CHAPTER 5

CATALYST AGENTS

5.1 INTRODUCTION

Agents are computer programs, that simulate a human relationship whereby one agent does the work needed by another. The traditional definition of intelligent agents is that, they are autonomous software entities that can navigate heterogeneous computing environment and can, either alone or working with other agents to achieve some goal. These agents do a specific job in the background for the parent system, while the parent is performing other tasks and hence, avoids synchronization problem and the physical involvement of parent, in group activities like planning, scheduling, distributed learning etc., (Petrie 1996, Falchuk 1998, Maes1994). As already discussed in chapter 3, in order to reduce workload, the collaborator uses the catalyst agents (Anandakumar 1998c) for information exchange and gathering. These catalyst agents are generated by the Skill Managing Agent (SMA). The functions of SMA and the design and implementation of the catalyst agent are discussed in the following sections.

5.2 SKILL MANAGING AGENT (SMA)

The main functions of the SMA are: i) Generating the Catalyst Agent(s), and replicating as many copies of it as suggested by the CAA, ii) Selection of skill(s) to incorporate into catalyst agent, and iii) Maintenance of skill-base. It contains all the skills (technical or managerial) the
collaborator itself possesses. Here, the SMA generates the catalyst agents in three different situations (see figure 5.1).

1. As a communicating agent

2. As a Negotiating agent

3. Information exchange and gathering agent

Figure 5.1 Different situations in which catalyst agents are created

(1) When the promising group (see chapter 6) is formed by the Mind Protocol Agent (MPA), the catalyst agents are used as delivering agents to deliver project requirement profile (see chapter 6) to the group members as chosen by the MPA. (2) At the time of task allocation, the catalyst agents are used as negotiating agents by the MPA among the group members. (3) During the development process, the catalyst agents acts as a information exchange or gathering (sending partial results /reliability values /sub goals) agents.
5.2.1 SMA at source collaborator

Whenever the collaborator (source collaborator) needs help in information gathering or in solving complex problems, the CAA of that collaborator decides to create catalyst agents and for this purpose activates the SMA. It is the CAA which decides the purpose of the catalyst agent, and the destination collaborators to which the catalyst agents have to be sent. The SMA, on instruction from CAA generates the appropriate catalyst agent selects the particular skill from the skill-base and incorporates it into the catalyst agent. The SMA makes multiple copies of that agent, and these agents are then allowed to migrate to the chosen set of collaborators. The source collaborator then wait for response from catalyst agents, and assumes rejection of the catalyst agent by the other team members (destination collaborators), if the result is not received within a pre-specified time interval.

5.2.2 SMA at destination collaborator

At the destination, the catalyst agent knocks at the door for entry. Figure 5.2 gives the major steps involved in servicing. The catalyst agent. Mechanism of catalyst agent allows the destination collaborator, the option of either catering to the catalyst agent or not entertaining it. The CAA of the destination collaborator first analyzes its tasks, priorities, load factor, and then decides whether to service the catalyst agent's request or not. If the destination collaborator decides not to service the request it just kills the catalyst agent. On the other hand, if it is willing to help, it allows the agent to enter into its system. It is the CAA, then examines the service requested by the catalyst agent and modifies the catalyst agent's parameter values (if needed), and then the catalyst agent is executed. This parameter changing provision, allows work done by the destination collaborator for the catalyst.
agent to be modified to suit its own priorities. The result is then sent back to the source collaborator.

Figure 5.2 High level view of the Catalyst Agents at destination collaborator
While servicing the catalyst agent, the Similarity Analysis Agent of the destination collaborator compares the skill present in catalyst agent with the local skills and if it is new, then this new skill is added to the skill-base depending on the relationship (described in chapter 6) that exists. If the skill already exists, then depending on the results obtained from the Similarity Analysis Agent, the local skill is refined or replaced. This analysis gives more information about the source collaborator, and also helps in building the relationship between the two collaborators. However, if the catalyst agent affects the destination collaborator's performance, it can kill the catalyst agent at any time.

5.3 CATALYST AGENT

Here, Catalyst agents are designed to assist the collaborator in a number of different ways: sending out information (partial results/reliability values/sub goals), information gathering, and negotiating at the time of task allocation. The catalyst agent's characteristics are listed and are compared with other agent models in the following section.

5.3.1 Catalyst Agent's characteristics

Important characteristics of catalyst agent are: (a) It has three parts: Invitation, Body, and Echo parts (discussed later in this section). (b) It is dynamically created, (c) It is a special-purpose migrating agent (d) Any skill can be placed into the agent, (e) It returns only the Echo part, (f) Mechanism of catalyst agent, allows to modify parameter values by the destination collaborator, (g) It commits suicide when the job is over, (h) Destination collaborator has the right to kill it, at any time.

