CHAPTER 5

5. MULTI-AGENT COORDINATION

5.1 INTRODUCTION

Traditional problems of project scheduling have not adequately incorporated number of factors that are important for resource allocation in concurrent engineering environment. The model described here explains the effect of resource-availability constraints on project scheduling. Modelling the project schedule with precedence relations among activities and resource allocation on the planning phase remains important because it allows the schedule to estimate feasibility of the project and plan.

The work presents an optimisation scheduling approach for projects, in which activities are executed in an overlapping fashion based on the renewable as well as non-renewable resources. At first, an integration criterion function is proposed to ensure the optimal concurrent scheduling and the effective utilisation of resources. Besides, two cruxes in concurrent engineering: role allocation and pre-release of resources are introduced. This work attempts to find optimal or near optimal solutions to project schedule problems with resource and precedence constraints. Therefore, to achieve an integrated scheduling algorithm for concurrent activities, modelling of resources and task overlapping is done giving attention to the limitations on resources. This work derives three insights: ① description of overlapping activities; ② allocation of resources to activities; ③ integrated scheduling algorithm.
5.2 GOAL MODELLING

In Resource Constrained Project Scheduling Problem (RCPSP), tasks compete for resources and it is assumed that the constraints among the operations of a project can be described solely by precedence relations between tasks. The determination of valid start times for the tasks with respect to precedence and resource constraints is the solution of an RCPSP. The objective function is optimisation and typical optimisation criteria is minimising the makespan. The project is identified by the major sections of work and represented by way of a project goal, sub-goals, workflow tasks and detailed activities. This successive breakdown from overall project to detailed activities is shown in Figure 5.1 and modelling of work at the detailed level can then take place related to the particular parts of a project.

![Figure 5.1 Goal modeling (Stephenson and Armeson. 2002)](image)

The model is established indicating the different parts of a project with successive sub-divisions of identified sub-goals, workflow tasks and detailed activities relating to the project process. Moreover, the identified work tasks and activities form the basis for event modeling in relation to the actual sequence of activities. Besides
tasks, there is another type of goal which can be handled and a scheduling goal corresponds to a task to be solved by scheduling these tasks. Figure 5.2 states the relationship between goals, tasks and scheduling goals.

Coordination is the process by which individual efforts in an environment are exerted towards achieving a set of goals. By determining which of the tasks’ inputs have to be in place before work on it can be started, and by finding out which tasks produce the required parameters or features, the precedence relationship between the tasks is determined, and a plan can be stated. A plan is a partially ordered sequence of tasks; the order is enforced by the tasks’ input / output relationships.

**Agent model**

The tasks and sub-tasks can be represented through process model as shown in Figure 5.3 and the agents that carry out the process are defined in the agent model.

---

**Figure 5.2 Relationship between goals and tasks**

<table>
<thead>
<tr>
<th>Goal</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling goal</td>
<td>Duration</td>
</tr>
<tr>
<td>Start time</td>
<td>Inputs</td>
</tr>
<tr>
<td>End time</td>
<td>Outputs</td>
</tr>
<tr>
<td>Assigned agent</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Goal</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling goal</td>
<td>Duration</td>
</tr>
<tr>
<td>Start time</td>
<td>Inputs</td>
</tr>
<tr>
<td>End time</td>
<td>Outputs</td>
</tr>
<tr>
<td>Assigned agent</td>
<td></td>
</tr>
</tbody>
</table>
A Multi-Agent System (MAS) is a coordinated execution of tasks to carry out a process. MAS requires a coordination mechanism that facilitates dynamic collaboration of agents, with the goal of satisfying both local and global system objectives. The primary phase of multi-agent coordination involves gathering the agents into appropriate groups to carry out the process. Each higher-level task (process) in the system is divided into partial sub-tasks and distributed to the relevant agents. The group in turn generates a set of collectively agreed objectives through an agent interaction model to attain a common goal. The processes are carried out by a group of agents and each output is valuable in its own right and also helps to generate the next process.

