Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

Goal Oriented Acceptance Testing For Multi Agent System: V-Model Extension

Mandeep Kaur Balraj Singh Amandeep Kaur

***Abstract* -- Agent based Software Engineering, initially derived from Artificial Intelligent (AI), is now becoming increasingly popular among software engineers to develop modern and complex intelligent systems. Agent oriented systems contains intelligent agents that can perform a task autonomously. They are goal oriented extension of objects. In the recent years, with the emergence of AOSE, trails of various traditional Object oriented approach are being applied on it, to make it more and more acceptable in Software Industry. Acceptance testing is an integral part of traditional testing and it has drawn the interest of various researchers who are working on AOSE concept. No formal acceptance testing technique has been proposed yet for AO systems. The paper proposes a formal way of conducting Acceptance testing for agent oriented system by extending the popular V-Model for software testing. A two steps testing approach is proposed and a new phase "Goal Oriented Acceptance Testing" is added in V-Model. Goal Oriented Acceptance Testing lies on the demarcation of Internal and External tests. A tester from the developer team performs Goal Oriented Acceptance Testing on user's end. Once the Goal Oriented Acceptance Testing is passed, the user can go for general acceptance testing with non-agent-based and non-technical tests for his own satisfaction.**

***Keywords* –agent; software agent; testing; acceptance testing; goal oriented acceptance testing**

Mandeep Kaur
Department of Computer Science and Engineering,
Lovely Professional University
India

Balraj Singh
Department of Computer Science and Engineering,
Lovely Professional University
India

Amandeep Kaur
Department of Computer Science and Engineering,
Lovely Professional University
India

I. Introduction

Agent-Oriented Software Engineering is a programming paradigm where the software agents is the centeric idea behind construction of the software is centered-around the concept of software agents. They could be taken as

abstractions of objects. In a way specific to its class of agents, exchanged messages are interpreted using receiving agents. At its core, in contrast to object-oriented programming which has objects, AOP has externally specified agents [1].

A. *Properties of an Software Agent*

By an agent-based system, we mean one in which the key abstraction used is that of an *agent*. By an agent, we mean a system that enjoys the following properties [2]:

- *Pro-Activeness*: agents are able to exhibit goal-directed behaviour by taking the initiative and do not simply act in response to their environment.
- *Autonomy*: agents encapsulate some state, and make decisions about what to do based on this state, with no inference of human or other system
- *Social Ability*: agents interact with other agents via some kind of agent-communication language, and typically have the ability to engage in social activities in order to achieve their goals.
- *Reactivity*: agents are situated in an environment, are able to *perceive* this environment, react to the changes occurring in the environment due to controllable or uncontrollable parameters.

B. *Tropos*

An AOSE methodology, Tropos, which covers the whole software development process and is based on two key ideas [3]:

- First, from early analysis down to the actual implementation, the notion of agent and all

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

related mentalistic notions that are used in all phases of software development.

- Second, the kind of interactions that should occur between software and human agents, Tropos covers also the very early phases of requirements analysis, thus allowing for a deeper understanding of the environment where the software is operational.

Tropos methodology spans five phases:

- *Early requirements*, concerned with the problem understanding by studying an organizational setting where the intended system will operate
- *Late requirements*, where the intended system is described with relevant functions (hard goals) and qualities (soft goals) and within its operational environment. The intended system is introduced as a new actor.
- *Architectural design*, where the system's total architecture is defined in terms of interconnected through data, control, subsystems and other dependencies. More system actors are introduced.
- *Detailed design*, defines the behaviour of each architectural component in more detail including specification of communication and coordination protocols. Agents' beliefs, capabilities, and goals are specified in detail using existing modelling languages like UML or AUML, along with the interaction between them should occur between software and human agents.
- *Implementation*, during this phase, the Tropos specification, produced during detailed design, is transformed into a skeleton for the implementation. This is done through a mapping from the Tropos constructs to those of a target agent programming platform, such as JADE.

