Agent Oriented Software Testing – Role Oriented approach

N.Sivakumar
Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry, India

K.Vivekanandan
Department of Computer Science and Engineering
Pondicherry Engineering College
Puducherry, India.

***Abstract*—Several Agent Oriented Software Engineering (AOSE) methodologies were proposed to build open, heterogeneous and complex internet based systems. AOSE methodologies offer different conceptual frameworks, notations and techniques, thereby provide a platform to make the system abstract, generalize, dynamic and autonomous. Lifecycle coverage is one of the important criteria for evaluating an AOSE methodology. Most of the existing AOSE methodologies focuses only on analysis, design, implementation and disregarded testing, stating that the testing can be done by extending the existing object-oriented testing techniques. Though objects and agents have some similarities, they both differ widely. Role is an important attribute of an agent that has a huge scope and support for the analysis, design and implementation of Multi-Agent System (MAS). The main objective of the paper is to extend the scope and support of role towards testing, thereby the vacancy for software testing perception in the AOSE series will be filled up. This paper presents an overview of role based testing based on the V-Model in order to add the next new component as of Agent-Oriented Software testing in the agent oriented development life cycle.**

Keywords-Agent oriented software enginerring; Multi-Agent System; Role oriented testing.

I. Introduction

A software development methodology refers to the framework that is used to structure, plan, and control the process of developing a software system. A wide variety of such frameworks have evolved over the years, each with its own recognized strengths and weaknesses. Now an increasing number of problems in industrial, commercial, medical, networking and educational application domains are being solved by agent-based solutions [1]. The key abstraction in these solutions is the agent. An "agent" is an autonomous, flexible and social system that interacts with its environment in order to satisfy its design agenda. In some cases, two or more agents should interact with each other in a multi agent system (MAS) to solve a problem that they cannot handle alone. The agent oriented methodologies provide us a platform for making system abstract, generalize, dynamic and autonomous. This important factor calls for an investigation of suitable agent-oriented engineering frameworks and testing techniques, to provide high-quality software development process and products

Agent Oriented Software Engineering (AOSE) is an umbrella term in which several researches have been proposed on new varieties of metaphors, formal modelling approaches and techniques, and development methodologies and tools, specifically suited towards agent-oriented paradigm. Several AOSE methodologies [4] were proposed for developing software, equipped with distinct concepts and modelling tools, in which the key abstraction used in its concepts is that of an agent. Some of the more popular AOSE methodologies were MASCommonKADS (1996-1998), MaSE(1999), GAIA(2000), MESSAGE(2001), TROPOS(2002), PROMETHEUS(2002), ADLEFE(2002), INGENIAS(2002), PASSI(2002), AOR Modeling(2003).

Several AOSE methodologies were analysed and compared and found that the strong weakness observed from almost all the methodologies were, there is no proper testing mechanism for testing the agent-oriented software. Our survey states that the agent based software are currently been tested by using Object-Oriented (OO) testing techniques, upon mapping of Agent-Oriented (AO) abstractions into OO constructs[3]. However agent properties such as Autonomy, Proactivity, and Reactivity etc., cannot be mapped into OO constructs. There arises the need for proper testing techniques for agent based software.

Role is an important attribute of an agent that has a huge scope and support for the analysis, design and implementation of Multi-Agent System (MAS) [2]. A role can be defined as the capability enabler that exposes to the agent that plays it a set of actions [9]. Roles are created to do something and it has the responsibility of achieving specific system goals and subgoals. Roles provide a well-defined interface between agents and cooperative processes [10]. This allows an agent to read and follow, normative rules established by the cooperation process even if not previously known by the agent. The major motivation to introduce such roles is to increase the agent system's adaptability to structural changes.

The main objective of our paper is to propose a testing mechanism to test a multi-agent system based on agent's important mental state, the role.

II. Background And Related Work

A. Lifecycle Coverage of AOSE Methodologies

For designing and building software systems, several software development paradigms were proposed such as structural, procedural, declarative and object-oriented technique. Recently agent oriented paradigm is used for designing and building software systems which is considered to be an extension of object oriented paradigm. Agents are the software program which works on behalf of human to carry some task which has been delegated to it and take their own decision according to the requirement. Agent based systems are meant for solving complex problem that too in distributed environment.