The model allows the user to specify the goals to be achieved in a specific project. A task is a chore to be done to achieve a goal. In this model, task completion is represented as a type of goal with a duration that can be specified by the user. Tasks produce output which serves as input for other tasks. These input and output

Figure 5.3 Task modelling in MAS (Eswaramurthy et al. 2002)
are predicted, and have to be verified by comparing them to the actual inputs and outputs which are determined by making a decision for the task. Inputs and outputs are passed in the form of parameters or features, which can represent data as well as physical objects.

Tasks are executed by agents and resources are shared by tasks. An agent is a computer or software which provides a sophisticated notification system that keeps the agent up to date on all changes in the plan, schedule and design with which the agent is concerned. Agents and resources need to be scheduled to do the various tasks of the plan, by assigning them to tasks for a specified time interval. Each task takes up a certain percentage of an agent’s or a resource’s time. The sum of all the tasks scheduled for an agent (or resource) at a time is its utilisation. At times when an agent or resource is available, its utilisation will vary between 0% and 100%. The utilisation of an agent or resource that is unavailable must be 0%. By assigning an agent and the needed resources to each of the tasks of the plan, the timing of each task is determined and the schedule is created. The created schedule is similar to PERT plan and the critical path is identified.

5.3 PROBLEM DESCRIPTION

The objective of the resource-constrained allocation model is to minimise the project makespan when constrained by precedence relationships among project activities and the availability of resources. Resource requirements and processing times for each activity are deterministic and also known in advance and preemption of activities is not allowed. The following basic requirements are identified for modelling the project scheduling.
Basic assumptions

- The resources considered in this research are only human resources.

- Among the various heuristic resource allocation methods, priority rules based resource allocation is considered. The impact of multi agent coordination using priority rules based resource allocation on project time is considered.

- A project consists of different activities, which are represented, in the activity-on-the-node format. Without loss of generality generally, the activity 0 denotes the start of the project and is a predecessor of every other activity in the project, activity N+1 denotes the end activity of the project and is a successor of every other activity in the project.

- The activities are related by a set of finish-start, finish-finish, start-start and start-finish precedence relations with zero time-lag, implying that no activity can be started before all its predecessors have completed or partially completed and the activities are to be performed without preemption.

- The resources, agents, tasks relevant to the project are known and they may be either determined or approximated.

- A resource is released in the intermediate time point but not at the end of task execution and a resource is required in the intermediate time point but not in the beginning of a task execution.

- A resource can be allocated to a task based on the time period of resource requirement and it is converted into utility function value which can be used in the resource allocation game.

- The available resource assigned to each activity is known and fixed over the processing interval of the activity. Once a resource is assigned to an activity, the resource is used in single time interval (no pause is allowed).
• The objective is to complete the project as early as possible without violating any deterministic resource and precedence constraints, as well as uncertain resource constraints for the pre-given confidence level.

Based on the above requirements, the following problem is defined and this is used in resource allocation for project scheduling. The multi-agent resource-constrained resource allocation task scheduling problem is a tuple \( \langle T, A, R, F, U, R_r, R_i, P, H \rangle \) where,

