Optimized Agent Based System Performance - A Role Oriented Approach

SOUMYA SURAVITA[1], PRABHAT RANJAN [1], R.K. SINGH [2] & A. K. MISRA[1]
[1]Department of Computer Science & Engineering,
[2] Electrical Engineering Departments,
Motilal Nehru National Institute of Technology,
Allahabad -211004, Uttar Pradesh, INDIA

Abstract: - In this paper, we propose a system performance-efficient clustering and mapping algorithm for agent-based system. Optimal cluster size is obtained by user-defined performance parameter (η).This 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 concept of roles has been advocated to model application domain agents, which evolve dynamically during their lifespan. These agents may acquire new roles and drop old roles in order to comply with the requirements, where as each role and their instances are associated during the whole lifespan. The concept of clustering of role and mapping the role to agent by analyzing Interaction among Roles, Resource Dependencies among Roles and Relationship among Roles is presented in this paper.

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

1 Introduction

Roles are essential concept in agent-oriented software engineering (AOSE). The role has the active property to cooperate with other roles for accomplishing the task. 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 [8,9].

The role concept is widely used in AOSE methodologies. 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 [1]. Moreover the implicit conflict among roles can also be identified. Identifying roles and mapping the role to an agent are essential phases in many proposed AOSE methodologies like GAIA [5], PROMETHEUS [4], ROADMAP [10] and TROPOS [3].

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 [6,7] is used as fundamental ingredient in the new proposed methodology. This paper concentrates mainly on the Role models.

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 (η) is considered for optimal clustering of role.

We optimize the system performance by minimizing the overall interaction, data transmission and competition of shared resource between roles/agents. The proposed clustering algorithm partitions the overall system roles/agents into several clusters. Optimal cluster size can be obtained by performance parameter (η).

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 droppes previous one and acquires the new role. The agent will play different roles at different time and in different conditions during its life cycle.

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.

The lifespan of a role instance is a single time interval between its acquisition and dropping by an agent and is represented as L_{Ri} for i^{th} instance of role R. The lifespan of a role R with an agent A, denoted as $L_{R(A)}$, is defined as an ordered set of one or more intervals during which the agents was attached with instance of R.

i.e. $L_{R(A)} = \{L_{R1}, L_{R2}, ..., L_{Rn}\}$

Where,

$_{R1, R2}$..... $_{Rn}$ is instances of the role R attached to agent A. The lifespan of an agent can be defined in terms of life it has acquired.

When we consider what kind of agents and what mapping between agents and roles we need, there

are some considerations proposed by Chen [2]:

- 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 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 the role instance that we are trying to add in cluster and map to agent already exists for the agent.
- Role Activation Exception: It is raised if a role cannot be activated on the agent because of its characteristics R_A and R_C.

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_I representing frequent interaction between the roles and make cluster of the role.

i.e. $C_I = \{R_{In}, \eta\}$

Where,

C_I 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

η : $\rho(R_I)$ is a performance function which defines the time of interaction 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_I is decided by performance parameter η.

3. If cluster size of C_I $< R_n$ 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

$R_{Rn} = \{R_{R1}, R_{R2}, ……..,R_{Rn}\}$ is the set of role which is performing similar nature of task, similar role capabilities and role dependencies.

η : $\rho(R_R)$ 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 η.

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}

$R_{SRn} = \{R_{SR1}, R_{SR2}, ……..,R_{SRn}\}$ is the set of role which is using shared resource.

η : $\rho(R_{SR})$ 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 characteristics depending upon the cluster.

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.

For better clustering and mapping roles to agent, it is essential that system analyzer captures more specific requirements, analyzes the requirements from users' point of view and system point of view carefully. By doing this the system analyzer understands the relationship and dependency among roles and better mapping criteria between roles to agents is achieved.

5 Conclusions and Future Work

In this paper, a novel methodology is proposed for optimized agent based systems performance. The clustering of role and mapping role to agent is highly application dependent. So clustering and mapping roles to agent are essential phases to optimize system performance. Optimal cluster size is obtained by user-defined performance parameter (η). It is found

that that the system performance is optimal when the intersections of the cluster are minimal.

We are currently working to refine the proposed methodology via implementation of real life case study.

References:

[1] Chiung Hui, Leon Lee and Alan Liu, *"A Method* for Agent-Based System Requirements Analysis", Proceedings of the IEEE Fourth International Symposium on Multimedia Software Engineering (MSE'02), 2002.

[2] C. W. Chen, *"An analysis method for cooperation issues in multi-agent systems"*, Master's thesis, National Chung Cheng University, 2000.

[3] Giunchiglia, J. Mylopoulos and A. Perini, *"The Tropos Software Development Methodology: Processes"*, Models and Diagrams. Technical Report No. 0111-20, ITC - IRST, Nov 2001.

[4] L. Padgham and M. Winikoff, *"Prometheus: A Methodology for Developing Intelligent Agents"*, Proceedings of the Third International Workshop on Agent Oriented Software Engineering, at AAMAS 2002. Bologna, Italy, July 2002.

[5] M. Wooldridge, N. Jennings and D. Kinny, *"The Gaia Methodology for Agent-Oriented Analysis and Design"*, Journal of Autonomous Agents and Multi-Agent Systems 3 (3), p285-312, 2000.

[6] P. Ranjan and A. K. Misra, *"An Enhanced Model For Agent Based Requirement Gathering and Pre-System Analysis"*, Proceedings of 13th Annual IEEE International Symposium and Workshop on the Engineering of Computer Based Systems (ECBS) 2006, p187-195, Potsdam, Germany, March 27th-30th, 2006.

[7] P. Ranjan and A. K. Misra, *"A Hybrid Model for Agent Based System Requirements Analysis"*, ACM SIGSOFT Software Engineering Notes, Volume 31, No. 3, May 2006.

[8] R. Depke, R. Heckel and J.M. Kuster, *"Improving the Agent_Oriented Modeling Process by Roles"*, ACM AGENTS'01, May 28-June1, 2001, Canada.

[9] R. Depke, R. Heckel and J.M. Kuster, *" Roles in Agent_Oriented Modeling"*, International Journal of Software Engineering and Knowledge Engineering, Volume 11, No. 3, 2001.

[10] T. Juan, A. Pearce and L. Sterling, *"ROADMAP: Extending the Gaia Methodology for Complex Open Systems"*, Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), p3-10, Bologna, Italy, July 2002.