A methodology is indented to provide guidelines at every stage of software development process such as requirement analysis, design, implementation and testing. Several AOSE methodologies were proposed in the literature and every methodology deals with analysis and design phase of the agent based software development and very little attention is made for implementation and testing. When compared to implementation, testing has been neglected overall by the methodologies. Based on the survey made on several AOSE methodologies, it is very clear that the existing AOSE methodologies does not support testing phase, stating that testing the agent system has been accommodated using the traditional and object-oriented testing techniques. Table.1 tabulates the AOSE methodologies and their corresponding lifecycle coverage [4].

TABLE I. L IFE CYCLE COVERAGE OF AOSE METHODOLOGIES

Sl.No	**Name of the Methodology**	**Life-Cycle Coverage**
1.	MAS-CommonKADS (1996)	Analysis Design
2.	MASE(1999)	Analysis Design
3.	GAIA (2000)	Analysis Design
4.	MESSAGE(2000)	Analysis Design
5.	TROPOS(2001)	Analysis Design Implementation
6.	PROMETHEUS(2002)	Analysis Design
7.	PASSI(2002)	Analysis Design Implementation
8.	ADELFE(2002)	Analysis Design Implementation
9.	INGENIAS(2002)	Analysis Design Implementation
10.	RAP(2003)	Analysis Design

B. *Testing in AOSE Methodologies*

The goal oriented testing methodology [5] contributes to the existing Tropos methodology by providing a testing process model, which complements the Tropos methodology and strengthens the mutual relationship between goals and test cases. The support for testing in prometheus methodology is limited to only debugging support [5].

Role is an important mental attribute of an agent and often agent changes its roles to achieve its designated goal. Roles are intuitively used to analyze agent systems, model social activities and construct coherent and robust teams of agents. Roles are a useful concept in assisting designers and developers with the need for interactions. Roles provide a well-defined interface between agents and cooperative processes. Their major motivation to introduce such roles is to increase the agent system’s adaptability to structural changes. Among the existing AOSE methodologies, Generic Architecture for Information Access (GAIA) methodology [6] and Multiagent Systems Engineering (MaSE) methodology [7] were role-based methodologies for development of multi-agent systems. The GAIA methodology models both the social aspect and agent internals aspect of MAS. The methodology covers the analysis and design phase. Role model and Interaction model are constructed in analysis phase. With reference to the analysis phase, the agent model, service model and acquaintance model are constructed in the design phase. During the analysis phase of MaSE[7], a set of roles are produced, that describes entities which perform some function within the system. Each role is responsible for achieving the goal.

III. PROPOSED WORK

A software is been tested at different abstraction level such as unit, integration, system and acceptance. The main objective of this paper is to propose an effective agent-oriented testing technique that suits specifically for an agent based system. The proposed testing technique is oriented towards role, which is one of the important state/attribute of an agent. Role is defined as a capability enabler that exposes to the agent that plays it a set of actions. The V-model [8] represented in figure 1 shows the development and its corresponding role based testing activity. The left branch of the V represents the specification flow, and the right branch represents the role oriented testing flow.

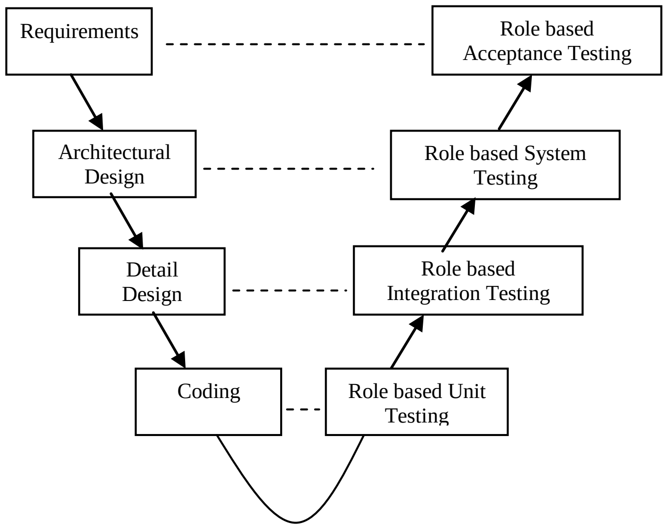

Requirements
Role based Acceptance Testing
Architectural Design
Role based System Testing
Detail Design
Role based Integration Testing
Coding
Role based Unit Testing

Figure 2. V-Model representation

A. *Role Model*

An attractive feature of agent-orientation is that it provides a powerful metaphor for describing, understanding and modeling information systems. A role is intended to enable software engineers to use it as a metaphor effectively to develop such cooperative information systems. The entire Role model is represented as Goal, Role and Responsibilities (GRRe).