- \( T = \{T_1, \ldots, T_n\} \) is a set of tasks;
- \( A = \{A_1, \ldots, A_m\} \) is a set of agents;
- \( R = \{R_1, \ldots, R_n\} \) is a set of resources;
- \( F: A \times T \rightarrow \{0,1\} \) determines which agents are involved in which tasks: given an agent \( A \in A \) and a task \( T \in T \) such that \( F(A, T) = 1 \) \( \Leftrightarrow \) agent \( A \) is required to perform task \( T \):
- \( U: R \times A \times T \rightarrow N^+ \cup \{0\} \) determines the resource usage attributes of the tasks: given \( R \in R \), \( A \in A \), \( T \in T \) such that \( U(R, A, T) = t \) \( \Leftrightarrow \) agent \( A \) uses a resource \( R \) for \( t \) time units to execute task \( T \);
- \( R_r: R \times A \times T \rightarrow N^+ \cup \{0\} \) determines the time point of a resource required for the tasks: given \( R \in R_r \), \( A \in A \), \( T \in T \) such that \( R_r(R, A, T) = t_{req} \) \( \Leftrightarrow \) agent \( A \) required the resource \( R \) a time point \( t_{req} \) to execute task \( T \);
- \( R_i: R \times A \times T \rightarrow N^+ \cup \{0\} \) determines the time point of a resource released for the tasks: given \( R \in R_i \), \( A \in A \), \( T \in T \) such that \( R_i(R, A, T) = t_{rel} \) \( \Leftrightarrow \) agent \( A \) releases the resource \( R \) a time point \( t_{rel} \) to execute task \( T \).
P is a set of precedence relations between any two tasks: given two tasks Ti, Tj ∈ T such that Ti < Tj ⇔ Ti must occur before Tj and H is a set of pairs of tasks indicating precedence relations of the type SSy, SFy, FSy and FFy.

There are set of activities N = {0, 1,..., n} which is to be scheduled on a number of resource types K with availability a_k, k = 1,..., K ; N is referred to as a project. Activity i has duration d_i ∈ N and requires resource type k. Activities 0 and n have zero duration and zero resource usage; Also d_i > 0 for i ≠ 0 and n. A is a partial order on N, i.e. an irreflexive and transitive relation, which represents technological precedence constraints. Dummy activities 0 and n represent start and end of the project, respectively: ∀i ∈ N \ {0}: (0, i) ∈ A and ∀i ∈ N \ {n}: (i, n) ∈ A. The use of resource flows has been advanced by various sources, among which the word flow mostly refers to a resource flow. A flow f assigns to each triple (i, j, k) ∈ N x N x K a value f(i,j,k) ∈ N. These values must satisfy the following constraints, which constitute flow conversation constraints and lower and upper bounds on flow through intermediate (non-dummy nodes):

$$\sum_{j \in N} f(j, i, k) = \sum_{j \in N} f(i, j, k) = A \quad \forall i \in N \setminus \{0, n\}, \forall k \in K$$

Flow quantity a_k (the availability of resource type k ∈ K) is sent into the network from the dummy start node and collected at the end node:

$$\sum_{j \in N} f(0, j, k) = \sum_{j \in N} f(j, n, k) = 1 \quad \forall k \in K$$

Each resource type has its own distinct capacity and for a flow f, the set of activity pairs Φ(f) = {(i, j) ∈ N x N: f(i, j, k) > 0 at least one k ∈ K}, containing the arcs
carries non-zero flow. Also $C(f) = \Phi(f) \setminus A$: the arcs in $C(f)$ are the flow carrying arcs that do not coincide with technological precedence constraints.

5.4 MATHEMATICAL MODEL

The resource constraint is one of the major constraints that directly impacts on the task execution and another type of constraints is precedence constraint. The resource constraints are introduced that cause the resource dependencies among the task execution of the project and the following functions are designed to manage task dependencies.

**Resource require:** If an execution of the task needs resource $r$, say task $T$ requires resource $r$. This can be represented as $\text{require} (T, r)$ and it is used to find whether the task $T$ requires a resource $r$ or not.

**Resource release:** If an execution of task $T$ releases a resource $r$, say resource $r$ is released by $T$. This relation is written as $\text{release} (T, r)$ and it is used to find whether the task $T$ releases a resource $r$ or not.

**Task dependency based on resource:** If task $T_2$ requires a resource $r$ that is released by Task $T_1$, say there is a resource dependency between task $T_1$ and $T_2$, and this can be written as $\text{depend}_\text{on}_\text{for} (T_2, T_1, r)$. It means that the execution of $T_2$ depends on the execution of $T_1$ because of resource $r$.

By using the three functions the following constraint is formed and this constraint is used to group the activities based on a particular resource.

$$\text{require} (T_2, r) \land \text{release} (T_1, r) \rightarrow \text{depend}_\text{on}_\text{for} (T_2, T_1, r).$$
To meet the objective of this work, two more functions namely \textit{require} \((T, r, t_{req})\) and \textit{release} \((T, r, t_{rel})\) are designed.