C. *Test type*

There are four types of testing: Agent testing, Integration testing, System testing and Acceptance testing. The objectives and scope of each type is described as follows:

- *Agent testing*: The smallest unit of testing in agent-oriented programming is an agent. Testing a single agent consists of testing its inner functionality and agent's capabilities to fulfil its goals and to sense and effect the environment.
- *Integration testing*: An agent has been unit-tested; we have to test its integration with existing agents. Integration testing make sure that a group of agents and environmental resources work correctly together which involves checking an agent works properly with the agents that have been integrated before it and with the "future" agents that are in the course of Agent testing or that are not ready to be integrated.
- *System testing*: Agents may operate correctly when they run alone but incorrectly when they are put together. System testing involves making sure all agents in the system work together as intended. Specifically, one must test the interactions among agents (protocol, incompatible content or convention, etc.) and other concerns like security, deadlock.
- *Acceptance testing*: Test the MAS in the customer execution environment and verify that it meets the stakeholder goals, with the participation of stakeholders.

D. *Goal type*

Different perspectives give different goal classifications. For instance, classify agent goals in agent programming into three categories, namely perform, achieve, and maintain, according to the agent's attitude toward them.

Goals are classified into the following types according to the different phases of the process:

- *Stakeholder goals*: Represent stakeholder objectives and requirements towards the intended system. This type of goal is mainly identified at the early requirements phase of Tropos.
- *System goals*: Represent system-level objectives or qualities that the intended system has to reach or provide. This type of goal is mainly specified at the late requirements phase of Tropos.
- *Collaborative goals*: Require the agents to cooperate or share tasks, or goals that are related to emergent properties resulting from interactions. This type of goal can be called also as group goal, and they often appear at the architectural design phase of Tropos.
- *Agent goals*: Belong to or are assigned to particular agents. This type of goal appears when designing agents.

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

E. *Goal-oriented testing*

Tropos integrates testing by proposing the lower branch of the V and a systematic way to derive test cases from Tropos modelling results. The left branch of the V represents the specification stream, and the right branch of the V represents the testing stream where the systems are being tested (against the specifications defined on the left-branch). The V-Model is a representation of the system development process, which extends the traditional waterfall model. Tropos guides the software engineers in building a conceptual model, which is incrementally refined and extended, from an early requirements model to system design artefacts and then to code, according to the upper branch of the V. One of the advantages of the V-model is that it describes not only construction stream but also testing stream (unit test, integration test, acceptance test) and the mutual relationships between them.

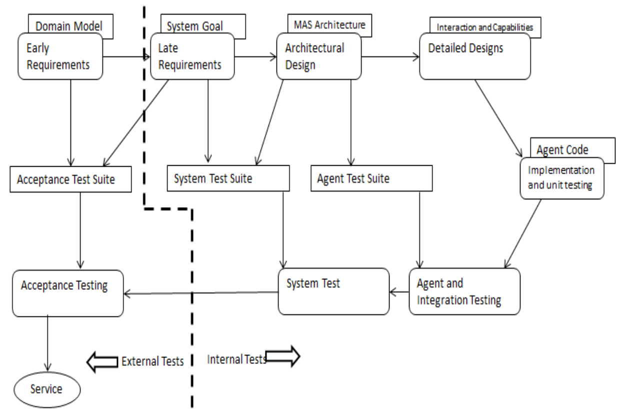


Fig 1: V-Model of Goal-Oriented Testing

Two levels of testing are distinguished in the model. At the first level of the model (external test executed after release), stakeholders (in collaboration with the analysts), during requirement acquisition time produce the specification of acceptance test suites. These test suites are one of the premises to judge whether the system fulfils stakeholders' goals. At the second level (internal test executed before release), developers refer to: goals that are assigned to the intended system, high-level architecture, detailed design of interactions and capabilities of single agents, and implement these agents.

From the systematic literature review, it has been noted that there had been very less attention given to formal Acceptance Testing of Agent Oriented System. Most of the things have been done for Agent Testing, Integration Testing and Unit Testing. So, the problem that study deals is Acceptance Testing of Agent Oriented System, which is still and area of concern. Confidence building of users and developers in autonomous agents is the primary goal of testing MAS.