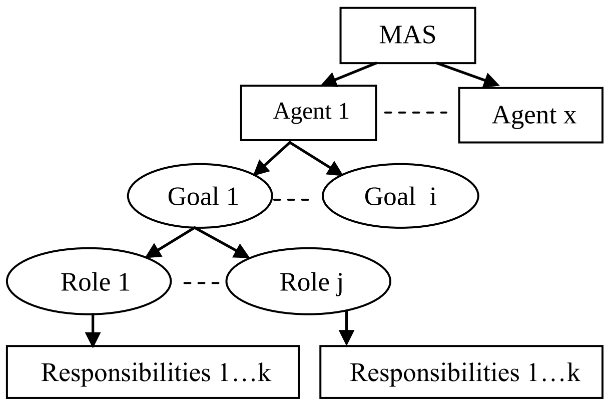


Figure 2. Role Model

A Multi-agent system is one which involves more than one agent that works in a coordinated way to achieve the overall goal of the system. Let us consider that, the number of agents involved in the MAS as ‘x’. Every agent in the MAS is intended to achieve a goal in turn many sub-goals. Let us assume that the number of goals of an agent is ‘i’.

The agent to achieve its goals and sub-goals has to play many roles within the agent as well as takes role in another agent i.e in certain cases the output from an agent might be the input to another agent. In such cases, the agent changes its current role and takes another role to accomplish the task given. Let us assume that the number of roles to achieve the agent’s goal is ‘j’. Moreover, a role is defined as the set of capabilities and expected behavior. In simple words, role is a set of actions or responsibilities to be handled by an agent. In this regard, a role can have ‘k’ number of responsibilities. Hierarchical representation of MAS down to responsibility is depicted in fig.3 and is mathematically been represented as below.

$$\sum_{x=1}^{m} A_n = \sum_{i=1}^{n} G_i \left(\sum_{j=1}^{o} R_j \left(\sum_{k=1}^{p} Re_k \right) \right)$$

Testing follows bottom-up approach in the proposed role-oriented testing. Test cases are written to test whether the responsibilities hold by the individual roles taken by the agent works as per the requirement.

Testing the responsibilities corresponding to their respective role ensures that the agent plays its role as per the intended goal. As long as the goals are getting satisfied, it is understood that the individual agent is performing well that suits the system. The following is the steps involved in role-oriented testing.

STEP 1: Select the Agent to be tested.

STEP 2: Identify Goals (G_i), Roles (R_j) and their corresponding Responsibilities(Re_k)

STEP 3: Design Role Model Diagram (A_x (G_i R_j Re_k))

STEP 4: Analyze role model (A_x(G_i R_j Re_k))

STEP 5: Define the interacting agents and situations.

STEP 6: Make the interacting agents as Pseudo agents.

STEP 7: Identify environmental factors pre-conditioning input trigger Re_k

STEP 8: Identify fulfillment criteria that satisfies Responsibility

STEP 9: Create test suite for Re_k corresponding to R_j

STEP 10: Run test cases

B. Role Oriented Unit Testing

Our testing approach focuses primarily on the smallest building block of the Multi-Agent System, the agent. The basic idea behind the unit testing in MAS is to verify whether the individual agent (unit) in isolation performs its responsibilities under various conditions.

Every individual agent has its own goal to be achieved and plans to do to fulfill the goal. In addition to goal and plan, role is one important mental state of the agent, which is defined as a set of capabilities and expected behavior. A role [9][10] can be represented as <Goal, Responsibilities, Protocol, Permissions>

- Goal, for which the agent playing this role is responsible
- Responsibilities, Which indicates the functionalities of agents playing such roles
- Protocol, which indicates how an agent playing such role can interact with agents playing other role
- Permissions, which are a set of rights associated with the role.

Let us consider an agent based university information system in which staff is one of the agent involved in the system. The goal of the staff agent is to be the best staff in the university. To achieve this goal, the staff agent has to take many roles such as teacher role, student counselor role, researcher role, administrator role etc., Every role has its own responsibilities, say for example, teacher role has the following responsibilities such as regular to class, handling classes properly, taking attendance, evaluating students, identifying weak students, etc.

Thus to test a single agent, scenarios (test cases) are to be developed to test whether all the responsibilities of the corresponding agent’s role got satisfied. When all the roles are been tested and working fine, then by default we claim that the goal of the agent is been tested, thereby the individual agent is tested successfully. XML notations are used to define roles and their capabilities.