**Resource require at time point:** If the execution of the Task \(T\) requires a resource \(r\) at time point \(t_{req}\), it means that task \(T\) requires the resource \(r\) at the intermediate time point \(t_{req}\) after the starting of the execution. This can be represented as \textit{require} \((T, r, t_{req})\).

**Resource release at time point:** If the execution of a task \(T\) releases a resource \(r\) at time point \(t_{rel}\), it means that task \(T\) releases the resource \(r\) at the intermediate time point \(t_{rel}\) after starting the execution. This can be represented as \textit{release} \((T, r, t_{rel})\).

**Resource constraint:** If task \(T_1\), task \(T_2\), resource \(r\) and time point \(t_{req}\) and \(t_{rel}\) are given, then there is a resource constraint if \textit{require} \((T_2, r, t_{req})\) and \textit{release} \((T_1, r, t_{rel})\). This is a resource constraint on task \(T_2\) from task \(T_1\), which can be represented as \textit{constraint} \((T_2, T_1, r, t_{req}, t_{rel})\).

**Constraint satisfaction**

From the above, the following resource constraint is formed which is then used to identify the task execution overlaps. The resource constraint \textit{constraint} \((T_2, T_1, r, t_{req}, t_{rel})\) is satisfied, if exists \textit{require} \((T_2, r, t_{req})\), \textit{release} \((T_1, r, t_{rel})\) and \(t_{req} < t_{rel}\).

A network \(G = (N, E)\) comprises a set of activities (node) \(N = \{T_1, T_2, ..., T_n\}\) and a set of edges \(E = \{(i,j) \mid T_i, T_j \in N\}\) and representing the precedence relation. A schedule of a project is mapping a set of activities to meet the activity precedence relations and resource requirements. What information is needed to assign task priorities in priority scheduling? The resources are to assign task periods, execution times and constraints on response times. In addition, currently assigned task priorities
need to be known too and each task is defined by a set of parameters: resource requirements, task duration, time point of resource require, time point of resource release, time period of resource usage and the relationship with other tasks. These are the minimal information needed for making effective scheduling decisions.

If there is at least one precedence constraint in task set, then it is referred as a set of dependent tasks. Otherwise, it is a set of independent tasks. Depending on the extent of dependency between activities, the relation between activities can be categorised into two types: independent and dependent. In a dependent relation, one activity requires resource from other activity and in an independent relation, each activity does not require any resource from other activity.

5.5 MODELLING

A project is specified as a finite set of activities that require the resources for the processing. For each activity, the user specifies its duration as a positive integer number. Let $D_i$ denotes the duration of the activity $i$ and the activity requires the resource from its start time till its partial completion time and no other activity can be processed at that time and the resources are often called as unary or disjunctive resources. Resource capacity constraints ensure that the total of the resource requirements of the tasks processed concurrently never exceeds the available capacity. A resource capacity constraint is always responsible for one resource $r$, and concerns all the tasks which require this resource. Resources with one unit capacity are called unary resources (resources with $q(r) = 1$) and they are often distinguished from cumulative resources ($q(r) > 1$).
There are some possible situations for defining an appropriate model and these must matter on the knowledge dispose, one versus the other, the agents that interact one to other and a set of agents' shares resources. The resource is constituted of an integer amount of units. The agreement is sought that all the agents will be able to use the resource and an agreement is a schedule that allows the usage of the resource among the agents. Sharing a common resource requires a coordination mechanism that will manage the usage of the resource. It is obvious to outline that in real systems there are more than one resource that is used in a time instant by an agent. Also it must be fixed that between the agents exist different and conflicting goals. From these considerations the mechanisms used for resource allocation should be stable and symmetric. Two sets of decisions, resource allocation and task scheduling are made simultaneously in this model. The decision of resource allocation to tasks is made based on resource-constraint and the decision of task scheduling is made based on the precedence-constraints.