An agent model where agents are part of the system is discussed in RETSINA (Sycara 1996). Here the predefined agents are distributed to
different nodes (collaborators). These agents communicate with each other to request or provide information, and are designed for Net-based information environment. The work carried out by Lander (1997) includes library of expert agents, where the agents may be modified and reused. They are able to adapt quickly to changes without discarding the main body of existing work. These agents are designed to improve distributed AI system's performance by sharing meta-level search information. However, in the work discussed here, the catalyst agents never exist before they are needed, and hence are created as and when they are required, that is, no library of expert agents exist. Catalyst agents are also not reusable as compared to reusable agents discussed by (Lander 1997) and mobile agent (Kiniry, 1997). In this work, the catalyst agents are actually killed or allowed to commit suicide after use. In agents like Newsfinder (O'Leary 1997), Softbots (Etzioni 1994), the complete structure of the agent sent to the destination will be returned along with the results. Whereas, only the part of the catalyst agent that contains requested information is transmitted back to the source collaborator, while rest of the catalyst agent is disposed by the destination collaborator itself. An important issue that catalyst agent tackles is, the right of the destination collaborator to refuse to service the request made to it. The overview of the Catalyst Agent is given below.

5.3.2 Catalyst Agent's Internal Structure

The internal structure of the catalyst agent is as shown in Figure 5.3. It has three parts: INVITATION part, BODY (or stomach) part, and ECHO part. The Invitation part has four slots: Destination address, catalyst agent type (information gathering or negotiation), Task description, and catalyst agent's parameters. The BODY part is the heart of the catalyst agent. One of the most difficult aspects of agent design is to define specific tasks (Kautz 1994). In the proposed work, the SMA selects the specified skill from the
skill-base, and incorporates it into catalyst agent's stomach or BODY part. Thus, it is possible to generate various types of agents, depending on the need of services. The ECHO part has three slots: Source address slot, catalyst agent id, and Result slot. The SMA initializes the catalyst agent's type (as specified by CAA), and Task description slot. Then, the SMA places the specified skill in to the BODY part of the catalyst agent, and initializes input parameters of the INVITATION part. Finally, the SMA fills the Source address slot of the ECHO part. This completes the birth of the Catalyst Agent. Now the SMA replicates the agent as many as required, and are then given to the CAA. The CAA fills the Destination address slot of the ECHO part, which are given by the Mind Protocol Agent. Then, the agents are distributed to the destination collaborators.

<table>
<thead>
<tr>
<th>INVITATION PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination address:</td>
</tr>
<tr>
<td>Catalyst agent id:</td>
</tr>
<tr>
<td>Agent Type:</td>
</tr>
<tr>
<td>Task description:</td>
</tr>
<tr>
<td>Parameters:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BODY PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skill/ method</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source address:</td>
</tr>
<tr>
<td>Catalyst agent id:</td>
</tr>
<tr>
<td>Results:</td>
</tr>
</tbody>
</table>

Figure 5.3 Internal structure of the catalyst agent
The destination collaborator treats this incoming catalyst agent as an ordinary process. The priority for this agent is assigned depending on the relationship (see chapter 6) between those two collaborators. Once the catalyst agent is embedded into the local system, the destination collaborator gets full control over the catalyst agent (i.e. No global control).

Catalyst agents, work in all environment considered in this work, that is, cooperative, noncooperative, and partially cooperative environment. The information given by the catalyst agent is measured by the trust factors (see chapter 4) depending on the environment. All interactions with the collaborators are performed locally which results in considerable saving in network bandwidth. The catalyst agent has two main advantages over RPC (Birrell 1984) and message passing (Litzer 1996). First, moving processing functions close to where the information is available, reducing the amount of communication between source and destination collaborators. The catalyst agent is sent only once (with its initial parameters) to the destination collaborator. Then, it interacts with that collaborator locally without using network; and after completing its task, it sends the result back to the source collaborator. Second, it decreases the degree of dependency between the collaborators.