Figure.3 represents all the semantic information about the agent such as goal, role, responsibilities and protocols.

```
<Agent>
  <Goal>
    <Role1>
        <Responsibilities 1>
                <Protocol>
             .
        <Responsibilities n>
                 <Protocols>
      </Role1>
      <Role n>
          <Responsibilities 1>
                  <Protocol>
                 .
          <Responsibilities n>
                   <Protocol>
         </Role n>
  </Goal>
 </Agent>
```

```
<Staff>
  <To be Best Staff>
    <Teacher>
        <Regular to class>
       <Handling classes>
       <Taking attendence>
       <Identifying weak students>
       <Evaluating students>
    </Teacher>

    <Researcher>
        <Involvement in research>
       <Publishing papers >
       <Usage of R&D activities>
       <Producing Ph.Ds>
     </Researcher>
 </To be Best Staff>
</Staff>
```

Figure 3. XML Notation for defining Role and its

C. Role Oriented Integration Testing

Integration testing in conventional software development is that to progressively integrate the tested units (module, program, procedure, function) either incrementally or non-incrementally, so as to check whether the software units in an integrated mode are working properly. In an object-oriented software development, integrated testing is to verify the interaction among classes (interclass). The relationships among classes are the basic characteristics of an object-oriented system and define the nature of interaction among classes and objects at runtime. Multi-agent system is a logical collection of agents that interact with each other in a way that implements the functionality of the system. After ensuring that the individual agent in isolation is working as per the requirement, the next immediate step is to integrate the agents involved in the MAS so as to test the interaction, and communication among agents. Scenarios in which one agent interacts with another agent so as to comply with the role it holds are identified and tested. The protocol involved in communication among agent is also tested during integration.

D. Role Oriented System Testing

System testing verifies that all elements (hardware, people, databases) are integrated properly so as to ensure whether the overall product met its requirement and achieved the expected performance. System testing also deals with non-functional requirements of the software such as recovery testing, security testing, stress testing and performance testing. System testing in agent oriented approach will test the complete functionality and test the system as a whole. Here the perceptions and actions of all the agents are tested as a whole by providing proper test cases.

E. Role Schema

Role schema provides a well-defined interface between agents and cooperative processes. This allows an agent to read and follow, normative rules established by the cooperation process even if not previously known by the agent. Their major motivation to introduce such roles is to increase the agent system's adaptability to structural changes. Role schema involves role name, agent name, goal to be achieved, description of the role, protocol and related activities, permissions and responsibilities.