5.5.1 Task modelling

A resource is renewable and indivisible and it is made available to only one task at a time. The resource is released from a task immediately when its usage is over before the completion of the task. Then, the resource status condition is changed and this resource is available to another task. Task agents and resource agents represent activities and resources in a project. Scheduling agents are automated experts for special project management tasks. The tasks are also prioritised, dependencies between tasks are identified, and this information is made available during project schedule. The dependencies between the tasks can affect the length of the overall project (dependency-constrained), because of the availability of resources (resource-
constrained). A task can begin as soon as the last of its predecessors are finished if there are no limitations on resources. An integrated criterion function is proposed to deal with the scheduling problem under limited resources.

**Sequencing of tasks**

Task decomposition includes task hierarchy, the information about resource requirements a task requires and the resource released after the task execution. Task sequencing is explicitly modeled within components as task control. Task control includes not only the information of which tasks should be activated when and how, but also information of the goals associated with task activation. Also, it determines which information is needed to decide whether access to a resource is allowed and recognises the need to access a limited resource and activates the component cooperation management to determine which access is allowed or not.

**Resource exchange between tasks**

Resource exchange between tasks is specified as resource information links. Each resource link relates output of one component to input of another. The main entities of the process representation are tasks and resources. The order of the execution of the various tasks is specified by control flow dependencies among them, while the resources exchanged between the various tasks are modeled by a resource flow relationship between the tasks. The event-control-action mechanism is employed to specify control flow. Timing constructs are incorporated in the task model in order to provide important information regarding the time within which an agent needs to start its work after being assigned to a task, and the maximum time the agent is given
to complete the task. An approach has been proposed for the assignment of resources to tasks. Modelling concurrent tasks requires a good understanding of the relations of the various concurrent activities of the project. The model is based on the PERT project model which describes the concurrency relationships that contain the sequencing of tasks as the effects of resource allocation to tasks. This model identifies a process structure that can be used to describe the completion of artifacts in each task.

### 5.5.2 Task overlapping

Consider tasks $T_i$ and $T_j$ of length $t_i$ and $t_j$ respectively. The predecessor and the successor activities are usually called as upstream and downstream activities respectively. Denote the first activity is upstream and the second as downstream. In the "over the wall" approach, total completion time is $t_i + t_j$ and the objective is to minimise the total completion time. The proportion $\lambda$ (\% of downstream task length) of $T_j$ which is to be scheduled in parallel to $T_i$ is positive. If $T_j \leq T_i$, no more than $T_j / T_i$ can be paralleled. Let $A = \{0, \lambda_{\text{max}}\}$, where $\lambda_{\text{max}} = \min \{1, T_i / T_j\}$, be the interval of all possible levels of overlapping and $\lambda$ provides a continuous measure of concurrency. Without taking into account any drawbacks of concurrency, the project completion time $T$ benefits from overlapping is

$$T = \sum T_i + (1 - \lambda)T_j \quad \text{where } i, j \in [1, N]$$

In a fully sequential process, downstream starts with resource release from upstream whereas in an overlapping process it has to rely on pre-release of resource.
5.5.3 Resource modelling and allocation

Resource model is used to represent plans and actions. The goal situation is to have a set of resources of a particular resource type and the initial solution is a set of resources. Agents can perform actions that require resources and release resources. A plan is represented as a directed acyclic graph where the vertices are tasks and the arcs connect resources with tasks to indicate either a require or release relationship uses a resource. Activity sequencing involves identification of logical relationships and activities are sequenced accurately to support later development of a realistic and achievable schedule. Activity duration in estimating the number of work periods required to complete an activity will often require consideration of elapsed time as well and resource requirements are a description of what types of resources are required.

