Improved Performance of Agent Based Placement Cell System - A Performance Efficient Role Clustering Technique

SOUMYA SURAVITA\textsuperscript{1}, PRABHAT RANJAN \textsuperscript{1}, R.K. SINGH \textsuperscript{2} & A. K. MISRA\textsuperscript{1}

\textsuperscript{1}Department of Computer Science & Engineering, \textsuperscript{2}Electrical Engineering Departments, Motilal Nehru National Institute of Technology, Allahabad -211004, Uttar Pradesh, INDIA

Abstract: - This paper provides three criterias of clustering within a placement cell multi-agent system. Our goal is to group roles with similar objectives. Optimize the system performance by minimizing the overall interaction, data transmission and competition of shared resource between roles/agents. This paper presents a novel systematic approach to optimize the system performance by exploiting the relationships and dependencies among roles as well as clustering of roles and mapping criteria between roles to agents. The proposed clustering algorithm partition the overall system roles/agents into several clusters. Optimal cluster size can be obtained by user-defined performance parameter (\(\eta\)). The performance of the agent-based system, enhanced with our algorithm are investigated via implementation of placement cell case study. The results indicate that our proposed algorithm enhances the system performance if intersections of the cluster are minimal.

Keywords: - Role, Agents, System Performance, Role Dependencies, Lifespan, Clustering and Mapping of Role.

1 Introduction

The roles define expected behaviors of the agents and are an important concept used for different purposes like modeling of structure of multi-agent system, modeling of protocols and components of agent design [11,12]. The concept of role has been advocated to model application domain agents, which evolve dynamically during their lifespan. Understanding the relationship among roles can help the system analyzer to refine and optimize the role model [2]. Moreover the implicit conflict among roles can also be identified. Agent performs operations independently but communication between the agent and their access to shared data establishes causal dependencies in the operation [11,12]. Dependency of roles from their base agent or role is the main characteristics of roles in the Cassiopeia methodology [1]. Identifying roles and mapping the role to an agent are essential phases in many proposed AOSE methodologies like GAIA [8], PROMETHEUS [6], ROADMAP [13] and TROPOS [4].

A role can be defined as an entity consisting of a set of required permissions, a set of granted permissions, a directed graph of service invocations, and a state visible to the runtime environment but not to other agents [7]. A role can also be defined as an abstraction of the behavior of an object, which consists of a subset of the interactions of that object together with a set of constraints on which they may occur [5]. The agent based system communication can be divided into two broad categories: Tree based and Cluster based. From Figure 1, it is clear that cluster based communication is more efficient than the tree based communication. Clustering supports all communication protocols like one-to-one, one to-all, one-to-any, and many-to-one.
Clustering is viewed as a search problem in multi-agent system in which individual roles/agents have the goal of finding other “similar” roles/agent. Roles/agents within a cluster coordinate, combing their local views to each member.

The existing methodologies generally do not consider one or all of the following:
- The relationships and dependencies among roles.
- Performance-efficient clustering of roles.

The proposed methodology considers the relationships and dependencies among roles and the clustering of role and mapping criteria between roles to agents. Our previous methodology [9,10] is used as fundamental ingredient in the new proposed methodology.

The paper is organized into several sections. Section 2 describes the criteria of our proposed methodology. Section 3 explains the Role Model of the proposed methodology. Section 4 explains the proposed algorithm. Section 5 provides an overview of the Design Phase. In section 6, the algorithm is investigated via implementation of placement cell case study. Section 7 presents conclusion and future work.

2 Proposed Methodology

The proposed methodology is to optimize the system performance. An agent-based system consists of software entities called agent, which interact with themselves and other resources to perform goals. Each agent plays some role in the environment. To optimize system performance in an agents-based system
- The relationships and dependencies among roles are identified and analyzed.
- User-defined performance parameter (\( \eta \)) is considered for optimal clustering of role.