II. Proposed System

As we have seen in the above figures and graphs, that an AO system can work well on developers end but may fail on user's end, due to agent's autonomous specifications. User is complete layman on the technical issues of agent and its working. So, it proposed that Acceptance testing should be on two levels:

- Once the system is installed on uses side, a member/tester from development team must visit the site and conduct an in-depth technical acceptance testing to ensure that all agents are working correctly on the user side also, according to the specifications. These should be those technical aspects that user may ignore or may not know. This is what is referred as Goal Oriented Acceptance Testing.
- Once the Goal Oriented Acceptance testing is passed and it is made sure that all agents are working correctly in user scenario also, then second level of acceptance testing must be conducted by user. This would be general acceptance testing as conducted in all other paradigms to for the user satisfaction. This level of acceptance testing will not include details about agents and their automations. User will just check the AO system is meeting his general requirements.

To make this "two level acceptance testing" successful, V-Model of testing have been extended and an addition step of "Goal Oriented Acceptance Testing" have been added.

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

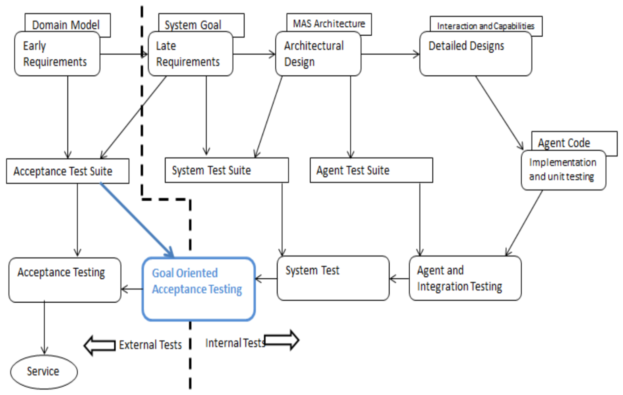
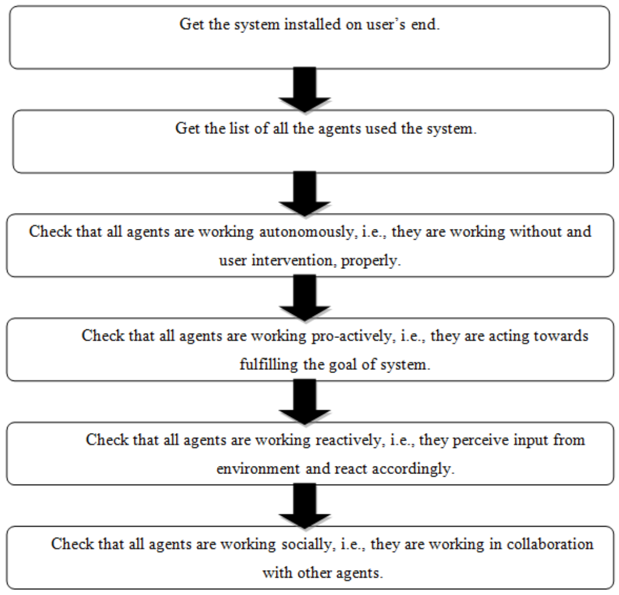


Fig 2: Extension for V-Model for Goal Oriented Acceptance Testing

In the extension for V-Model, an additions phase called "Goal Oriented Acceptance Testing" has been placed on the demarcation of Internal and External Tests. This is because Goal Oriented Acceptance Testing is done by a tester who is a part of internal development team, but it is done at user's end which is an external place for him.

For Goal Oriented Acceptance Testing, the tester must follow Fig 3.


Get the system installed on user's end.

Get the list of all the agents used the system.

Check that all agents are working autonomously, i.e., they are working without and user intervention, properly.

Check that all agents are working pro-actively, i.e., they are acting towards fulfilling the goal of system.

Check that all agents are working reactively, i.e., they perceive input from environment and react accordingly.

Check that all agents are working socially, i.e., they are working in collaboration with other agents.

