Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 15 Issue 6 Version 1.0 Year 2015
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Agent based Software Testing for Multi Agent Systems

By Manjeet Kumar & Dr. Rabins Porwal

Mewar University Rajasthan, India

Abstract- Software testing starts with verification and validation and fulfills the requirement of the customer. Testing can be done by automation tool like Win runner, QTP or manually. If we talk about manual testing it takes lot of time and manpower also so nowadays we are using automation software. When we talk about automation testing so the cost of such kind of testing is very high so each company cannot afford. In this paper we are presenting agent based testing which is helpful for both kind of testing. Multi-Agent Systems (MAS) are characterized by autonomous and collaborative behaviors [1, 2]. Developing such systems is a complex process. As a result, a methodology for developing MAS is highly necessary. In this paper, a methodology using roles and ontology for such a purpose is presented [2]. The functionality of roles is estimated in the various phases of the MAS development. It is based on an emphasis on the properties and behaviors associated with each agent in MAS.

Keywords: *software agent, muti agent system (mas), roles, ontology, bug, regression testing.*

GJCST-C Classification : *G.4 K.7.3*

Strictly as per the compliance and regulations of:

Agent based Software Testing for Multi Agent Systems

Manjeet Kumar [α] & Dr. Rabins Porwal [σ]

Abstract- Software testing starts with verification and validation and fulfills the requirement of the customer. Testing can be done by automation tool like Win runner, QTP or manually. If we talk about manual testing it takes lot of time and manpower also so nowadays we are using automation software. When we talk about automation testing so the cost of such kind of testing is very high so each company cannot afford. In this paper we are presenting agent based testing which is helpful for both kind of testing. Multi-Agent Systems (MAS) are characterized by autonomous and collaborative behaviors [1, 2]. Developing such systems is a complex process. As a result, a methodology for developing MAS is highly necessary. In this paper, a methodology using roles and ontology for such a purpose is presented [2]. The functionality of roles is estimated in the various phases of the MAS development. It is based on an emphasis on the properties and behaviors associated with each agent in MAS.

Keywords: software agent, muti agent system (mas), roles, ontology, bug, regression testing.

I. Introduction

Software testing is an exercise to simulate a system. Testing provides the program to get the desired goal. Testing analyze a program with the intent of finding problems and errors that measures system reliability. Testing cannot show the absence of bugs. It proceed the evaluation to SRS and is indication of software correctness. Testing consists of identification of required requirement and design as well as execution test of code. Along with the progress of computer network and communication, the research on MAS has become one of the hotspots in distributed AI [5,6]. Agents have been steadily moving into more and more significant applications [3]. Agents are capable to support more naturally the development of software systems whose components are heterogeneous and autonomous. These properties make agents ideally suited to applications in electronic commerce, virtual enterprises, and other open settings [2]. In these applications, agents must work in cooperation with traditional systems. Because of the importance of these applications and the risks of developing invalid systems, techniques for building agents must compare well with the techniques for building traditional application. There is, thus, a major need for industrial-strength approaches for engineering agent-based systems and to develop an approach. Mean while, the concept of role and role model [2]. This paper presents the significance of roles in MAS and proposes a method to realize its potential. The method supports dynamic binding between role and agent.

Agent technology is very useful in Internet computing. It is suitable for service oriented computing and interact with each other . The OO methodology is insufficient for developing agent-based systems because it cannot naturally represent the essential characteristics of agents, such as autonomous behavior, designated environment. For example, a web services application would be able to search for service providers from a web services registry and then dynamically establish a cooperation relationship with the service provider and request services and emergent behaviors resulting from the above characteristics commonly occur in MAS. These properties are not properly addressed in OO methodologies. Agent as a key concept is necessary for systematic, effective and efficient development of MAS.

Software agents represent an interesting paradigm to develop intelligent and distributed systems, because of their autonomy, proactiveness and reactivity; in addition, their sociality enables the distribution of the application logic in different agents that can interact together and with the host environment. In such a scenario interactions must be carefully designed and managed at run-time[5]. The concept of role has been adopted in different kind of agent approaches to flexibly manage interactions. In particular, the approach presented here can support and help the agent deciding the role.