The clustering algorithm is to partition the overall system roles/agents into several clusters. The objective of clustering is to minimize the
- Global interaction or overall interaction between the role/agents.
- Data transmission between the role/agents.
- Competition of shared resource between the role/agents.

Optimal cluster size can be obtained by performance parameter (\( \eta \)). The clustering of overall system roles/agents is depends on the following criteria:
- Frequently interaction (\( R_I \)) between roles
- Closely related role (\( R_R \)) on the basis of
  - Performing similar nature of task
  - Role capabilities
  - Role dependencies
- Roles using shared resource (\( R_{SR} \))

In general, an agent can play more than one role, but it is very application specific to map the role to agent. The mapping between the role and the agent can be one to one, one to many, many to one, or many to many. In most cases, there may be a one-to-one correspondence between roles and agents. But an agent may also play some closely related roles for purpose of convenience and modularity. The role instances only exist in association with agent instances. Role’s life begins when the agent acquires it according to current condition. When condition changes, agent drops previous one and acquires the new role. The agent will play different roles at different time and in different conditions during its life cycle [14]. We illustrate this methodology through a placement cell case study.

3 Role Model

The role model divides the goal/sub-goal into tasks and further into sub-tasks. The sub-tasks are grouped on the basis of relative interconnectivity and closeness among them. The subtasks are performed by the cooperation of fewer roles. The role model describes the properties of a role. The role model consists of role identification, role description, tasks identification and agent identification/mapping criteria.

A role identified by its name \( R_N \). Role description \( R_D \) is composed of a set of six specific characteristics.

i.e. \( R_D = \{ R_T, R_A, R_C, R_B, R_ST, R_CC \} \)

Where,
\( R_T \) is the Role Type
\( R_A \) is the Role valid Activation
\( R_C \) is the Role Cardinality
\( R_B \) is the Role Behavior(Norms and Rule)
\( R_ST \) is the Role specific Task
\( R_CC \) is the Role certain Capabilities.

Roles Type \( R_T \) is composed of a set of four specific role of the system, each of which serves some specific task of the system in accomplishing the overall objective of the system.

i.e. \( R_T = \{ I_R, D_R, PI_R, PD_R \} \)

Where,
\( I_R \) is the Independent Role. An independent role may
be acquired or dropped without any consideration of other roles.

\(D_R\) is the Dependent Role. A dependents role has some form of dependency relation with other roles.

\(PI_R\) is the Partially Independent Role. A partially independent role is performed/handled for some specific role instance or sub-tasks independently and all other instance of role or sub-tasks is dependent on other role.

\(PD_R\) is the Partially Dependent Role. A partially dependent role has some specific role instance or sub-tasks dependency relation with other role and all other instance of role or sub-tasks is performed/handled independently.

In Partially Independent and Partially Dependent case, an agent may acquire or drop the role only in case of the role instance being independent. Understanding the dependency or relationship among roles help the system analyzer to refine and optimize the role model. It also optimizes system performance in an agents-based system by analyzing dependency or relationship among roles.

The role valid activation \(R_A\) gives number of times a role can be taken by an agent. The cardinality of role \(R_C\) specifies the maximum limit on its instances at any time with any agent. The behavior of role \(R_B\) describes that the role requires certain nature of behavior to perform a task. There are some specific norms and rules to perform a task. In some cases, the role behavior conflict. For example, in an organization if a staff member who is also a private consultant may have conflicting job responsibility. In this case different roles by the same person is possible, but it would require appropriate rule and norms to resolve the conflicting behavior. The role specific task \(R_{ST}\) is responsible for achieving, or helps to achieve some system specific task. The certain capabilities of role \(R_{CC}\) describes how well an agent may play that role in light of the capability it possesses. Capabilities are key to determining exactly which role can be assigned to what tasks in the organization.