Fig 3: Flowchart for Goal Oriented Acceptance Testing

The V-Model is a representation of the system development process, which extends the traditional waterfall model. Tropos guides the software engineers in building a conceptual model, which is incrementally refined and extended, from an early requirements model to system design artefacts and then to code, according to the upper branch of the V. With an added phase of Goal Oriented Acceptance Testing, AO systems will perform better on user's side and both developers and users will gain confidence on AOSE.

III. Implementation

A Jadex based game called Hunter Prey was downloaded for testing. This game is freely available with its source codes on Jadex website [4]. The game Hunter Prey was executed in Eclipse IDE [5]. The game had some specifications:

- The hunter prey scenario consists of two kinds of creatures living in a grid world.
- The basic task of hunters is to chase, while preys move around looking for food.
- Both kinds of creatures have to act autonomously in the environment on basis of their current local view, experiences made in the past and communications with others. Besides hunter and preys the environment accommodates other passive world objects.
- On the one hand there are trees on many squares that prohibit creatures running on such fields and on the other hand little plants grow at random squares at the map.
- These plants can be eaten by the preys if they are on the same field.
- The scenario is round-based with a fixed time slot for each round. This means that all creatures in the world have to issue their next action (moving to some adjacent square or eating something on the current square) with that round time.
- If no action is announced no action will be executed.
- The environment will decide in each round if an action succeeds or fails.

Finally, Zeta Test [6] was used to create and execute test cases on the game.

Fig 4: Snapshot of Hunter Prey Game

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

The game is also available in executable form on the web server of Jadex website [7].

When the game was executed on the user end, it was noted that some the specifications of the game were not met. Not all agents were working properly. But the same game was running perfectly on the web server, meeting all the specifications.

So, it was some compatibility error which was occurring on user's end. So, this required a Goal Oriented Acceptance testing. Some acceptance test case based on Game and agent scenarios were designed.

The test cases were then feed in Zeta Test software and executed.

A. *Test Case Run for Hunter Prey Game*

The testing procedure was conducted three times.

Firstly on the correctly working game on web server and following results were achieved.

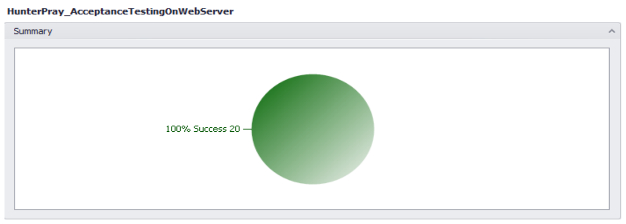


Fig 5: Test result for Goal Oriented Acceptance testing on Hunter Prey Game on Web Server

All the test cases conducted on the game Hunter Prey on web server were successful. The game worked perfectly on the web server and showed no deviation from the user specification. 100% of them were successful.

Secondly, testing was done for Hunter Prey game on user end and following results were achieved

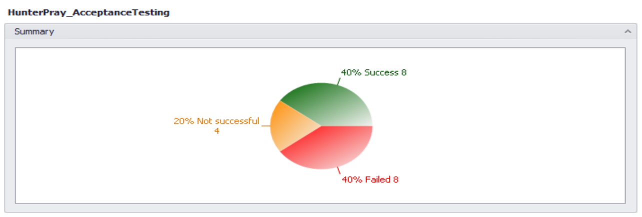


Fig 6: Test result for Goal Oriented Acceptance testing on Hunter Prey Game on User End

Not all test cases were successful for the Hunter Prey game when executed on the user's end. 40% of them were successful, 40% of them failed, and 20% of them were not successful.

Finally a formal retesting is done by user to ensure that all basic concepts are met irrespective to agent automation.

