Expert Systems with Applications 38 (2011) 1405–1416

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# An application perspective evaluation of multi-agent system in versatile environments

T. Vengattaraman [a], S. Abiramy [b], P. Dhavachelvan [a,*], R. Baskaran [c]

[a] *Department of Computer Science, Pondicherry University, Puducherry, India*
[b] *Department of Computer Science, Saradha Gangadharan College, Puducherry, India*
[c] *Department of Computer Science and Engineering, Anna University, Chennai, India*

A R T I C L E   I N F O

*Keywords:*
Testing and debugging
Distributed artificial intelligence
Multi-agent systems
Exception handling
Metrics-performance measures
Model validation and analysis

A B S T R A C T

Multi-agent systems (MAS) based computing is the most appropriate paradigm for the problem domain, where data, control, expertise or resources are distributed and also it is interesting to the user only if the technologies address the issues of interest to the user. The MAS has the hypothesis that the agent based computing offers better approach to manage the complex systems and process. They are large-scale systems and collaborate with one another to achieve their functions in a highly modular and flexible way. In this point of view, the work presented in this paper is an enhanced attempt to validate the MAS based on application perspective. As a test-bed, a distributed MAS for software testing is constructed such that to provide a hybrid testing environment based on variety of agents, which possibly incorporate several testing techniques. The developed framework is validated on two perspectives namely, efficiency of the application domain, i.e. software testing using MAS and efficiency of the proposed framework. The validation of the later case has been carried out on two conditions: regular working environment and exceptional working environment. The second type of validation provided the guidelines for implementing proper exception handling mechanism in the enhanced MAS, which is being developed for software testing Purpose.

## 1. Introduction

The technology of agent-based systems has generated lots of interest among the user and developer in recent years through its ability as a new paradigm for conceptualizing, designing, and implementing software systems. This characteristic is particularly attractive for creating software that operates in environments that are distributed and open, such as the internet. Currently, the great majority of agent-based systems consist of a single agent. However, as the technology matures and addresses increasingly complex applications, the need for systems that consist of multiple agents that communicate in a peer-to peer fashion is becoming apparent. Central to the design and effective operation of such MASs are a core set of issues and research questions that have been studied over the years by the distributed AI community (Platon, Sabouret, & Honiden, 2007). Multi-agent systems are considered as the next major generation of software to deal with the increasing complexity in modern applications. MAS are distributed systems of autonomous and interacting entities called as agents. They are large-scale systems and the agent research community aims at agents collaborate or compete with one another to achieve their functions in a highly modular and flexible way. A variety of applications of agent technologies can be observed in software developed from autonomous robots in manufacturing to software agents that assist users over the Internet. Multi-agent systems are therefore promising models in the future advances in software engineering and artificial intelligence fields.

Despite the rapid advances in agent technologies, their adoption in mainstream software application areas is still limited. It is generally recognized that a major reason is the lack of systematic methods to guide the development of agent-oriented systems. Agent-oriented methodologies have thus become an important and urgent area of research (Brauer, Nickles, Rpbatsos, Weiss, & Lorentzen, 2001; Bresciani, Perini, Giorgini, Giunchiglia, & Mylopoulos, 2001). In the last few years, many diverse AOSE approaches and methodologies have been proposed. They offer a range of modeling concepts, elaboration and analysis techniques, and opportunities for tool support. They vary in maturity and scope of coverage. The diversity of approaches offers rich resources for developers to draw on, but can also be a hindrance to progress if their commonalities and divergences are not readily understood (Caire, Cossentino, Negri, Poggi, & Turci, 2004). One way to advance the state of research in agent-oriented methodologies is to define a suitable example problem that can serve as a focal point for discussion and exchange of research ideas and results. Examples are indispensable for illustrating and

* Corresponding author.
*E-mail addresses:* vengat.mailbox@gmail.com (T. Vengattaraman), abiramy.hari@gmail.com (S. Abiramy), dhavachelvan@gmail.com (P. Dhavachelvan), baaski@cs.annauniv.edu (R. Baskaran).

explaining methodologies. They provide concrete instances that allow abstract concepts and descriptions to come alive through domain settings and scenarios.

Once a designer has made the decision to use a multi-agent system, a number of methodologies exist for building multi-agent systems (Brauer et al., 2001; Bresciani et al., 2001; Caire et al., 2001; Rana, 2000). These methodologies range from extensions of existing object-oriented methodologies to new agent-oriented techniques, which offer a new perspective for developing multi-agent systems by increasing the level of abstraction to analyze and design the system. But unfortunately, in the existing multi-agent systems, too few are described with sufficient details to use them for real world problems. The solution is to construct and use objective specific frameworks that are best suited for the given problem space. From this perspective here, an objective specific multi-agent framework for software testing is developed. The problem solvers of the multi-agent system, besides being autonomous, may also make use of heterogeneous design, which is also followed in this framework (Coelho, Kulesza, von Staa, & Lucena, 2006; Dhavachelvan & Uma, 2003, 2004, 2005; Dhavachelvan, Uma, & Venkatachalapathy, 2006; Rana, 2000).