When we consider what kind of agents and what mapping between agents and roles we need, there are some considerations proposed by Chen [3]:

- If the roles are distributed at different places, they are not suitable to be played by one agent.
- We can make an agent to play roles to decrease the communication load of the whole system when situations described below occurs.
  - The communications and interactions frequently occur between roles and that may seriously increase the communication load of the whole system.
  - There is great amount of data transmissions between roles.

- For a basic mechanism to implementation stage to resolve the competition of public resource, we need to analyze the usage of a public resource by agent. Because an agent can only play a role at one time, the roles that would be played by an agent would not compete with the resource. However, if the roles that the agent plays are frequently required to serve in the system or they require of time to accomplish their tasks, this mechanism would not be suitable. The system would have a bottleneck in the agent, and the performance would be reduced seriously.

- For a simple mechanism to implementation stage to resolve the conflict with goal between agents, the controlling actions of agents in timing must be considered.

Role may interact with another role only if they belong to the same group. Interaction is done through asynchronous message passing. If the message fails to satisfy constraints \((R_T, R_A, R_C, R_B, R_{ST}, R_{CC})\) from any roles concerned, the message will be rejected and action will be taken to handle the error. The cohesion of the whole system is maintained by the fact that role may belong to any number of groups depending upon the above-mentioned criteria, so that the communication between two groups may be done by roles that belong to the both group.

At the time of clustering of roles and mapping to the agent we take care of the role specific characteristics \((R_T, R_A, R_C, R_B, R_{ST}, R_{CC})\), which is describe in the role description \(R_D\). The addition and deletion of role into cluster and mapping to agent is permitted only when the role specific characteristics constraints \(R_T, R_A, R_C, R_B, R_{ST}, R_{CC}\) are valid. If any role constraint is violated, it raises corresponding exception. There are five different kinds of exceptions:

- Role Relationship Exception: It is raised when any of the role relations are violated.
- No Such Role Exception: It is thrown if a role instance referred to, does not exist.
- Conflict Role Behavior Exception: It is raised if such role that we are trying to add in cluster and map to agent, may cause conflict in other role behavior.
- Duplicate Role Exception: It is thrown if
4 The Proposed Algorithm

The proposed clustering and mapping methodology is as follows:

Assumptions for our proposed algorithm are:

(i) Every role has a unique ID. Role ID is dependent on the cluster.
(ii) Every role must come/assign into any one the cluster type.

1. Initially find out the total number of role in the agent based system i.e. \( R_n \).
   where, \( R_n \) would be the total number of expected role and \( n \) is varying from 1 to \( n \).

2. Apply the first criteria \( R_1 \) representing frequent interaction between the roles and make cluster of the role.
   i.e. \( C_1 = \{R_{in}, \eta\} \)
   Where, \( C_1 \) is the cluster by apply the interaction criteria \( R_{in} = \{R_{i1}, R_{i2}, \ldots \ldots, R_{in}\} \) would be the total number of expected frequent interaction specific role
   \( \eta : \rho(R_i) \) is a performance function which defines the time of interacting between each subset of roles.
   The performance function must satisfy the constraints, that by adding role in a cluster never decreases the performance of the system.
   Formally, this is defined as follows:
   If \( R_{i1}, R_{i2} \subseteq R_{in} \) are sets of role such that \( R_{i1} \subseteq R_{i2} \), then \( \eta(R_{i1}) \geq \eta(R_{i2}) \).
   The cluster size of \( C_1 \) is decided by performance parameter \( \eta \).