5.5.4 Solution strategies

A solution for the project scheduling problem is a set of activities (agent tasks) and resources necessary to satisfy the constraints (precedence and resource) given by the problem. A feasible solution has neither precedence nor resource conflicts, i.e. it satisfies both precedence and all resource constraints. For resource constraints to be satisfied, no resource over-allocation must occur with any task of the solution, in any agent, at any movement of the scheduling. For precedence constraint to be satisfied, the precedence relations between activities are preserved at any point in time. In this approach, two special sets of solutions are considered: the set of precedence-feasible solutions, which satisfies all precedence constraints and the set of resource-feasible solutions, which satisfies all resource constraints. As Figure 5.4 shows, the set of
feasible solutions is the intersection of those two sets, i.e., a feasible solution is one that is both precedence-feasible and resource-feasible.

Figure 5.4 Solutions of a scheduling problem

A problem is termed precedence over-constrained if the set of precedence-feasible solutions is empty and is termed resource over-constrained if the set of resource-feasible solutions is empty. If a problem has both non-empty precedence-feasible and resource-feasible solution sets and their interaction is a non-empty set, then feasible solutions are obtained.

5.5.5 Game model

In game theory, the optimal strategy is determined based on the interaction as a game between identical players and seeking its equilibrium. The strategy so determined is optional for an agent given the same rules, the fixed payoffs and the goals.
5.6 ACTIVITY AND RESOURCE STATE MAPS

The activity state map and resource state map are introduced which gives the state of an activity at any moment during the execution of a process instance. In planning, an approach is followed that brings tasks require the same set of resources into one group activity. This simple way of grouping is easily understood, where project planning is performed before the preparation of the detailed technological plans, based on estimates.

Activity state map

Let \( N \) be the set of all activities of a project and \( \mathbb{N} \) be the set of natural numbers representing the time axis for each process instance (\( 0 \in \mathbb{N} \) represents the point in time when the associated instance of the process model is created). The activity state map

\[
\omega : \mathbb{N} \times \mathbb{N} \rightarrow S
\]

associates at any point in time \( i \in \mathbb{N} \) each activity \( T \) with its actual state \( \omega (i, T) \), where \( S \) includes all activity states relevant to execution: initial, executable, active, ready to anticipate, anticipating, terminated, completed and suspended.

Activities in state ready to anticipate is a control flow strategy and activity \( T \) becomes ready to anticipate at time \( t \) if

1. \( \omega (i - 1, T) = \text{initial} \)

2. \( \sum T^+, \omega (i, x) \in \{\text{executing, anticipating}\} \) if \( A \) is an AND join node

\( \sum T^+, \omega(i, x) \in \{\text{executing, anticipating}\} \) if \( A \) is an OR join node
where \( T^{\sim} \) denotes the set of all direct predecessors of activity \( T \). An AND-join activity in initial state enters the ready to anticipate state when its predecessors are activated for execution or anticipation (executing or anticipating state). An OR-join activity enters the ready to anticipate state when one of its predecessors is activated for execution or anticipation (active or anticipating state). When a process instance is created, all activities are in the initial state (except the start activities that are directly executable). An activity enters the executable state as soon as the preceding ones have completed and its preconditions are fulfilled (Figure 5.5).

As soon as an activity becomes executable, it is scheduled and when it is chosen by an agent (or automatically), the activity becomes active. When its execution terminates normally, the activity reaches the completed state (final state) and an active activity can be suspended, i.e. quiescent until returned to the executable state, or terminated, i.e. stopped before its normal completion without the possibility to return to the executable state.

\begin{center}
\textbf{Figure 5.5 Activity states (Daniela et al. 2004)}
\end{center}
Resource state map

A deterministic resource allocation is considered and let $R$ be the set of all resources (resource needed as input by the activities, resource required by conditions and the resource to be exchanged between activities) with the $S$ that corresponds to the set of possible resource states: Every operation effects the state of the resource and the resource state map is,

$$\delta : \mathbb{N} \times R \rightarrow S$$

which associates at any point in time $i \in \mathbb{N}$ each resource $r$ with its actual state $\delta (i, r)$. The relevant states of a resource are initial, intermediate and final. The state of a resource contains all relevant information about it and an operation affects the state. An output resource of an activity enters the intermediate state when the activity executes; it enters the final state when activity completes. To specify the above model, the following rules are formed.