A critical challenge in creating effective agent-based systems is making them robust in the face of potential failures. Most work to date on multi-agent systems has focused, however, on supporting external exception handling and has typically assumed relatively simple closed environments where the infrastructure is reliable and agents can be trusted to work correctly. In the complex and open environment the departures of the agent behavior from "ideal" multi-agent system behavior, can be called exceptions, results of inadequate exception handlings include the potential for poor performance (Denis & Bruno, 2007; Kaminka, 2009; Klein, Rodriguez-Aguilar, & Dellarocas, 2003; Platon et al., 2007; Shah, Chao, & Godwin, 2007; Zhu, 2001). The appropriate exception handlings in the MAS framework not only affect the quality of the framework, also affects the performance of the application domain. The possible set of exceptions and exception handling mechanism are also described and assessed in this paper.

The organization of paper is as follows: Section 2 defines the proposed enhanced version of multi-agent system for software testing as an application domain as proposed in Dhavachelvan et al. (2006). Section 3 explains the experimentation methodology to validate the proposed system in two perspectives including exceptional run time environment. Section 4 presents the quantitative statistical analysis over the results obtained in the experiments and the Section 5 gives the concluding remarks and future enhancements.

## 2. Proposed system

In this paper, a framework for multi-agent system based software testing is described. It is an enhanced version of the proposed framework as described in the research works (Dhavachelvan & Uma, 2005; Dhavachelvan et al., 2006). The framework is able to accommodate variety of distinct testing techniques and also can accommodate more number of products. From the product perspective, the proposed system can be defined in terms of product specific agents as in Eq. (1) as follows:

**Definition 1.** Let '$A_{p_i}$' be the set of agents needed for testing the product '$P_i$' and with respect to $P_i$, '$A_{p_i}$' can be defined as,

$$A_{P_i} = \left\{ \begin{array}{l} (D_i, a_{i1}, a_{i2}, a_{i3}, a_{i4}, \ldots, a_{iy}), \\ a_{i1} = (a_{i1}, ac_{i(11)}, ac_{i(12)}, \ldots, ac_{i(1K_{i1}-1)}), \\ a_{i2} = (a_{i2}, ac_{i(21)}, ac_{i(22)}, \ldots, ac_{i(2K_{i2}-1)}), \\ \vdots \\ a_{iy} = (a_{iy}, ac_{i(y1)}, ac_{i(y2)}, \ldots\ldots.ac_{i(yK_{iy}-1)}) \end{array} \right\} \quad (1)$$

where

- '$D_i$' is the distributor agent of the product '$P_i$' and '$H$' is the number of products to be tested simultaneously and then $0 < i \leqslant H$.
- '$a_{ij}$' is the one of the testing agents of '$A_i$' and '$ac_{i(ju)}$' is the one of the clones of '$a_{ij}$' and then $0 < j \leqslant y$, $0 < u \leqslant k_{ij} - 1$.
- '$y$' is the number of testing agents and also the number of different testing environments required for the product '$P_i$'.
- '$K_{ij} - 1$' refers to the maximum number of clones of '$a_{ij}$' and '$K_{ij}$' refers to the total number of agents available in the particular testing environment '$j$' of the product '$P_i$'.

Since this multi-agent framework provides, scalar type testing environment, at any instant, $A_{i_1} \cap A_{i_2} = \phi$, where, $0 < i_1, i_2 \leqslant H$ and $i_1 \neq i_2$, i.e. at any specific service duration, neither a single agent (distributor) nor an agent set (testing agent + clones) can be shared by more than one product simultaneously.

The state representation of the proposed MAS framework is shown in Fig. 1. The input to the distributor agent '$D_i$' is from the tester and it includes the set of the testing product, time specification for testing, defect detector estimations and the specification about the required testing techniques. This is transferred to the reactive layer of '$D_i$. The output of the reactive layer is the estimated values of products' complexity arranged in the non-increasing order.

The high priority product will be considered for service at first and the specifications will be transferred to communication layer and the other products will be given to the next distributor agent '$D_j$'. Based on the testing service specifications, the appropriate set of agents can be defined and identified through the negotiation process in the communication layer. Then the assignments will be distributed to the identified set of testing agents and their outputs can be obtained for integration. The Environmental Test Reports from the identified testing agents will be integrated in the deliberative layer and then passed to the external world. The distributor agent is responsible for all types of co-ordination activities in this system.

The initial input to the testing agent $a_{ij}$ is from the corresponding distributor agent. It includes the set of the testing product, time specification for testing, defect detector estimations and the estimated complexity of the product. This is transferred to the database and the deliberative layer of $a_{ij}$ assesses it. The output of the deliberative layer is a set of estimated values on average size of the modules of the product and the predicted values of total number of test cases to be built by $a_{ij}$, average size of the test case and average time required for generating and executing an unit test case.

Communication layer will define the mode of load distribution and it is based on the input from the distributor agent. This layer is also responsible for defining the number of clone(s) that are needed to generate. Moreover processes of clone registration, load distribution and collection of results from the clones are to be done in the communication layer. At the same time it will distribute the load to the reactive layer of same $a_{ij}$. The results from the reactive layer and Environmental Partial Test Reports (EPTRs) from the clones will be processed in the deliberative layer. Then the generated Environmental Test Report (ETR) of $a_{ij}$ will be transferred to the distributor agent $D_{i.}$

### 2.1. Distributor agent

The structure of the distributor agent in relation to its environment is shown in Fig. 2. The overall functionality of the distributor agent is composed in the three layers: Complexity assessment and Testing Products Rating and Selection are implemented in the reactive layer; integration of ETR is to be done in the deliberative layer;