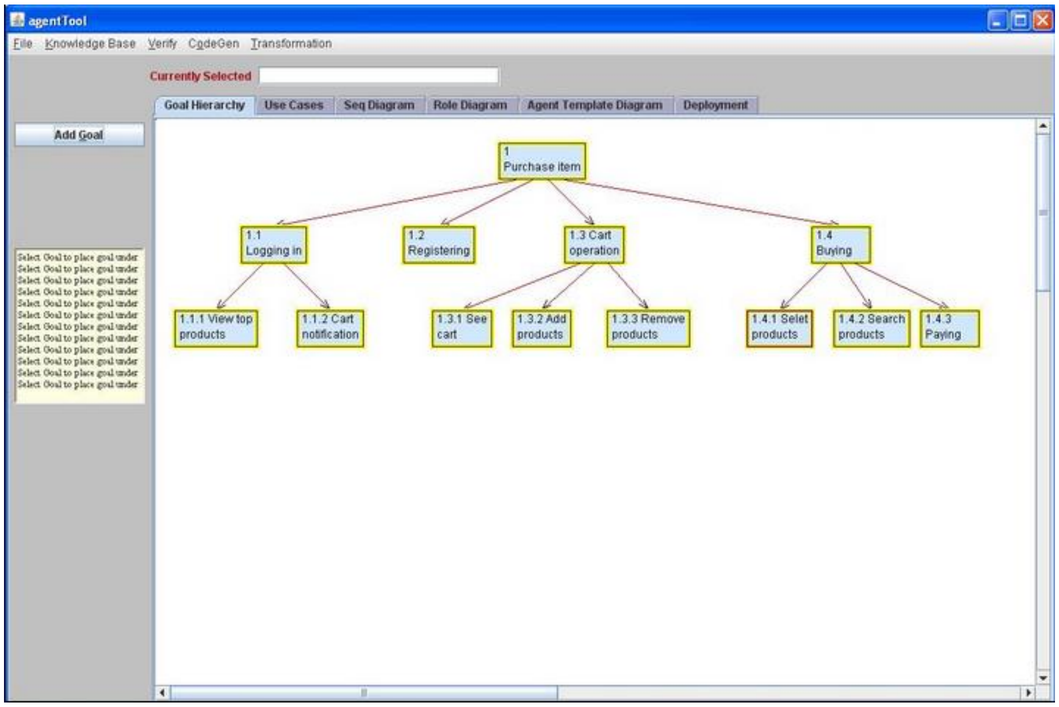
TABLE II. ROLE SCHEMA

Role Name : Teacher
Agent involved : Staff
Goal: To be the best staff in the university
Description: This role helps in identifying whether the teacher teaches well or not
Protocol and Activities: Be regular, Handling classes, Taking attendance, Evaluating students
Permissions: Read Request query, Result, Security policy**,** Change Result format // encrypt, Request format // decrypt
Responsibilities: *Activeness:* (Take attendance + Be regular+[Receive Questions from students+ Answer question from students]+ Teach subject+ Evaluate students + Submit result) *Completeness:* Lecturer is a good teacher

IV. CASE STUDY

To illustrate the role-based unit testing approach, an agent based online shopping system involving buyer agent, seller agent and bidder agent was developed using MaSE methodology. MASE is an iterative process. It deals the capturing the goals and refining the roles of an agent. It appears to have significant tool support. agentTool is a graphically based, fully interactive software engineering tool, which fully supports each step of MaSE analysis and design. Fig.3 shows the snapshot of the agentTool with which online shopping system is been analysed and designed. The analysis phase involves capturing goal, Applying use cases and Refining roles whereas the design phase involves Creating agent classes, Constructing conversations, Assembling agent classes and System design. Agent-based Online shopping system is been implemented using JADE(Java Agent Development

Framework), a software platform that provides basic middleware layer functionalities which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction.



Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under
Select Goal to place goal under

Figure 4. Goal Hierarchy Diagram using agentTool

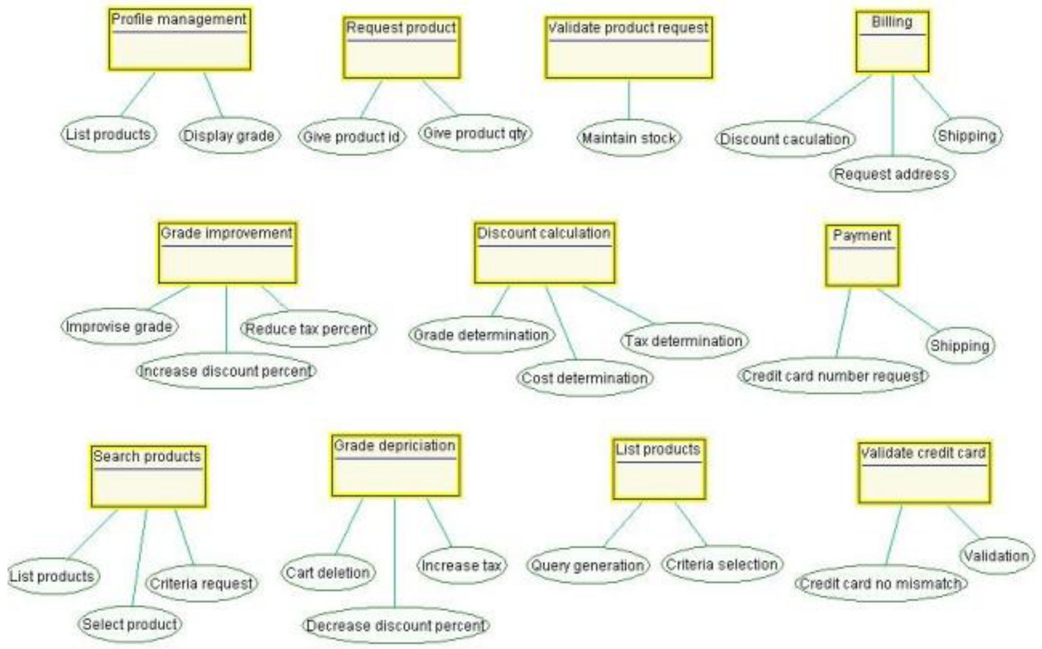
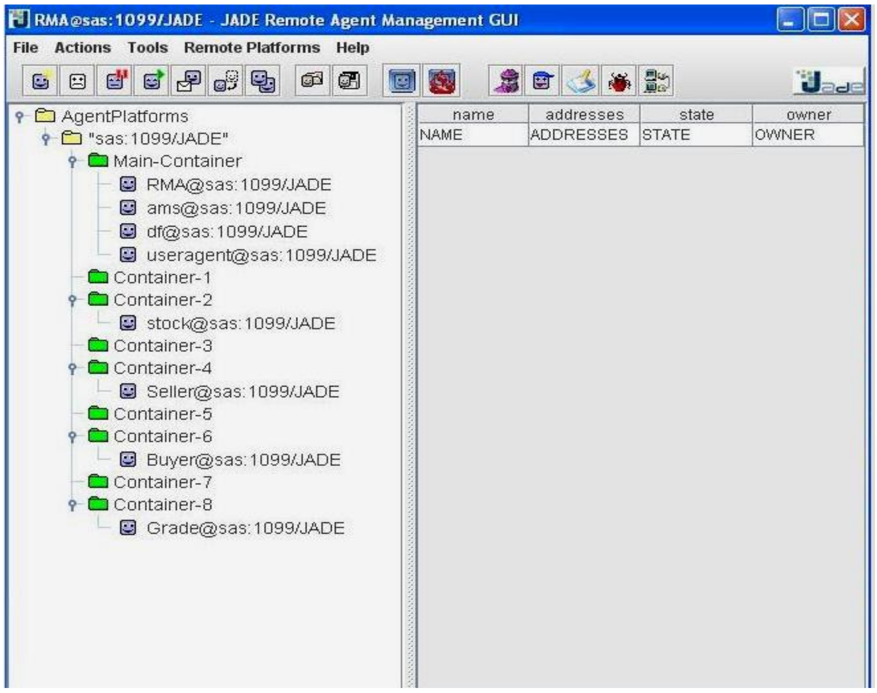