5.4 BENEFITS OF THE PROPOSED METHOD

The proposed method gives a number of benefits to both the source and the destination collaborators. From the source collaborator's point of view, it uses its own skill at the destination collaborator, and this helps in many ways. (i) the results are more reliable (trustable) as source collaborator's own skill is used in computing them. (ii) the results are in an acceptable format, that is, there is no conflict due to format mismatch. Since, it is the source collaborator itself who specifies the returning format. Three,
catalyst agent's reaction helps the collaborator to know more about destination collaborator's state (busy or free), interest, willingness (based on accepts/rejects) etc. This gathered information helps in refining the relationship between those two collaborators. Other advantages are, the participation burden is reduced, that is, catalyst agent does the job on behalf of a collaborator, and it gives a natural way of coordination (i.e. Believe me to Believe you).

From the point of view of the destination collaborator, the catalyst agent may be accepted or rejected depending on their interest or relationship with that source collaborator. The destination collaborator has full control over the catalyst agent, hence it can kill the catalyst agent at any time. The main advantage of this method is that, the destination collaborator gets more knowledge about the source collaborators regarding their skills, accuracy of results, and percentage of similarity with respect to local information etc. There are three other advantages in servicing (i.e. sending requested information) the catalyst agent. (i) Without requesting any other collaborator for a particular skill, the destination collaborator can gain new skill, provided the catalyst agent's skill is new. (ii) The destination collaborator can refine the existing skill. (iii) The destination collaborator can make use of computed results without extra effort. That is, the destination collaborator can gain extra information from the catalyst agent by changing parameter values to its requirement, also associated with the catalyst agent's skill.

5.5 TYPES OF CATALYST AGENT

In this work, the CAA of each collaborator maintains catalyst agent's table. The table contains catalyst agent name (or id), agent's type, skill, to whom it was sent, when it was sent, expected time for results, other status information and remarks. It also maintains, counter variables, to keep track
number of requests made to the other collaborator and number of collaborators who accepted the request (or rejected requests). This information helps in knowing more about destination collaborators and are used in computing behavior trust factors (see chapter 4). The Increment Learning Agent (ILA) uses this information and produces impressions (discussed in chapter 7), which acts as background knowledge for future use. As it is discussed in chapter 3, the catalyst agent's features are highlighted through the various applications considered in this work. The different types of the catalyst agents are explained one by one in forthcoming sections.

5.5.1 Catalyst Agent as a Communication Agent

The first step in global software development is the group formation. The Mind Protocol Agent (MPA) constructs the most promising group of collaborators (discussed in chapter 6), and catalyst agents are generated. Now, the catalyst agent acts as a communication agent, which delivers the project requirement profile of the problem (see chapter 6) to the group members specified by the MPA. Then, the catalyst agent commits suicide. Further, while developing software, catalyst agents are often generated for information exchange and gathering (partial results/information/sub goal).

5.5.2 Catalyst Agent as a Information Gathering Agent

Here, two different applications are considered to demonstrate the concept of the catalyst agent. One, automating the reliability checking (Anandakumar 1999d) of sub components designed and developed by individual team members, as well as other team members through the catalyst agent. Two, automating the Software Review process at the requirement phase of the software development cycle.
5.5.2.1 Catalyst Agents in Reliability Checking

To simplify the problem, a small group (say size = N) is considered with a team leader. It is assumed that the leader has already distributed the subtasks and its specifications, responsibilities, standards etc., to the other team members via the Web media (Jerry 1999). Since the modules are developed by different teams at different places the reliability factor plays an important role in developing quality and reliable software product (Beam 1989). Hence, while developing, each collaborator finds the reliability of local subtasks and interacts with the other collaborators (other team members or with the leader) in order to control and to achieve overall reliability of the product. Further catalyst agent helps in computing group reliability as well as the internal consistence reliability (Kaplan 1987, Govil 1983).

5.5.2.1.1 Reliability at team members level

If the subtask is large or complicated, the team member further divides (Anandakumar 1999d) it into number of submodules (say k modules). Let \( r_{aij} \) be the reliability of submodule \( M_j \), \( 1 \leq j \leq k \). The member is allowed to use any of the reliability methods such as Redundancy techniques, Replacement techniques and K-out-of-n replacement techniques unless otherwise specified by the leader. The reliability of each submodule \( M_j \) varies according to their internal characteristics. A priority \( IC_j \) is associated with each submodule \( M_j \) according their role or importance. Hence the reliability of the given subtask is

\[
\text{rst} = \sum_{j=1}^{k} r_{aij} \cdot IC_j
\]  