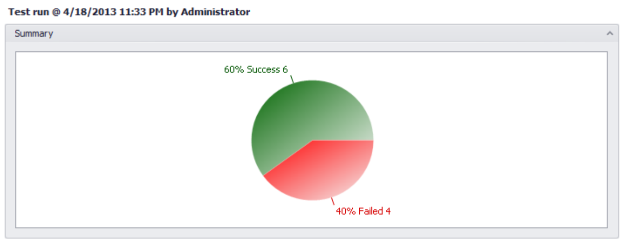


Fig 7: Test result for General Acceptance testing on Hunter Prey Game by User

Not all test cases were successful for the Hunter Prey game when executed by user with non-technical aspects. 60% of them were successful, 40% of them failed.

B. *Collective Analysis of all three testing*

Table 5: Collective Table for Test Score of each test case for Hunter Prey Game in all three scenarios

Sr. No.	**Test Cases**	**Acceptance Testing on Developer's System**	**Acceptance Testing on User's System**	**Retesting by User**
1	Pray is displayed on screen	2	2	2
2	Pray moves autonomously around the grid	2	2	2
3	Pray doesn't collide with trees on grid	2	2	N/A
4	Pray eats grass	2	1	N/A
5	Grass is displayed on screen autonomously and randomly	2	2	2
6	Grass disappear when eaten	2	1	N/A
7	Grass	2	1	N/A

Proc. of the Second Intl. Conf. on Advances in Electronics, Electrical and Computer Engineering -- EEC 2013
Copyright © Institute of Research Engineers and Doctors. All rights reserved.
ISBN: 978-981-07-6935-2

	reappears after random time			
8	Hunter is displayed on screen	2	0	0
9	Hunter moves around the grid autonomously	2	0	0
10	Hunter doesn't collide with trees on grid	2	0	N/A
11	Hunter eats pray	2	0	N/A
12	Multiple Prays are displayed on screen	2	2	2
13	Multiple Prays moves autonomously around the grid	2	2	2
14	Multiple Prays doesn't collide with trees on grid	2	2	N/A
15	Multiple prays eats grass	2	1	N/A
16	Multiple Hunters are displayed on screen	2	0	0
17	Multiple Hunters moves around the grid autonomously	2	0	0
18	Multiple Hunters doesn't collide with trees on grid	2	0	N/A
19	Multiple Hunters eats pray	2	0	N/A
20	An empty grid is displayed with trees only	2	2	2
	Total Testing Score	**40**	**20**	**12**

Table Legends for Y-Axis

Successful	2
Not Successful	1
Failure	0
Tests not performed by user	N/A

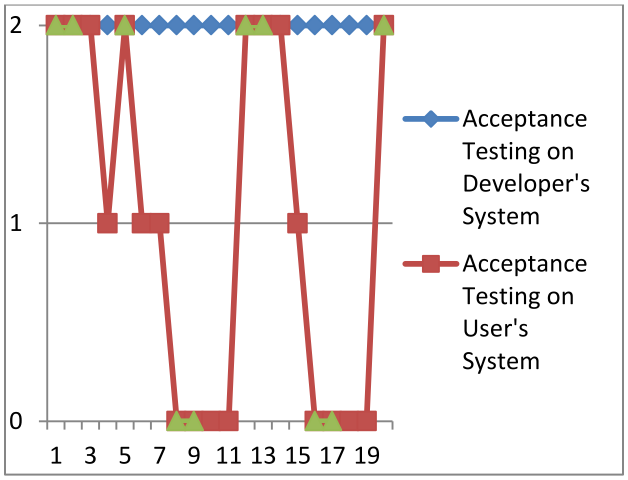


Fig 8: Collective Graph for Test Score of each test case for Hunter Prey Game in all three scenarios

Collectively, it has been noted that,

All the test cases conducted on the game Hunter Prey on web server were successful. The game worked perfectly

ISBN: 978-981-07-6935-2

on the web server and showed no deviation from the user specification. 100% of them were successful.

Not all test cases were successful for the Hunter Prey game when executed on the user's end. 40% of them were successful, 40% of them failed, and 20% of them were not successful.

Not all test cases were successful for the Hunter Prey game when executed by user with non-technical aspects. 60% of them were successful, 40% of them failed.