Rule 1: The normal activity is uninterrupted and it cannot be interrupted (paused) from the starting time until it is accomplished.

Rule 2: The required renewable resources will be released immediately when the task is completed or partially completed.

Rule 3: The schedulable activity will keep waiting until the resource requirement is satisfied.

Rule 4: The required resources of an activity are invariable during the performing process.
Priority rules

An activity’s resource utilisation is an important characteristic in the RCPSP and it is proved that successful priority rules incorporate the measure of time or resource usage. To suite the above, the following four priority rules are found and these rules are used to allocate the resources to tasks during schedule.

Table 5.1 List of new priority rules

<table>
<thead>
<tr>
<th>Priority rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_{req}TP (Resource require time point)</td>
<td>Select the available task based on the resource require time point where t_{req} is the resource require time point.</td>
</tr>
<tr>
<td>R_{rel}TP (Resource release time point)</td>
<td>Select the available task based on the resource release time point where t_{rel} is the resource release time point.</td>
</tr>
<tr>
<td>LRTP (Longest resource usage time period)</td>
<td>Select the available task with longest resource usage time period. Max (t_i) where t_i is the resource usage time period of the i^{th} activity.</td>
</tr>
<tr>
<td>SRTP (Shortest resource usage time period)</td>
<td>Select the available task with shortest resource time period. Min (t_i) where t_i is the resource usage time period of the i^{th} activity.</td>
</tr>
</tbody>
</table>

5.7 MULTI-AGENT COORDINATION ARCHITECTURE

The multi-agent coordination can be represented as shown in Figure 5.6. The agents in this system are allowed to communicate directly with other agents which is desirable for executing concurrent activities.
The central contribution of this work allows the agents in the multi-agent coordination system to have an automated decision-making using the event-control-action rule. A process is carried out to solve a problem, in other words to attain a specified goal. The problem (goal) is decomposed into smaller sub-problems (sub-goals) at various levels; corresponding to this, the overall process is divided into subprocesses, tasks and sub-tasks. Once the individual solutions of the lower most sub-problems are obtained, these sub-solutions at the lower levels can be aggregated to reach the solution of their parent problem and this continues in a bottom up fashion till the top level process to reach the overall solution. This process involves cooperative sharing of resources among a group of agents. Thus, a multi-agent coordination system involves a group of heterogeneous agents which requires effective resource allocation.

To model the project, various agents are defined with the roles to be played to carry out the process with the aim of minimising the project makespan. The agents
are: task agent(s), resource agent, resource allocation agent, scheduling agent, resource require / release agent and constraint management agent and schedule generation agent.

Task agent(s): Holds the information about the set of tasks with attributes and the agent is responsible for executing the tasks in the schedule in a non-preemptive way (no pause is allowed) and is responsible for the start and end of execution of each task.

Resource agent: This agent holds the information about the resources which are required by the task agents and it also holds the status information.

Resource allocation agent: This agent is responsible for allocating resources to tasks one at a time based on various control and resource flow policies. Determines which of two agents may access the resource first if a conflict is detected.

Schedule generation agent: This agent produces schedules based on the schedule generation scheme which are used to find feasible solutions.

Scheduling agent: This agent is responsible for producing optimal scheduling based on event-control-action rule. This agent is responsible for resource identification, resource selection and task scheduling. Resource Identification is the first task of the scheduling is resource identification and the goal is to identify the list of resources that are available to a project. Resource Selection is the second phase of scheduling is the selection of those resources to meet the time constraints. In order to fulfill the restrictions, the scheduling agent has to gather dynamic information about the resource availability. Task Scheduling is the next stage of scheduling agent is task scheduling, i.e., the mapping of tasks to resources, trying to minimise project completion time.
Resource require / release agent: This agent is responsible for identifying when the resource is required / released based on the task information. Also maintains the utility value over a resource for each task.

Constraint management agent: This agent is used to manage the constraints based on the information received from resource and task agents.