Figure 5. Role Diagram


RMA@sas:1099/JADE - JADE Remote Agent Management GUI
File Actions Tools Remote Platforms Help

AgentPlatforms
"sas:1099/JADE"
Main-Container
RMA@sas:1099/JADE
ams@sas:1099/JADE
df@sas:1099/JADE
useragent@sas:1099/JADE
Container-1
Container-2
stock@sas:1099/JADE
Container-3
Container-4
Seller@sas:1099/JADE
Container-5
Container-6
Buyer@sas:1099/JADE
Container-7
Container-8
Grade@sas:1099/JADE

name	addresses	state	owner
NAME	ADDRESSES	STATE	OWNER

Figure 6. JADE Development Environment

V. Role Oriented Testing Of Online Shopping System

Testing is an important activity of the software development process. Efficiency of the testing techniques determines the quality of the software. AOSE methodologies struggle to compete with the existing programming paradigm as there is a lack of proper testing mechanism.

This paper comes out with a testing mechanism specifically for the agent software based on agent's mental attribute, the role. The first step in the role-oriented testing is to identify the GRRe (Goal, Role, Responsibilities).

Let us consider a simple example derived from online shopping system. One of the goals of the online shopping system is to buy an item. To achieve this goal the agent has to take registrar role (registration process) and the Payer role (payment). Every role has its own responsibilities, say providing the registration detail is the responsibility of the registrar role and providing shipping detail is the responsibility of the payer role.

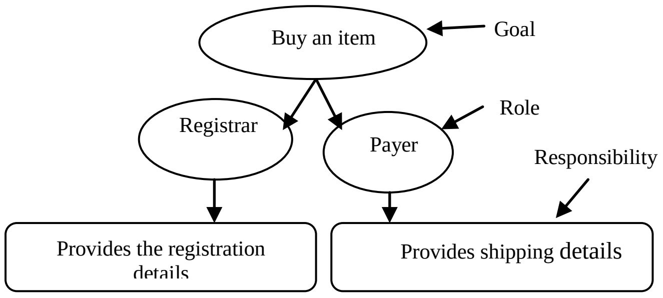


Figure 7. Sample Goal, Role, Responsibility representation

A. *Deriving Test cases*

According to our approach, the role of an agent comprises the logic of the test. As every role of an agent has number of responsibilities to get satisfied, the derivation of test case focuses on the responsibilities and thereby validates whether the role hold by the agent servers the purpose. Table 3 brings out the test case structure and some sample test case adopted for role oriented testing mechanism towards agent based online shopping system.

TABLE III. Test Case Structure For Role Based Testing

T.ID	TESTED AGENT	GOAL	ROLE	SITUATION	INPUT	EXPECTED RESULT	OBSERVED RESULT	RESULT
SE1	Seller Agent	Sell an item	Authenticator	Seller needs to authenticate elements	Username and Password	Authenticate and accept the Buyer	Buyer agent authenticated by seller	Passed

VI. Test Result Interpretation And Evaluation

In addition to the agent based online shopping system (P1) that we explored in the previous chapters we also developed few other agent based systems namely E-learning system (P2), Air ticket reservation system (P3), E-Novel System (P4) and E-Auction system (P5).

