Abstract— Agent-oriented techniques represent an exciting new means of analysing, designing and building complex software systems. They have the potential to significantly improve current practice in software engineering and to extend the range of applications that can feasibly be tackled. Agent-Oriented Software Engineering (AOSE) methodologies are proposed to develop complex distributed systems based upon the agent paradigm. The natural implementation for such systems has usually the form of Multi-Agent Systems (MAS). MAS are rapidly emerging as a powerful paradigm for modelling and developing distributed information systems. AOSE methodologies offer different conceptual frameworks, notations and techniques, thereby provide a platform to make the system abstract, generalize, dynamic and autonomous. Life cycle coverage is one of the important criteria for evaluating an AOSE methodology. Most of the existing AOSE methodologies focus only on analysis, design, implementation and disregarded testing, stating that the testing can be done by extending the existing object-oriented testing techniques. The main objective of this paper is to bring out the scope for testing in GAIA methodology. GAIA is one of the widely used AOSE methodologies for the analysis and design of MAS. Role is an important attribute that had a huge scope and support for the analysis, the design and the implementation phase of MAS development. Our work extends the scope and support of role towards testing, thereby the vacancy for software testing perception in the GAIA methodology can be filled up.

Keywords – Software Agent, Software Testing, Multi Agent System, Agent Oriented Software Engineering

I. INTRODUCTION

In recent years, agent-based systems have received considerable attention in both academics and industry. The agent-oriented paradigm can be considered a natural extension to the object-oriented (OO) paradigm [1]. Agents differ from objects in many issues which require special modelling elements but have some similarities. An object model does not capture some of the important knowledge associated with the agent, nor do we have any use of the semantics (e.g., inheritance in agent development etc.) especially, when we are developing multi-agent systems, the traditional analysis and design methodologies are poorly suited. This mainly depends on the fundamental mismatch between the abstractions they provide [2]. Hence, there is a need of new software development methodologies that support the design and implementation of organizations of agents that interact with one another in order to achieve some common or individual goal. Multi-Agent Systems are becoming more popular within the development mainstream because, as the name suggests, an agent aims to handle tasks autonomously with intelligence [3]. AOSE methodologies offer different conceptual frameworks, notations and techniques, thereby provide a platform to make the system abstract, generalize, dynamic and autonomous. Some of the AOSE methodologies are MaSE, Tropos, Gaia, Prometheus, Passi, Soda, Message etc [4],[5]. Although there is a well-defined OO testing technique, agent-oriented development has neither a standard development process nor a standard testing technique [6]. AOSE focuses in proposing methodologies that identifies the guidelines and the abstraction to be exploited in the various phases of agent-based software development such as analysis, design and implementation. However, very little attention has been paid towards the testing phase of agent-based software development process, stating that testing can be done by extending the object-oriented testing technique [6].

Multi-agent system testing are complex and challenging too. Existing software testing techniques (both conventional and object-oriented) cannot be applied over software agents as these agents are designed to be autonomous, proactive, collaborative and ultimately intelligent. One more important fact that traditional testing techniques cannot be applied towards software agents is that the agents communicate through message passing rather than method invocation. Moreover, agents can be programmed to learn and thus the successive tests with the same test data may give different result.

II. GAIA METHODOLOGY

Generic Architecture for Information Availability (GAIA) methodology aims to provide the designers with a modelling framework and several associated techniques to design agent-oriented systems [7]. Gaia is based on the organization abstractions and environmental abstractions. Gaia methodology was tailored specifically for the analysis and design of MAS.

A. Analysis phase.

The main goal of the analysis phase is to organize the collected specifications and requirements for the system-to-be into an environmental model, preliminary role and interaction models, and a set of organizational rules for each of the sub organizations composing the overall system.

1) The Organizations: At this step it is important to identify useful decomposition of the global organization into sub-organizations.
2) The Environment Model: The environmental model represents the environment (in terms of computational variables/resources) in which the multiagent system will be situated. The environmental model can be viewed as a list of resources, each associated with a symbolic name, characterized by the type of actions that the agents can perform on it, and possibly associated with additional textual comments and descriptions.