3. If cluster size of \( C_1 < R_n \) then apply the second criteria \( R_{SR} \) else exit.

4. Apply the second criteria \( R_R \) representing closely related role and make cluster of the role.
   i.e. \( C_R = \{R_{Rn}, \eta\} \)
   Where, \( C_R \) is the cluster by apply the closely related role criteria \( R_{Rn} = \{R_{R1}, R_{R2}, \ldots \ldots, R_{Rn}\} \) is the set of role which is performing similar nature of task, similar role capabilities and role dependencies.
   \( \eta : \rho(R_{R}) \) is a performance function which defines the time of executing each subset of tasks by role.
   The performance function must satisfy the constraints, that the adding task never decreases the performance of the system.
   Formally, this is defined as follows:
   If \( R_{R1}, R_{R2} \subseteq R_{Rn} \) are sets of role such that \( R_{R1} \subseteq R_{R2} \), then \( \eta(R_{R1}) \geq \eta(R_{R2}) \).
   The cluster size of \( C_R \) is decided by performance parameter \( \eta \).

5. After applying the second criteria we find out the intersection of cluster
   i.e. \( C_I \cap C_R \)
   The performance function must satisfy the constraints, that the removal of role from \( C_I \) and \( C_R \) and assigning it to intersection of cluster \( C_I \cap C_R \) does not decreases the performance of the system.
   If cluster size of \( (C_I + C_R + C_I \cap C_R) < R_n \) then apply the third criteria \( R_{SR} \) else only apply first (\( R_I \)) and second (\( R_R \)) criteria is applied and exit.

6. Apply the third criteria \( R_{SR} \) representing roles using shared resource and make cluster of the role
   i.e. \( C_{SR} = \{R_{SRn}, \eta\} \)
   Where, \( C_{SR} \) is the cluster by apply the third criteria \( R_{SR} \) \( R_{SRn} = \{R_{SR1}, R_{SR2}, \ldots \ldots, R_{SRn}\} \) is the set of role which is using shared resource.
   \( \eta : \rho(R_{SR}) \) is a performance function which defines the time/amount of access shared resource by role.
   The performance function must satisfy the constraint, that the adding shared resource role never decreases the performance of the system.
   Formally, this is defined as follows:
   If \( R_{SR1}, R_{SR2} \subseteq R_{SRn} \) are sets of shared resource role such that \( R_{SR1} \subseteq R_{SR2} \), then \( \eta(R_{SR1}) \geq \eta(R_{SR2}) \).

7. After applying the third criteria we find out the intersection of cluster
   i.e. \( C_I \cap C_{SR} \)
   \( C_R \cap C_{SR} \)
   \( C_I \cap C_{SR} \cap C_R \)
   The performance function must satisfy the constraints, that the removal role from \( C_I \), \( C_R \) and \( C_{SR} \) and assigning it to intersection of cluster \( C_I \cap C_{SR} \), \( C_R \cap C_{SR} \) and \( C_I \cap C_{SR} \cap C_R \) does not decreases the performance of the system.

8. Map the individual cluster of \( C_I \), \( C_R \) and \( C_{SR} \) to individual capable agent which is having desired
5 Design Phase

At the time of mapping role cluster \( C_I, C_R, \) and \( C_{SR} \) to agent, the mapping must satisfy the role constraints. The mapping of roles to agents should never violate any role constraint \( (R_T, R_A, R_C, R_B, R_ST, R_{CC}) \), and if any role constraint is violated, it raises corresponding exception.

9. Map the intersections of cluster \( (C_I \cap C_{SR}, C_I \cap C_R, C_R \cap C_{SR} \) and \( C_I \cap C_{SR} \cap C_R ) \) to individual capable agent which is having desired characteristics depending upon the cluster intersection.

10. At the time of mapping intersections of role cluster \( C_I \cap C_{SR}, C_I \cap C_R, C_R \cap C_{SR} \) and \( C_I \cap C_{SR} \cap C_R \) to agent, the mapping must satisfy the role constraints. The mapping of roles to agents should never violate any role constraint \( (R_T, R_A, R_C, R_B, R_ST, R_{CC}) \), and if any role constraint is violated, it raises corresponding exception.

The system performance is optimal when the intersections of the cluster are minimal as described in the implementatation of case study.

