, task $T_1$ and $T_7$ are put on the schedule first. $T_1$ releases the resources $R_1$ during 2 time units before completion and $T_2$ requires the resource after 2 time units and hence overlapped. The overlapping period is 4 and to total duration after first overlap is 10 time units.

![Figure 7.9 Project duration for the overlapped schedule](image)

Similarly $T_7$ releases the resource 2 time units before completion and $T_8$ requires the resources at the beginning and the overlapping period is 2. Similarly $T_8$
and $T_9$ are overlapped and then $T_3$ and $T_4$ are overlapped and so on. The total project makespan is 18 time units which uses the resource $R_1$ and also the project makespan is 18 time units which uses the resource $R_2$. The project duration is reduced from 25 to 18 for the sample project (Figure 7.9) which is a good solution in view of minimising the project makespan.

### 7.4 DISCUSSION

A multi-agent prototype system has been defined and developed to test and verify the project schedule optimisation and to provide a foundation for facilitating coordination among software agents involved in project scheduling. The prototype is consistent with the object oriented paradigm and is implemented using a standard object oriented language. The developed multi-agent system is used for project scheduling with scared resources and tracking and monitoring of projects during scheduling is possible. Overlapping is concerned with shortening the project makespan not by shortening the individual activities, but by increasing the overlap period among them. The developed software agents of the prototype are stand-alone procedures that can be called interactively; they can be used as basic blocks on which a system can be built.

In traditional project management, a resource is occupied throughout the period of a task. In this research, after utilising a resource for the required duration, a task releases the resource even before completion of the task. Having the ability to describe resource classes, their membership also facilitates improvements in the execution of coordinated activities without requiring change to the activity specification.