All the systems were developed using MaSE methodology. We intend to test all the agent based systems (P1 to P5) with the existing object oriented testing technique and with our proposed role based testing technique. We made a statistical comparison of object-oriented testing versus role-oriented testing techniques using the following metrics [11][12],

1. Defect Removal Efficiency (DRE),
DRE = No. of defects resolved / Total no. of defects at the moment of measurement.

2. Test coverage (TC),
TC= number of detected faults / number of predicted faults.

3. Test Case Effectiveness (TCE)
No. of defects detected using test cases*100/Total no of defects detected

4. Number of Tests Per Unit Size (TPUS)
Number of test cases per KLOC / FP where, KLOC represents Kilo Lines of Code & FP represents Function Point.

The above mentioned four metrics are calculated for five sample experimental projects P1 to P5. The calculated metrics are tabulated in Table.IV for further analysis and interpretation.

TABLE IV. COMPARISON OF OO TESTING VERSUS ROLE-ORIENTED TESTING

Projects	**DRE (%)**		**TC (%)**		**TCE**		**TPUS**	
	OO	***RO***	***OO***	***RO***	***OO***	***RO***	***OO***	***RO***
P1	57.53	78.50	52.30	64.61	63.23	80.95	21.42	37.52
P2	43.54	82.67	45.67	68.23	64.32	82.33	25.89	35.07
P3	65.43	86.38	46.21	65.13	65.23	84.23	23.88	40.98
P4	53.76	82.34	45.52	69.32	64.12	82.45	22.67	37.61
P5	58.45	87.65	47.23	72.56	67.23	85.23	27.13	39.04

From the above table, it is very clear that our proposed role oriented testing is more efficient for testing the agent-based system rather than existing object-oriented testing technique. For further analysis and comparison the following graphs are plotted from the table data. From the graph, we may come to a conclusion that for testing an agent based system, an agent oriented testing technique say, role oriented testing will be more appropriate and effective too.

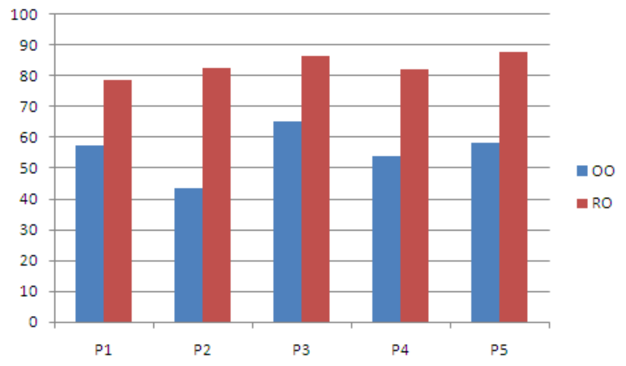


Figure 8. Defect Removal Efficiency

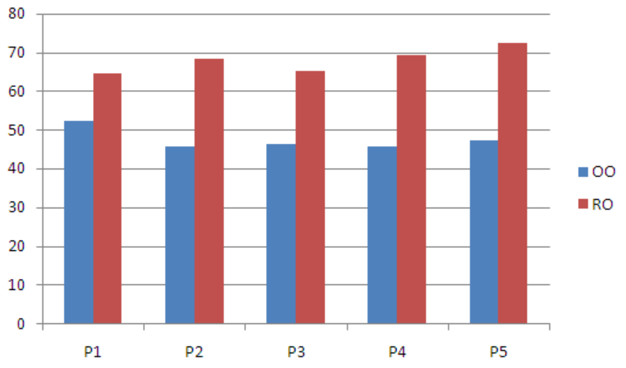
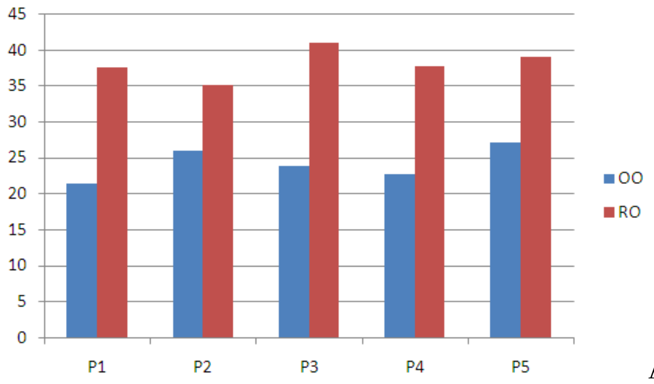