5.1 Agent model

Agent model describes the agent type, agent instances and the relationship between these two [11,12]. Each agent may carry out one or more role depending upon the assignment. So agent type can be viewed as a set of roles. The relationship between agent and roles may be one to one, one to many. To optimize the efficiency, it always advisable to have more than one role in one-agent types. This assignment of role has been determined in the analysis phase. Agent instance represents the number of instance that particular agent will have in the run time. This is represented by an annotation ‘n’. This ‘n’ may have different values, which carries different meaning. An annotation ‘n’ means that there will be exactly n agents of that agent type during the run time. +, *, m…n, n are the different annotation used to represent the number of instance of an agent type during run time. The meaning of each annotation is given below:

+ means that there will be one or more agent instance during runtime

* means that there will be zero or more instances during runtime.

\( m…n \) means that there will be minimum of ‘m’ instances and maximum of ‘n’ instance during run time.

5.2 Agent Class Diagram

The agent class diagram represents how to achieve the goal of the system whereas the analysis phase defines the goals to achieve. It represents the overall agent system organization. The agent system organization consists of a set of agent classes [11,12]. The agent class is similar to the agent type of agent model. This represents conversation between the different classes.

5.3 Protocol model

Protocols are the rules and norms that define the patterns of interaction occurring in the system between various agents. This captures the inter agent interaction. The protocol models are used to define the message-passing protocols between agent classes [11,12]. This has the ability to show the alternative and repetitive message structures. AMUL is used for representing the protocol model. Here the syntax of AUML is used to represent the conversation between the agent classes. It is used to design and depict the message passing in multi agent environment with an allowed sequence of messages between agents having different roles.

5.4 Service Model

The main aim of the service model is to identify the services associated with each agent [11,12]. A service indicates the function of the agent. Any other agent can’t invoke this. This is the single, coherent block of activity where the agent will be engaged. For each services the properties must be defined properly. Basic properties of the service are input, output, precondition and post condition. Each agent must have at least one service. The service that an agent can perform can be identified from protocol model and role model. It just tells about services to be performed by agent and don’t put any restriction on how to implement it.
6 A Case Study
This section briefly illustrates how proposed methodology can be applied, through a case study. We have considered Placement Cell of our institute for this case study. Our objective is to develop agent-based placement cell system, which provides the following:

- System has to contact various companies and it has to obtain the details regarding what packages they provide for students of B.Tech., M.Tech and MCA and eligibility criteria for each company.
- System has to notify from time to time, the information regarding group discussion, interviews and written exam to students, notify about the recruitments, notify about interviews and written exam to students, notify about the expiry of his eligibility when he has been offered two jobs.

The proposed methodologies use role model where they define the roles. These roles are performed by agents. But to which agent which role should be delegated so that the performance will be optimal should be based on capability. We have shown how to decide assignment of roles to different agent.

By using the proposed role model each role has been defined by certain attributes. From the case study we have found the following roles:

R₄ = Student_Profile_Holder role finds the list of eligible student on the basis of company’s eligibility criteria. Here the role description for ‘Student_Profile_Holder’ R₄={R₄, R₅, R₆, R₇, R₈} is {Ð₄, 1, 1, receiver, display, read}, maintain student profile, <read, write, modify>.

This role is partially dependent on the ‘Notifier’. It can start validation of student only when the ‘Notifier’ sends the request and eligibility criteria and is activated once. It exhibits the provider role when it executes the algorithm to find the eligible student list. It acts as requester when it receives the company’s eligibility criteria from the ‘Notifier’. To perform these roles, it should have the capability to read, write, and modify operation on the student master table.

R₅ = Marker role updates the job status field whenever company recruits a student. Here the description of role ‘Marker’ R₅={R₅, R₆, R₇, R₈, R₉} is {Ð₅, 1, 1, <receiver, provider>, marking, write}.

‘Marker’ role is totally dependent on Student_Profile_Holder. It performs the work of marking the status. It shows the behavior of consumer as it uses the information provided by the ‘Student_Profile_Holder’ to mark the status column of student master table. For this it should have the write capability on database.