3) The Preliminary Role Model: The Preliminary Role Model identifies the “basic skills” that are required by the organization to achieve its goals, as well as the basic interactions that are required for the exploitation of these skills. The role model identifies the key roles in the system. Here a role can be viewed as an abstract description of an entity’s expected function. In other terms, a role is more or less identical to the notion of an office in the sense that “prime minister”, “attorney general of the United States”, or “secretary of state for Education” is all offices. Such roles (or offices) are characterized by two types of attribute:

- The permissions/rights associated with the role.
  A role will have associated with it certain permissions, relating to the type and the amount of resources that can be exploited when carrying out the role. In our case, these aspects are captured in an attribute known as the role’s permissions.

- The responsibilities of the role.
  A role is created in order to do something. That is, a role has certain functionality. This functionality is represented by an attribute known as the role’s responsibilities. In Gaia, responsibilities are divided into two types: liveness properties and safety properties. Liveness properties intuitively state that “something good happens,” that is, describes those states of affairs that an agent must bring about, given certain conditions. In contrast, safety properties are invariants. Intuitively, a safety property states that “nothing bad happens,” that is, that an acceptable state of affairs is maintained.

4) Preliminary Interaction Model: The Gaia interaction model captures the dependencies and relationships between the various roles in the MAS organization in terms of one protocol definition for each type of inter-role interaction.

5) The Organizational Rules: The preliminary roles and interaction models capture the basic characteristics, functionalities, and interaction patterns that the MAS system must realize, independently of any predefined organizational structure. In Gaia, the perspective on organizational rules is consistent with that on roles’ responsibilities—organizational rules are considered responsibilities of the organization as a whole. Accordingly, it is possible to distinguish between safety and liveness organizational rules.

Liveness rules define how the dynamics of the organization should evolve over time whereas the Safety rules define time-independent global invariants for the organization that must be respected.

B. Design phase.

The design phase includes the following sub-phases:

1) Architectural Design.
   Defines the overall architecture of the system, i.e., of the organizational structure, taking care that it accommodates all preliminary roles and interactions identified in the analysis phase, and taking care that the adopted structure facilitates the enactment of the organizational rules.

2) Detailed Design
   The detailed design phase is responsible for eventually identifying the agent model and the services model that, in turn, act as guidelines for the actual implementation of agents and their activities. The agent model specifies the agent types (a set of agent roles) and agent instances whereas the services model, specifies the main services (blocks of activities with their pre-and post-conditions) that agent types have to provide.

III. EVALUATING THE LIFE CYCLE COVERAGE IN GAIA

A methodology is the set of guidelines for covering the whole lifecycle of system development. Lifecycle coverage specifies what elements of software development are dealt within the methodology. Every methodology may have elements that are useful in several stages of the development lifecycle such as requirement gathering, analysis, design, implementation, and testing. Lifecycle coverage is an important evaluation criterion for a methodology because a detailed description of the activities included in the development lifecycle would enhance the appropriate use of a
methodology and increase its acceptability as a well-formed engineering approach. Gaia methodology is comprehensive with respect to lifecycle coverage. The Gaia methodology is based on a well-founded organizational metaphor and exploits in a clean and rational way a suitable set of organizational abstractions. Gaia covers analysis and design phase of MAS development. Gaia does not deal with implementation issues and considers the output of the design phase as a specification that can be picked up by using a traditional method or that could be implemented using an appropriate agent-programming framework. The testing stage is not at all covered by Gaia. Traditional testing techniques cannot be applied towards software agents as the agents communicate through message passing rather than method invocation. Existing software testing techniques (both conventional and object-oriented) cannot be applied over software agents as these agents are designed to be autonomous, proactive, collaborative and ultimately intelligent. There arises the need for testing phase in Gaia that brings out how an agent based system can be tested.

IV. TESTING IN GAIA

Gaia methodology mainly focuses on roles and the interaction among the roles. Hence, Gaia methodology can be termed as a Role-Oriented methodology as roles are used as the main abstraction of MAS analysis and design. In this context, to incorporate testing phase in Gaia, role-oriented testing technique will be more suitable. At present, there are no realizations of agent role in the implemented system beyond the analysis and design stage. The main objective of this paper is to extend the scope of roles towards testing the MAS thereby, testing phase can be incorporated in Gaia methodology. A role describes the external characteristic of an agent in a particular context [8]. Role can be identified in the system by following a Goal-oriented approach. An Agent may be capable of playing several roles, and multiple agents may be able to play the same role. An agent may change its role dynamically in order to achieve its goal with respect to the environment. Consider a human organisation such as a typical company. The company has roles such as “president”, “vice president”, and so on. Note that in a concrete realization of a company, these roles will be instantiated with actual individuals. There will be an individual who takes on the role of president, an individual who takes on the role of vice president, and so on. However, the instantiation is not necessarily static. Throughout the company’s lifetime, many individuals may take on the role of company president, for example. Also, there is not necessarily a one-to-one mapping between roles and individuals. It is not unusual (particularly in small or informally defined organizations) for one individual to take on many roles. For example, a single individual might take on the role of “tea maker”, “mail fetcher”, and so on. Conversely, there may be many individuals that take on a single role, e.g., “salesman”.

A. Role-Oriented testing

A role is a class that defines a normative behavioral repertoire of an agent. While roles are defined independently from groups, they must be played within groups [9]. Roles can be composed of other roles. Furthermore, roles can have acquaintance associations with other roles, denoting that interaction may occur among the instances of the related roles. 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, and 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.

The entire Role model is represented in figure 2.

V. CASE STUDY

Let us consider an agent based college management 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 counsellor role, researcher role, administrator role etc. Every role has its own responsibilities, say for example, teacher role has the following responsibilities such as being 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. After ensuring that the individual agent is working as intended, then the interactions among agents are tested. During interactions, an agent might change its role so as to achieve the system goal. Test cases have to be generated.
in such a way to test the association among the agents during interaction.
Agent based college management system is analysed and designed using Gaia methodology. The activities involved in analysis and design phase are discussed below

A. Analysis phase

1) Sub-organizations
There are clearly different organizations in time
- The academic section,
- The administrative section.
- Training and Placement section
- Student Grievance section

2) Environment
The environment is clearly
- A computational environment of students, staff (both teaching and non-teaching), etc.
- Filled in with class rooms, buildings, playgrounds, Laboratories, etc

3) Roles
There are clearly such roles such as
- “Teacher” (who teaches and evaluates students)
- “Student counselor” (who counsels the students)
- “Researcher” (who takes up and do certain researches)
Each with different permissions related to the environment (e.g., A teacher should not evaluate his/her relatives or friends) and with different responsibilities (teachers should be regular)

5) Organizational Rules
The system should clearly function according to some rules
- A staff should not also act as a teacher for his/her own children, or for those of his/her “friends”
- A student should follow the rules and regulations of the college.

B. Architectural design & Detailed design
The final organizational of the system may imply
- Multi-level hierarchies to function the college (Under principal each department should have an Head and under H.O.D (Head of the Department) each department should have various staff and under them comes the students)
- A structure to avoid cheating or any malpractices in the college.

Let us consider the staff agent. One of the goals of the staff agent is to be the best staff. To achieve this goal the agent has to perform roles such as teacher, researcher, student counselor etc. To ensure whether the goal has been achieved by the staff agent, we have to ensure that all the roles (teacher, researcher, counsellor) are appropriately carried out by the agent. Now to ensure whether the individual roles are carried out at suitable instance, the responsibilities held by the individual roles should be tested. Say for example, the responsibilities of a teacher are that, handling classes regularly, clearing student’s doubts, identifying weak students and monitoring the performance etc. Thus test cases have to be derived to test whether the classes are properly handled by the teacher, students are monitored, etc., Similarly all the responsibilities held by the corresponding role are tested using appropriate test cases. 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 held by the agent serves the purpose.

Table I. Sample Role Schema

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

4) Protocols
Protocols can be easily identified
- “Teacher should be regular”
VI. CONCLUSION

In this paper, a role oriented testing approach has been proposed for Gaia methodology. The proposal helps the MAS developers in testing the agents involved in the agent based system. 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 based on the environment. 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.

REFERENCES