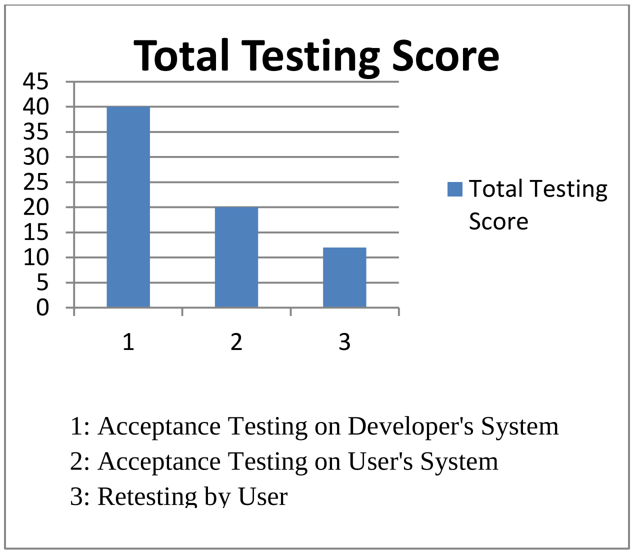


Fig 9: Graph based on total test scores

The above graph shows the total test scores acquired by all three testing scenarios. The acceptance testing on Developer's System passed all test cases and have total score of 40. Acceptance testing on User's System passed on 40% tests and 20% were not successful. So, it scored 20. Lastly, Retesting by user on non-technical non-agent based testing scored lowest 12.

IV. Conclusion and Future Scope

In this paper, two step acceptance testing and an extension for V-Model for testing has been proposed. This extended V-Model has an additional phase called "Goal Oriented Acceptance Testing". This phase lies on the demarcation of Internal and External tests and makes the Step 1 of Acceptance Testing. In this phase, a tester from developer team visits the site of customer where the AO system is installed and checks whether all agents are working according to their goals or not. The tester must ensure that all agents fulfil their basic agent properties, i.e., Pro-activeness, Social Ability, Reactivity, and Autonomous Behaviour. The phase Goal Oriented Acceptance Testing lies on the demarcation of Internal and External tests because it is performed by a tester of developer team on the user's end. Once the Goal Oriented Acceptance Testing is complete it proceeds to Step 2. The step 2 is general Acceptance testing done by user for his own satisfaction. It is done in a less technical way and in accordance to the user specification. When the AO system passes both the Acceptance Tests, it is ready to use.

In this thesis, a small AO game is tested using the proposed extended V-Model. In future, massive industry oriented AO systems can be tested using this extended model. Testing such massive AO systems will bring more enhancements to the newly proposed extension of V-Model.

V. References

[1]http://en.wikipedia.org/wiki/Agent- oriented_programming
[2]Wooldridge M. (1991), "*Agent-Based Software Engineering*", ACM
[3]Houhamdi Z. Athemena B.(2011) "Structured Sytem Test Guide Generation Process for Multi-Agent System", International Journal of Computer and Engineering
[4]http://jadex-agents.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview
[5]http://www.eclipse.org/
[6]http://www.zeta-test.com/index.html
[7]http://jadex-agents.informatik.uni-hamburg.de/xwiki/bin/view/Usages/Examples
[8]Castle C.J.E (2011) "*Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations*", Academia.edu
[9]Cu D. Nguyen, Anna Perini, Carole Bernon, Juan Pav´on, and John Thangarajah (2011) "Testing in Multi-Agent Systems", Springer
[10]Fazziki EL A, Nouzri S, Sadgal M.(2012) "An Agent-Oriented Information System: A Model Driven Approach", Internal Journal of Computer Applications
[11]Jorge J. Gomez-Sanz, Ruben Fuentes-Fern´andez, Juan Pav´on(2012), " Understanding Agent Oriented Software Engineering Methodologies", IEEE
[12]Mark F. Wood, Scott A. DeLoach(2001) "An Overview of the Multiagent Systems Engineering Methodology", First International Workshop on Agent-Oriented Software Engineering
[13]Nguyen C, Perini A, Tonella P. (2011) "A Goal-Oriented Software Testing Methodology Technical Report" ACM