R₆ = Placement_Cell role provides user interface form where the data for company and recruitment details are entered and acts as interface to access the system. Here the description of the role ‘Placement_Cell’ R₆={R₆, R₇, R₈, R₉} is {Ð₆, 1, 1, <sender, receiver>, generate forms, providing data from external entity}.

This is the main role, which is independent of other. It is activated once the system is started. It exhibits the behavior of receiver when it accepts the data from the external entity and shows the behavior of sender when sending the data from the user to the
respective agents

\( R_6 = \text{Company\_Database\_Storage} \) role manages the company details and keeps track of the notification date. Whenever there is a change in the system date, it detects it and depending on to the present conditions, it raises the request for notification. The description of role \( \text{Company\_Database\_Storage} \) \( R_6 \) is \( \{ R_7, R_A, R_C, R_B, R_S, R_T \} = \{ PD_R, 1, 1, \text{provider, receiver} \}, \text{<storage, processing for notification>, <write, read, modify>} \).

This role is partially dependent on the ‘Placement_Cell’. It is activated by placement cell and it stores the data in the database. It has the capability of writing, reading, and modifying the database. While checking the notification date of company, if there is a hit then it provides the details of that company for displaying the notice thus behaving as provider. Its behavior is receiver only when it receives the company details from ‘Placement_Cell’.

\( R_7 = \text{Placement\_Recorder} \) role is responsible for maintaining the previous placement detail such as the total number of B. Tech., M. Tech. and MCA student recruited by each company. The role description of ‘Placement\_Recorder’ \( R_7 \) is \( \{ R_T, R_A, R_C, R_B, R_S, R_ST, R_CC \} = \{ PD_R, 1, 1, \text{consumer, storing, <write, read, modify>} \} \).

‘Placement\_Recorder’ is dependent partially on ‘Job\_table\_Storage’. It should have the write, read, modify permission over the table as its capability. It shows the behavior of consumer when it gets the data from the ‘Job\_table\_Storage’ and processes the data to store in the database.

\( R_8 = \text{Job\_table\_Storage} \) role handles the recruitment details. Here the role description of ‘Job\_table\_Storage’ \( R_8 \) is \( \{ R_7, R_A, R_C, R_B, R_ST, R_CC \} = \{ PD_R, 1, 1, \text{consumer, sender, storage, <write, modify, read>} \} \).

This is partially dependent on the ‘Placement\_Cell’ which activates it when there is a requirement to enter the data in the job table. It has the permission to write, read, and modify on database which represents its capability. It exhibits the behavior of consumer only when it validates the recruitment data. It shows the sender behavior only when it sends required data to the ‘Placement\_Recorder’.

### 6.1 Implementation

Java is used for implementation of the case study because an agent requires intense use of threads and concurrent programming.

Here we have considered two environments:

Normal and Open.

**Normal**: It displays all students who have less than 2 and satisfy the company’s eligibility criteria.

**Open**: Here it shall display all the students list who satisfy the company’s eligibility criteria. It doesn’t consider whether they have got 2 or more jobs. It is done by Student\_Profile\_Holder agent.

### 6.2 Application of Algorithm on Case Study

**Step 1**: The first step is to find the total number of roles in the placement\_cell system. Here the total number of roles is 8.

\[ R_8 = \{ R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8 \} \]

**Step 2**: It applies the first criteria \( R_1 \) (Frequent interaction occurs) and creates cluster of roles that interacts frequently. In our case study the Placement\_Cell, Notifier and Display, have frequent communication among them so they belong to one cluster.

\[ C_1 = \{ \text{Placement\_Cell, Notifier, Display} \} \]

Here \( (R_5, R_3, R_1) \subseteq R_8 \) are sets of role such that \( R_5 \subseteq R_2 \subseteq R_1 \) and satisfying the condition \( \eta(R_3) \geq \eta(R_2) \geq \eta(R_1) \).