The multi-agent project coordination architecture that has been designed as shown is Figure 5.7.

Figure 5.7 Multi-agent project coordination architecture

For the agent set $A = \{A_1, A_2, \ldots, A_n\}$, and for the task set $T = \{T_1, T_2, \ldots, T_m\}$, the task base is recorded in the tuple. $<$Task, Description, Agent, Dependence$>$
More precisely, for an agent $A_i (A_i \in A)$ performing a task $T_j (T_j \in T)$ the tuple is shown as follows.

1. $(T_j, A_i, O)$ represents that the agent $A_i$ performs the task $T_j$.

2. $(T_j, A_i, \bigcup_{k=1}^{m} (T_{jk}, A_{jk}))$, states that the agent $A_i$ is a coordinator of the joint Task $T_j$, which can be divided into $m$ sub-tasks $T_{jk}$ performed by corresponding agents $A_{jk}$.

As the planning and scheduling proceeds further down, the detailed information about the duration, timing, input, output and required resources becomes available, and the planning and scheduling decisions made at higher levels will have to be corrected, the overall schedule is updated. The ultimate goal of scheduling is to unfold the project plan into an executable detailed schedule. The scheduler determines the order of the operations and the resource allocations with respect to the procedure and resource constraints. The set of activities to be scheduled are determined by disaggregating the activities that fall into a given time unit in the plan.

In the scheduling model, tasks have specific processing and resource times. Temporal constraints are the precedence between the tasks and resource constraints describe that the resource requirements of the particular operations should be satisfied by limited resources. Therefore, the solution of this problem is an assignment of start times to tasks so that all temporal and resource constraints are satisfied. The scheduler generates detailed predictive schedules that satisfy all the precedence and resource constraints, and approach optimality with respect to the actual optimisation criteria.
5.8 DISCUSSION

Existing project management tools are all oriented towards repetitive and completely foreseen sequences of tasks from beginning to end. The CE approach introduced to project scheduling calls for task overlapping and joint planning of work. The feature which distinguishes the present work from the existing literature in both classical planning and distributed multi-agent coordination consists in the fact that the constraints on the plans/strategies to be obtained, namely precedence and resource constraints. The problem of reasoning about time and resources is usually referred to in the literature as scheduling.

A single dependent relation between an upstream activity $T_1$ and a downstream activity $T_2$ is called a non-overlap pattern if activity $T_2$ cannot start until $T_1$ is finished. Obviously, a zero degree of overlapping between activities is a non-overlap and the relation is referred as sequential (serial). In contrast, an overlap pattern represents the case when activity $T_2$ can begin prior to the completion of the upstream activity $T_1$ and proceed with the activity. The term degree of overlapping is used to represent how early activity $T_2$ can start before completion of activity $T_1$. An overlap can accelerate the progress of a downstream activity by starting some tasks in advance before an upstream activity is completed. Therefore, starting the downstream activity early might reduce the project duration.

Due to resource constraint, the activities in the branch-merge pattern are scheduled sequentially though the precedence relation permits parallel scheduling. This will increase the project duration due to resource constraint. However, the overlapping in branch-merge pattern reduces the total duration. Therefore, this model
considers both sequential and branch-merge patterns with precedence and resource constraints. The optimization of project scheduling focuses on constrained unique resources and the precedence relations among activities for minimizing the latest end time of the activities or minimizing the start time of activities. Since the concurrency is affected by limited resources and the complexity of coordination is dominated by resource constraints, the time consumed for coordinating activities is considered only at the initial stage (i.e. planning of allocation resources to activities).

In the architecture presented, the project management tool is used to setup the project and plan the activity schedule. The roles, resources and the application infrastructure to be used to execute the tasks are assigned to the tasks in workflow modelling phase. To support resource constraint-driven workflow execution semantics, the resource and the time points that the tasks require or release them should be specified in the workflow model. The constraint management agent will extract the resource constraints from the workflow according to the task dependencies and the specified resource, especially the resource require and release setting and the constraint management agent will manage these resource constraints.