Figure 9. Test Coverage



A

Figure 10. Test Per Unit Size

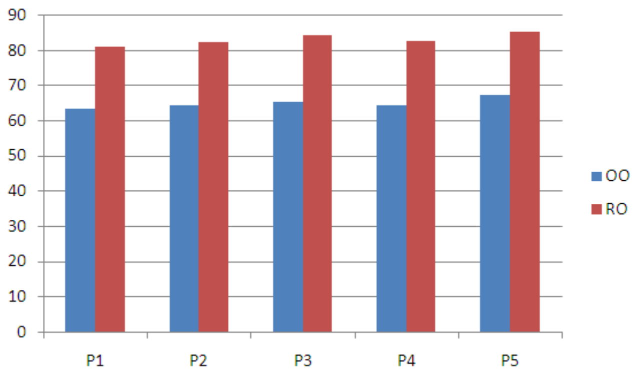


Figure 11. Test Case Effectiveness

VII. CONCLUSIONS

In this paper, a role oriented testing approach has been proposed for an agent system. The proposal helps the MAS developers in testing the individual unit of the agent based system namely, the agent. The testing mechanism is based on the role which is an important mental attribute of an agent. Every agent has its own role to perform so as to achieve its goal. Moreover the agents can even change their roles when the need arises so as to achieve the goal. Analysing the Goal-Role relationship, it is found that, as long as the agent performs its role properly, the goal of the system is been achieved by default. Thus testing whether the agent performs its role properly is a challenging task. This paved way for a role-oriented testing mechanism by which the role functionalities were tested by deriving appropriate test cases. To evaluate the proposed testing technique, 5 agent based systems were tested using object-oriented testing technique as well as role-oriented testing technique. The results of the testing techniques were tabulated and interpreted and it seems that the results are encouraging.

REFERENCES

[1] Shoham, Y. Agent oriented programming (Technical Report STAN-CS-90-1335) Stanford University: Computer science department, 1994.

[2] Giacomo Cabri, Letizia Leonardi, Luca Ferrari and Franco Zambonelli. Role-based Software agent interaction models: a survey. The Knowledge Engineering Review, Vol. 25:4, 397 419, 2010.

[3] Praveen Ranjan Srivastava , Karthik Anand V, Mayuri Rastogi, Vikrant Yadav, G.Raghurama. Extension of Object-oriented Software testing techniques to Agent Oriented software testing, in Journal of Object Technology, vol. 7, no. 8, November-December 2008, pp. 155-63.

[4] B. Henderson-Sellers and P. Giorgini. Agent-Oriented methodologies. Idea Group Inc., 2005.

[5] Duy Cu Nguyen, Anna Perini, Paolo Tonella. A goal-oriented software testing methodology. AOSE'07 Proceedings of the 8th international conference on Agent software engineering Springer –Verlag Berlin, Heidelberg-2008.

[6] B. Henderson, P. Giorgini. The Gaia Methodology for Agent-Oriented Analysis and Design Autonomous Agent and Multi-Agent Systems. 3, 285.312,2000 - 2000 Kluwer Academic Publishers.

[7] Wood, M. F. Multiagent system engineering: A methodology for analysis and design of muti-agent systems. Master thesis, School of Engineering, Air Force institute of technology, USA, (2000).

[8] Mailyn Moreno, Juan Pavon, Alejandro Rosete. Testing in Agent Oriented Methodologies. IWANN 2009, Part II, LNCS 5518, pp. 138–145, 2009.© Springer-Verlag Berlin Heidelberg 2009

[9] Haiping Xu, Xiaoqin Zhang, Rinkesh J. Patel. Developing Role-Based Open Multi-Agent Softwrae Systems. International Journal of Computational Theory and Practice Vol.2, No. 1, June 2007.

[10] Manjeet kumar. Roles and Ontology for Agent Systems. Global Journal of Computer Science and Technology Volume 11 Issue 23 Version 1.0 December 2011

[11] C. Wille, R. Dumke, and S, Stojanov. Quality Assurance in Agent-Based Systems Current State and Open Problems, Preprint No. 4. Fakultatfur Informatik, Otto-von-Guericke- Universitat, Magdeburg (2002).

[12] Daniel Galin. Software Quality Assurance. Pearson Education 2009.