The size of cluster obtained from here is 3.

\[ \text{i.e. Size of } C_1 = 3 \]

**Step 3**: Having got the value \( C_1 = 3 \) from step 2 and the value \( R_8 = 8 \) from step 1 we have

\[ C_1 < R_8 \]

So the second criteria \( R_8 \) is applicable and forms the next set of cluster for optimal performance.

**Step 4**: Applying the second criteria \( R_8 \) According to second criteria \( R_8 \), we make a cluster for all roles that have close relationships among them. In our case study, the cluster of Notifier, Marker, Placement\_Recorder and Student\_Profile\_Holder role work together to generate the eligible student list for a specific company and maintains the student master table.

\[ C_R = \{ \text{Notifier, Student\_Profile\_Holder, Placement\_Recorder, Marker} \} = \{ R_2, R_3, R_5, R_4 \} \]

Here \( \eta : \rho(R_8) \) performance function satisfying the constraints, that by adding role in a cluster does not decrease the performance of the system.

The size of cluster obtained here is 4.

\[ \text{i.e. Size of } C_R = 4 \]

**Step 5**: So now we have to find out the intersection.
C₁ ∩ C₂ = {R₁, R₂, R₃} ∩ {R₂, R₃, R₇, R₄} = {R₂}

Here, if R₂ is taken out from both cluster the C₁ and C₂ and assigned R₂ to the intersection of cluster C₁ and C₂. The performance function satisfies the constraints, that by removing role (R₂) from C₁ and C₂ and assign to intersection of cluster C₁ and C₂ does not decrease the performance of the system. The size of intersection of cluster obtained from here is 1.

i.e Size of C₁ ∩ C₂ = 1
    Size of C₁ = 1
    Size of C₂ = 1

Here the size of (C₁ + C₂ + C₁ ∩ C₂) = 6 and the value of R₆ = 8
i.e. (C₁ + C₂ + C₁ ∩ C₂) < R₆

So the third criteria is applicable here and we have to apply third criteria R₃SR.

Step 6: According to R₃SR criteria all roles that are sharing the same resources shall be kept in one cluster. As Jobtable_Storage, Student_Profile_Holder, Company_Database_Storage, Placement_Recorder share the same database they form one cluster.

i.e. C₃SR = {Student_Profile_Holder, Company_Database_Storage, Jobtable_Storage}

C₃SR = {R₃, R₆, R₈}

Here η : ρ(R₃SR) performance function satisfies the constraint, that by adding role in a cluster it does not decrease the performance of the system.

Step 7: After applying the third criteria we compute C₁ ∩ C₂ ∩ C₃SR , C₃SR ∩ C₁, C₂ ∩ C₃SR as under:

C₃SR ∩ C₁ = null
C₂ ∩ C₃SR = {R₃}
C₁ ∩ C₂ ∩ C₃SR = null

The performance function satisfies the constraint, that by removing role (R₃) from C₂ and C₃SR and assign to intersection of cluster C₂ and C₃SR does not decrease the performance of the system.

Step 8: Map the individual cluster of C₁, C₂, and C₃SR to individual capable agent which is having desired characteristics depending upon the cluster.

i.e. C₁ = A₁ (A₁) (R₁, R₃ roles are assigned)

C₃SR = A₃SR (A₂) (R₆, R₈ role is assigned)
C₂ = A₃ (A₃) (R₆, R₇ role is assigned)

Mapping role cluster C₁, C₂, and C₃SR to agent A₁ A₂ and A₃ satisfy the role constraints. The mapping of role to agent does not violate the role constraint (R₅, R₆, R₇, R₈, R₉, R₁₀).

Step 9: Map the intersections of cluster C₁ ∩ C₂ and C₂ ∩ C₃SR to individual capable agent that is having desired characteristics depending upon the cluster intersection.

i.e.C₁ ∩ C₂ = A₄ (R₂ role is assigned)
C₂ ∩ C₃SR = A₅ (R₃ role is assigned.)