The existing works on agent-oriented software engineering can be classified into two main camps: technique based methods and generalizing related methods. A technique specific method is based on a specific agent language or agent theory and/or aims at developing MAS to be executed on a specific agent platform or environment. Technique specific methods have a number of advantages [6,7]. They are practically usable, efficient and effective for certain types of software. These methods include guidance to development process and supporting languages without refer to any specific agent theory and implementations techniques.

Author α : Research Scholar, Mewar University.
e-mail: manjeet2005@gmail.com
Author σ: Rajasthan Associate Professor, ITS Mohan Nagar, Ghaziabad.
e-mail: rabinsp@rediffmail.com

II. Software Agents vs. Roles

If we talk about software industry, software agent can be used as a sunshade term for development. Task oriented robots, user bots, personal agents, autonomous agents and personal assistants are all software agents[3,4]. Those tha vast computer networks are known as soft bots[6]. In two general usages of the term agent are distinguished: one as a weak usage, two as a stronger and potentially more contentious usage. A weak agent is hardware or software based computer system with four key properties autonomy, social ability, reactivity and proactive ness. (i) Autonomous : An autonomous agent can operate without the direct intervention of anything. The internal state and goals should drive the agent to move its autonomous actions towards completion of the users or systems goals. (ii) Social ability: The ability to interact with other agents by way of some agent-communication language.(iii) Reactivity: A reactive agent can perceive its environment and respond in a timely fashion to changes that occur. (iv) Proactiveness: By being proactive, an agent does not simply act in response to platform. In [this attribute is a part of autonomy and is not considered unique. However, in [9,6], these are attributes "which agents should exhibit.

Several MAS methodologies such as MaSE [3,7] have adopted the concept of role (or role model) in analysis and design phases. In Gaia, a role is a quadruple <responsibilities, permissions, activities, protocols>. Role in MAS is defined as: (1) From the conception perspective, a role is a constraint under which an agent takes part in some interactions and evolves in a certain way. In MAS, an agent behaves under its bound roles. (2) From the implementation perspective, a role is an encapsulation of certain attributes and behaviors of the agent it is bound to. The characteristics of agent role relationship are: (i) Multiplicity: An agent can have more than one role at one time; (ii) Dynamicity: An agent can dynamically change its roles; (iii) Action ability (iv)Dependency: Roles are not isolated, they must be other roles related to an agent; (v)Role provides agent-to-agent interface (vi) Software reuse by roles: Roles provide a facility for efficient reuse.

III. Role-Based mas Development (rbmas)

An attractive feature of agent-orientation is that it provides a powerful metaphor for describing, understanding and modeling information systems that contain multiple autonomous active information processing agents and information sources and receivers. A role is intended to enable software engineers to use as a metaphor effectively to develop such cooperative information systems systematically through smooth and ordered transitions from models of the current system and users' requirements to the designs and implementations of new systems in an evolutionary way.

As shown in Fig1, an RBMAS considers such evolutionary development of information system as repeated cycles of modeling the current system and its operation environment, designing a new system to be executed in a new environment. This abstract model is then refined and realized using more concrete concepts to implement the new system. As this new system is subject to further modification as users' requirements change and the organizational environment and technology evolve, then a new cycle of modeling, design, refinement. Therefore, role's process of agent-oriented software development can be divided into three stages: (a) the analysis and modeling of the current system(b) modifications to the system hence the building of a model of the new system, (c) the implementation of the new system.

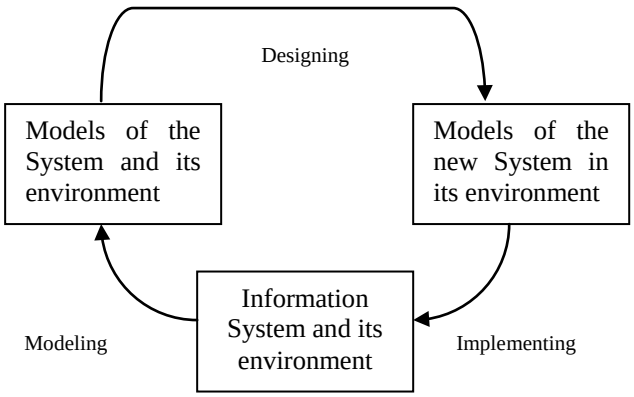

Figure 1: Evolutionary life cycle of MAS

Models of information systems play at least two important roles. The representation of the design of a new system so that properties of the new system can be inferred. The tester will be interested in different properties of the model. In the former case, software engineers will be interested in the following properties: (1) correctness in terms of whether the model accurately represents the real system to certain abstraction level; (2) comprehensibility in terms that complex systems can be represented in an comprehensible way. The following properties of the new system: (i) the correctness of the design in terms of whether users' requirements are met; (ii) the feasibility in terms of whether the design can be implemented and how costly the implementation will be; and, (iii) the sustainability in terms of its ease of maintenance and ease of modification for the evolution of the system in the future. Therefore, it is desirable to know how much modification to the existing systems required by the new design. It is also desirable to know the modifiability and reusability of the designed system in view of possible future development

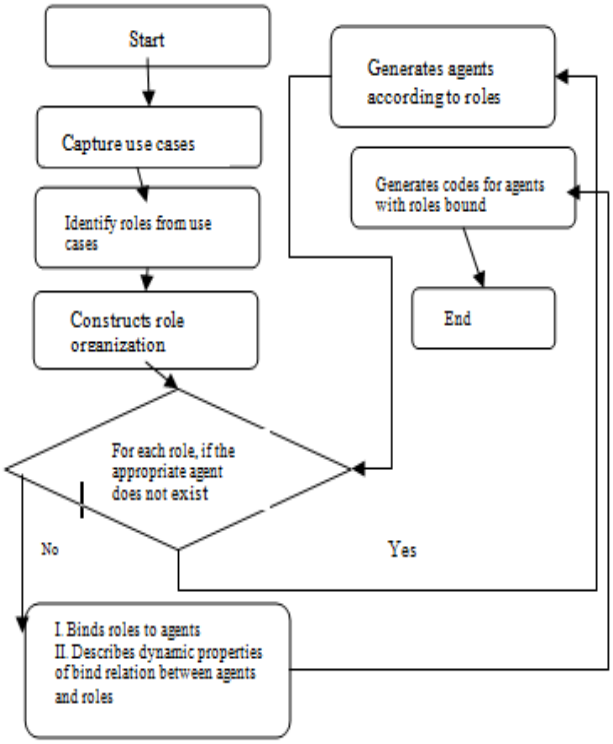

Figure 2 : Flow Chart : Analysis and modeling of roles

We propose a role-based modeling language tailored to (i) explicitly separate role from agent conceptually and linguistically; (ii) roles exist throughout the whole process of MAS development. The Flow chart for main development Process of analysis and modeling of roles is shown in fig 2.

a) Capture Use Cases

In fig 3, use case diagram for knowledge Transfer with an example is explained. Professor actor includes Explain Concept by Black Board use case or Explain Concept by Practical / Projector use case lecture in the class; student actor includes Grasp the Concept use case to understand the lecture. In some possible conditions, Explain Concept by Practical / Projector use case may be extended by Explain Concept by simulation use case. In [UML], detailed information about <<include>> and <<extend>> stereotypes is discussed.

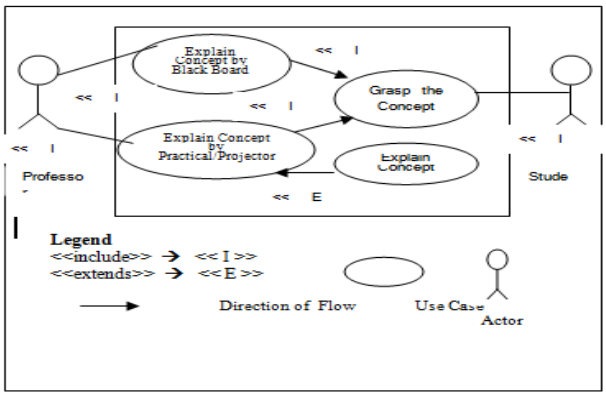

Figure 3 : Use Case Knowledge Transfer

b) Ascertain Roles

Roles can be identified from use cases [9, 8]. However, use cases are not sufficient for describing all the roles and events in the MAS. An assistant method is to check the words with *–er*, *-ist* or *–or* suffix in the requirement specification. Fig 4 shows an example notation of role.

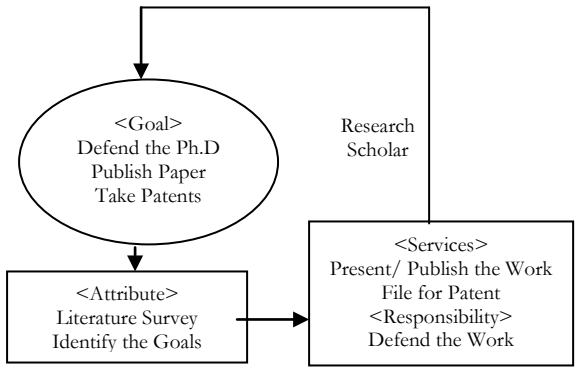

Figure 4 : Role Example of Research Scholar

c) Construct Role Organizations

Every role communicates and interacts with other roles. Besides, roles can be specialized or aggregated to other roles. Inheritance and aggregation associations respectively denote specialize/generalize and aggregate/decompose relations among roles. Fig 4 shows a role coordination chart triangle denotes inheritance relation, diamond denotes aggregation relation, and rectangle with a line on left-top corner denotes organization [3,5,7].

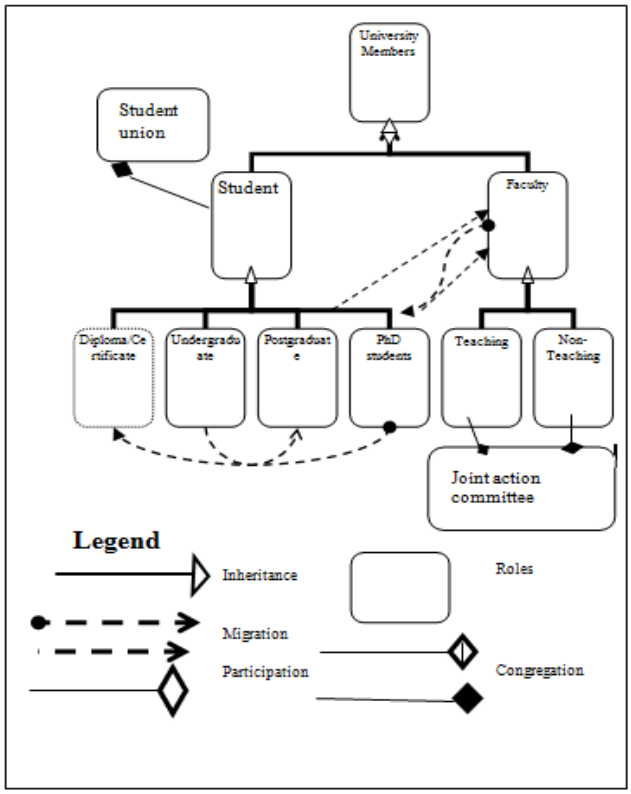

Figure 5 : Roles in a university

d) Bind Roles to Agents

For each role, the appropriate agent may belongs to to agent classes directly (fig 5). An agent has a name, attributes in the below figure Role Organization is given.An agent can change its roles dynamically. To make this property clear, we apply finite automata to describe agent's role transitions (role transition). All the roles are bound to the agent. Role binding describe initial binding of role to agent. The rectangle is a compact form of agent and the rectangle with semi-circle is a compact form of role

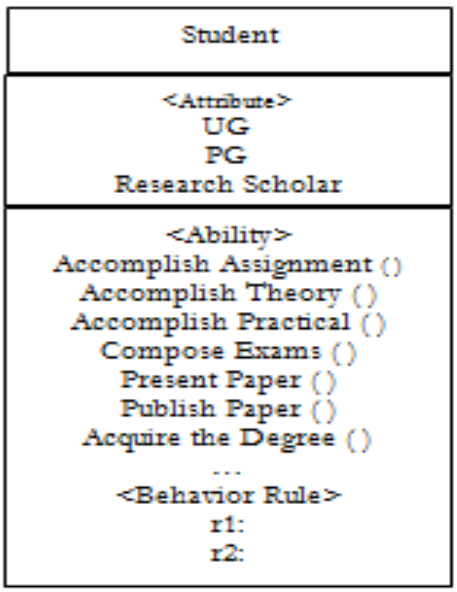

Figure 6 : Student Agent

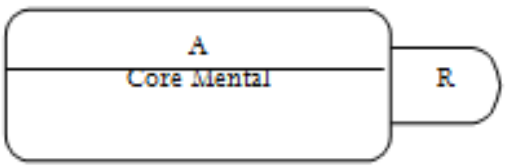

Figure 7 : Role Binding

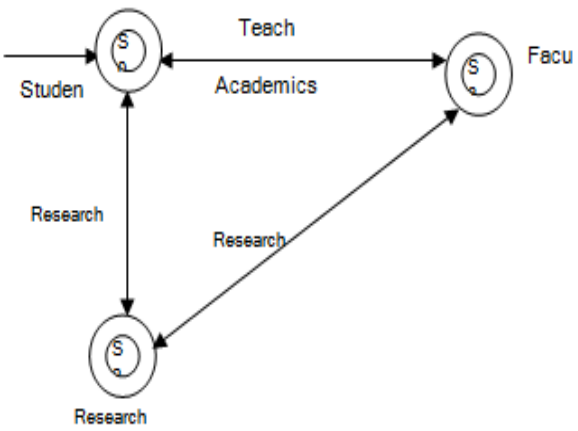

Figure 8 : Role

IV. Ontology Overview

Ontology is description of problem domain, where entities of the domain, its properties and its relations are described. In a sense it is vocabulary, thesaurus or taxonomy. It is a set of definitions of content-specific knowledge representation primitives (classes, relations, functions and constants). It represents the hierarchical structuring of knowledge about things by subcategorizing them according to their essential qualities.

The huge advantage of ontology is not in processing, but in sharing meaning, emergence and discovery of gaps and for improving a tacit knowledge transfer. Computer-based ontology provides formal and structured representation of domain knowledge. It is designed to serve as a raw material for computer reasoning and computer-based agents. It provides a formally defined specification of the meaning of those terms, which are used by agents during their interoperation. It is important, because agents can differ in their understandings of environment, goals capabilities, but they can still interoperate in order to perform a common task.

a) Agent Communications and Ontology

Common agent languages hold the promise of diverse agents communicating to provide more complex functions across the networked world. Indeed, as agents grow more powerful, their need for communication increases. The two agent communication languages with the broadest uptake for exchanging information and knowledge are KQML [3] and FIPA (Foundation for Intelligent Physical). The unproductive "standards war" scenario that might have arisen at one point seems now to have been avoided, with the most active participants supporting the FIPA effort, which incorporates many aspects of KQML [3]. FIPA standardization effort, seeks to address interoperability concerns through a sustained program. This is one area in which the visibility of agent technology is strong, with some of the most active take-up efforts from early adopters as, for example, is illustrated in [2]. Despite their merit, KQML and FIPA ACL only deal with agent-to-agent communication. An agent is understood as something that can act on behalf of a human or an organization.

Communication can be understood also as main sensors for software agents, since it is how agents can learn, share knowledge and interact with their environment, by communication, [FIPAACL] based communication is meant, where as content language RDF [RDFW3C] or OWL [OWLWEB] is used. Commercial technologies like WSDL, XML-RPC, SOAP or P2P came out of MAS research area. For example WSDL and UDDI can be understood, as a subset of what FIPA ACL is capable, however for most of applications features of XML-RPC or for most

complicated interaction SOAP and WSDL is sufficient. Agents can move forward only when they incorporate existing commercial communication technologies such as XML-RPC, SOAP and WSDL, and thus they will be able to communicate within the user and other existing software systems.

b) Ontology in MAS

Ontology defines the meaning of the terms in used content language and the relation among these terms. It ensures that the agents ascribe the same meaning to the symbols used in the message. Using ontology not only allows communication between agents but also gives the possibility for agents to reason about the concept.

Ontologies are dependent on used content language. In MAS ontologies are usually simple. The best-known implementation of ontology is in JADE agent system. Real Java classes with properties represent ontology classes. Instances of classes are individuals – information that can be stored or communicated. However UML or object oriented ontology is not sufficient because, multiple inheritance, inverse properties and other features present in RDF or OWL can not be used.

V. Conclusions and Future Work

Roles represent system goal and constrain agents' behavior. They exist throughout all phases from analysis to implementation, and this is the main difference to previous works on role both in OO and in agent orientation (AO) . Such a model can enable a natural realization of dynamic bindings between agents and roles. Besides, the RBMAS method generates roles from use cases. This prevents jumping from abstract use cases to concrete entities.

The paper presents the software infrastructure introduced into the agent service framework in order to support ontology design, implementation, and management. Software ontology's are a high level abstraction which is very useful for identifying the concepts of a problem domain, to define their relation, and to reason about them. Ontologyies give an added value to the interaction among software agents since they provide facilities to define a communication and to validate messages.

References Références Referencias

1. Agent Cities Consortium, Agent Cities.NET Project IST200028384,2002,
2. Andersen, E. (Egil), *Conceptual Modeling of Objects: A Role Modeling Approach*, PhD Thesis, 97.
3. http://www-ksl. kst/what-is-an-ontology.html
4. Bates, J. (1994), "The Role of Emotion in Believable Characters", *Comm of the ACM* 37 (7),
5. Bauer, B., Muller, J.P., Odell, J., Agent UML: a formalism for specifying multiagent software systems, in Agent-Oriented Software Engineering, Wooldridge, M, Editor. LNCS, Vol. 1957, 2001: Springer, pp. 91∼103.
6. Boella, G., van der Torre, L.: Attributing mental attitudes to roles: The agent metaphor applied to organizational design. In: Proc. of ICEC'04, IEEE Pres 04
7. Boella, G., van der Torre, L.: Organizations as socially constructed agents in the agent oriented paradigm. In: Procs. of ESAW'04, Berlin, Springer Verlag (2004)
8. Boella, G., van der Torre, L.: Groups as agents with mental attitudes. In: Procs. of AAMAS' 04, ACM Press (2004) 964–971
9. Boella, G., van der Torre, L.: Regulative and constitutive norms in normative multiagent systems. In: Procs. of KR'04, AAAI Press (2010)
10. Cabri, G., Ferrari, L., Leonardi, L.: Agent role-based collaboration and coordination: a survey about existing approaches. In: IEEE Systems, Man and Cybernetics Conference. (2008)
11. Chandra K. Sekharaiah , Md Abdul Muqsit Khan, U. Gopal Affective Computing: Next Generation AI Software Systems Icfai Journal of Information Technology, Vol. 3, No. 4, pp. 61-74, Dec 2007
12. Chandra K. Sekharaiah , Md Abdul Muqsit Khan, U. Gopal Computer Ergonomics: Relooking at Machines Vs. Environment International Ergonomics Conference HWWE IIT Guwahati Dec 2009
13. K. Chandra Sekharaiah, D. Janaki Ram, Mohd. Abdul Muqsit Khan The Peculiarities of Software Composition Models Journal of Digital Information Management (ISSN 0972-7272) Volume 3 Issue 3 Online Aug Print September 2009
14. Chandra K. Sekharaiah , Md Abdul Muqsit Khan, U. Gopal Obstacles to Machine Translation International Journal of Translation Aug 2006
15. Giovani Caire, JADE Tutorial Applicationdefined Content Languages
16. DARPA, DAML Website, 02, www.daml.org/
17. FIPA, FIPA Specification ACL Message Structure, 2008

This page is intentionally left blank