(5.1)

where \( \text{rst} \) is the total reliability of subtask assigned to that team member\((C_i)\). The \( IC_j \)'s are chosen such that the sum of all priorities is 100.
Now the team member needs the predicted reliability (RPci) value. The team member may predict reliability for each submodule, and subtask or it may be requested through the Catalyst agent from the leader or other team members. Depending on the previous trust factors (Anandakumar 1997a, 1998b) about the other team member Cj, he selects co-team members from whom he can get predicted reliability and attaches the associated trust factor (LTF) with the supplied predicted reliability RPcij. He then selects RPcij with maximum LTF.

$$\text{RPc}_{ij} = \text{RPc}_{ij} \cdot \text{TF}_j$$

(5.3)

If the RPcij is predicted locally, it depends on the mean of previously solved similar types of subtasks (or modules) assuming the same reliability for subtask and submodule. The similarity analysis (Anandakumar 1997) is used to find the similar subtasks (or modules). The locally calculated RPcij is given by

$$\text{RPc}_{ij} = \frac{1}{q} \sum_{j=1}^{q} \text{PSR}_j$$

(5.4)

where PSRj is the previously solved reliability values and q is the number of matched subtasks (or submodules). Then the member finds the difference (error) between the predicted value and the observed value.

For submodule

$$\text{error}_m = | \text{rm}_j - \text{RPc}_{ij} | \text{ for modules}$$

For subtask

$$\text{error}_s = | \text{rst} - \text{RPc}_{ij} | \text{ for subtask}$$
The Null hypothesis and the Alternative hypothesis (Kaplan 1987) are used to decide whether the submodule or subtask is reliable or unreliable. They are given below.

Null hypothesis
H0: Error <= acceptable error THEN Subtask is reliable, So accept it.

Alternative hypothesis
H1: Error > acceptable error THEN Subtask is unreliable, So reject it.

Here, the acceptable error is supplied by the leader, and depending on the hypothesis H0/H1, the collaborator decides whether to resolve the subtask with new modified skill or to send computed rst through the ECHO of the catalyst agent back to the team leader. If there are no more alternatives to solve the subtask, the catalyst agent may then be killed.

5.5.2.1.2 Reliability at the team leader level

The responsibility of a team leader is to determine contributions, monitor performance, review team needs etc. This concentrates on the performance to achieve the reliability of the developing process. The leader periodically sends the catalyst agent to gather reliability information (rst_{i}) from the team members. Then the total reliability (R) is computed by:

\[ R = \sum_{i=1}^{N} rst_i \cdot ISk_i \cdot TF_i \]  

(5.5)

Here, N is the total number of team members, LTF is the trust factor for the team member C_{i} which indicates how much trust the leader should put on that C_{i}, The ISk_{i} is the priority associated with the subtask ski depending on the importance of that subtask. The ISk_{i}'s are chosen such that
The reliability \( r_{st} \) (of team members) & \( R \) (of group) is used to improve the performance of the systems. The group reliability primarily depends on the functions and responsibility of the group which are reliability assessment, reliability design and review, reliability testing and coordination of related activities. In order to achieve good reliability, there is a need for information gathering and exchange among members. There is a need for monitoring and controlling of the group reliability and individuals reliability; so that the coordination among the group members, can be improved and it would be possible to achieve reliable end product. To analyze these reliabilities (individual/group), the leader uses the Cromback (Kaplan 1987) estimation called coefficient \( \alpha \) which is given by

\[
\alpha = \left( \frac{N}{N-1} \right) \left( \frac{S^2 - \sum S_i^2}{S^2} \right) \quad (5.7)
\]

Here \( N \) is the size of the team, alpha coefficient depends on \( S^2 \) the reliability variance of the group, and \( S_i^2 \) the reliability variance of the individual team member. The individual reliability variance \( S_i^2 \) is found by the equation

\[
S_i^2 = \frac{\sum (r_{st_i} - \bar{r}_{st})^2}{t} \quad (5.8)
\]

where \( r_{st_i} \) is the previously solved subtask's reliability, and \( r_{st} \) is the mean of these values, and \( t \) is the number of subtasks solved previously by the same team member (including the current subtask). The group reliability variance \( S^2 \) is given by the equation:
The $S_i^2$ parameter of $\alpha$ helps the leader to decide whether to replace any group member $C_i$ or not, whereas $S^2$ helps in deciding the overall effectiveness of the group. If any $S_i^2$ deviates more from the coefficient $\alpha$ the leader may replace the group member, if the overall group reliability is below acceptable level, otherwise he may give instruction to the team member to improve himself. This analysis can be carried out for groups in addition to individual in which case each group is a subsystem. In distributed system, reliability can be improved by group replacement. The group can be described either as a $k_1$-out-of-$N$:G system indicating that at least $k_1$ team members should be good for the system to be good or a $k_c$-out-of-$N$:F system indicating that at least $k_c$ team members should fail in order to cause the system to fail. The leader uses the logic-diagram approach to analyze the characteristics of group reliability characteristics. The order of number of minimal tie sets contained in TSLD would be

$$Mt = \frac{N!}{(N-k)! k!}$$  (5.10)

The system is good if at least one of the tie set is good and a tie set is good only if all the team members of that particular tie set are good. Therefore, the probability of success of the system is

$$SR = P(\text{At least one tie set is good})$$
where $E_i$ is the event that tie set $TS_j$ is good. For example, let $N = 3$, then

$$SR = P\left(\bigcup_{i=1}^{N} E_i\right) \quad (5.12)$$

and $P(E_i \cap E_j) \neq P(E_i) \cdot P(E_j)$ as the events are not mutually exclusive event. This estimation helps the team leader, to select more reliable group. Also helps in predicting, monitor and control the group activities, the quality of development process and according to that the leader will issue the instructions, so as to achieve good reliability (Grady 1993). Since, the checking of reliability is done and monitored while developing, the development cycle time is reduced and quality is controlled at the early stage itself.

5.5.2.2 **Catalyst agent in Software Review Process**

In traditional view of the software development life cycle, the testing is immediately prior to installation and maintenance (Musa 1996). 30-40 percent of project effort is spent on testing (much more in the case of "human-related" software) (Mehata 1998, Birger 1994, Montie 1988). If lower cost and higher quality system are the goals, verification must not be isolated to a single phase in the development process, but rather, incorporated into each phase of development. The issues of review process, in software development process assumes greater significance when the development
process is through the Web media. Hence, review process (Perry 1995) is automated at the requirement phase, using a special agent, the catalyst agent for information gathering.

Of the many review processes, the walk-through is the least structured and the most prone to creativity. The objective of walk-through is to create a situation in which a team (review team) of skilled individuals helps the project team in the development of the project solutions. Walk-through process has five steps (Knight 1993, Perry 1995). To simplify the problem, a review team of size three is considered who are distributed geographically over the Web. The team member's discussion is carried out through the catalyst agents. Walk-through process has five steps, which are to be completed in the sequence listed below. The amount of time allocated to each step will be dependent upon the size of the application being reviewed and the degree of assistance desired from the walk-through team. Step 1: Establishing the ground rules. Step 2: Selecting team (see chapter 6)/notifying participants. Step 3: Project presentation. Step 4: Questions / recommendations, and Step 5: Report generation. In each steps the collaborators (reviewers) exchange different kind of forms.

Here, the software review environment is simulated with team (size of three) members Senior Manager, Project Manager, and User Manager. Initially, the ground rules are (manual input) are established (form 1). This form is prepared individually and is communicated (through the catalyst agent) to each other to construct single form. These ground rules give the clear-cut picture of the entire process like team size, review team leader, walk-through time and location, confidentiality on information, who receives the results, etc. Then, in step 2, notification form (form 2) is prepared by the review team leader (here it is assumed to be senior manager). The leader fills this form 2 into the catalyst agent, replicates it and communicated to the
other two members. Now, the project team prepares (manual input) the presentation, which contains the project details, project-name, project-duration, modules involved, methods, test factors, reliability factor etc (Yegneshawer 1998). Further, they prepare individual module details if needed. The Project Manager give the presentation, the catalyst agents are used to deliver the presentation. Once the presentation is over, the review members rise questions (form 4a) / recommendations (form 4b). Question and recommendation process is continues till all the test factors (form 5) are satisfied by all the review members. Finally, the reports (individual) are generated and are collected using the catalyst agents by the reporter, who constructs the single report, and is issued to the project team.

5.5.3 Catalyst agent a as Negotiating agent

The aim of negotiation is to arrive at a common agreement between involved collaborators. In order to reach an agreement, collaborators exchange information in a multi-level way within a negotiation loop. The catalyst agent plays an important role in negotiation, at the time of task allocation. The figure 5.4 shows the negotiation steps and this concept is demonstrated with an application in the following chapter.

5.6 CONCLUSION

Catalyst Agent, and its characteristics have been presented in this chapter. The comparison of the catalyst agent with other agent models is also highlighted. The different type of Catalyst agents and their function id is brought out through different applications. The Mind protocol - a high level intelligent protocol has been proposed in the next chapter, which uses the information gathered by the catalyst agent and quantized by the Behavior Model described in chapter 4.
Figure 5.4 Negotiation steps via the catalyst agent with the Destination collaborator