Mapping intersections of role cluster C₁ and C₂ to agent A₄. Mapping intersections of role cluster C₂ and C₃SR to agent A₅. Both satisfy the role constraint. The mapping of role to agent does not violate the role constraint (R₅, R₆, R₇, R₈, R₉).

6.3 Mapping Role To Agent

Form our case study we have found five agents. Figure 2 shows the mapping between the roles and agent. A₂, A₃, A₄ agent type have one to one correspondence between the role and the agent. But the other two agents A₁ and A₅ have more than one role within them due their similar function and high degree of interdependency hence they are grouped into respective agent type.

A₃

Figure 2. Mapping Role to Agent

Figure 3 shows agent class diagram of our case study, which is the high level conversation between the agent classes. Each agent class is defined by the roles they play during runtime. An agent class is represented by a rectangular box, which contains the name of the agent and the roles it plays. Each arc has a name that gives overall details of conversation.
In the above agent class the

F\textsubscript{1} → Store company detail
F\textsubscript{2} → Notify notice
F\textsubscript{3} → Get eligible student
F\textsubscript{4} → Handle recruitment detail
F\textsubscript{5} → Marks status

A\textsubscript{1}, A\textsubscript{2}, A\textsubscript{3}, A\textsubscript{4}, A\textsubscript{5} \rightarrow Agent class name and

R\textsubscript{1}, R\textsubscript{2}, R\textsubscript{3}, R\textsubscript{4}, R\textsubscript{5}, R\textsubscript{6}, R\textsubscript{7}, R\textsubscript{8} \rightarrow Role

Figure 4 shows the actual message passing between the agent classes. The agent A\textsubscript{2} stores the company details in the database as well as it checks for the notification date. Whenever it finds the notification date matching with the system date, it finds that the notice for that company must be displayed. The agent A\textsubscript{4} places a request for A\textsubscript{5} to get the eligible student list, but if agent A\textsubscript{5} sends refuses signal then the agent A\textsubscript{4} will change it’s role and start finding the eligible student list itself. If A\textsubscript{5} accepts the request then it will send accept acknowledgement to A\textsubscript{4} and list of eligible student to the agent A\textsubscript{1} for displaying purpose.

Table 1 illustrates the service model for Agent A\textsubscript{2}

<table>
<thead>
<tr>
<th>Name of Agent</th>
<th>A\textsubscript{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Name</td>
<td>Check_notification</td>
</tr>
<tr>
<td>Input</td>
<td>System date, notification date</td>
</tr>
<tr>
<td>Output</td>
<td>Set of company list</td>
</tr>
<tr>
<td>Pre-condition</td>
<td>Availability of notification date</td>
</tr>
<tr>
<td>Post-condition</td>
<td>Place request</td>
</tr>
</tbody>
</table>

6.4 Result
We have applied all the three criteria of clustering of roles and mapping role to agent. According to implementation it has been found that the performance shall be optimal only if intersections of the cluster are minimal. The result also shows that if there exist null values among the intersections, this will produce optimal performance of the system. We also found that the size of cluster also affects the system performance. The system performance shall improve when the cluster size is optimal depending on the proposed performance parameter.

7 Conclusions and Future Work
In this paper, a new methodology is proposed for optimize agent based systems performance. We found that the clustering of roles and mapping roles to agent is highly application dependent. So clustering and mapping roles to agent are essential phases to optimize system performance. For better clustering and mapping roles to agent, it is essential that system analyzer captures more specific requirements, analyzes the requirements from users’ and system point of view carefully. By doing this the system analyzer understands the interaction, relationship and dependency among roles and better clustering is achieved. It is found that the system performance is optimal when the intersections of the cluster are minimal.
In future, we wish to refine the proposed methodology and apply the same as on an application so as to establish it.

References: