COMPLEXITY MEASURES FOR SOFTWARE SYSTEMS : TOWARDS MULTI-AGENT BASED SOFTWARE TESTING

#P. Dhavachelvan[1] and G.V. Uma[2]
[1]*Research Scholar, Department of Computer Science and Engineering, Anna University, Tamil Nadu, India.. E-mail: pd_chelvoume@yahoo.co.in*
[2]*Assistant Professor, Department of Computer Science and Engineering, Anna University, Tamil Nadu, India.*

ABSTRACT

Bringing together agents and other fields of software engineering might be difficult as the advantages of agent technology are still not widely recognized. Effectiveness claims of agent-oriented software engineering are based upon the strategies for addressing complex systems. Agent technologies facilitate the automated software testing by virtue of their high-level decomposition, independency and parallel activation. The informal interpretations of qualitative agent theories are not sufficient to distinguish agent-based approaches from other approaches in software testing. In this paper, we do not just described the agent-based approach in software testing, also developed an evaluation framework for agent-oriented approach in software testing and proposed a Multi-agent system for software testing. This paper therefore provides a timely summary and enhancement of agent theory in software testing, which motivates recent efforts in adapting concepts and methodologies for Agent-Oriented Software Testing (AOST) to complex systems, which has not previously done. The 'multi-modal' approach proposed here is to offer a definition for encompassing to cover the software testing phenomena, based on agents, at the preliminary level, yet sufficiently tight that it can rule out complex systems that are clearly not agent-based.

Key words: Agents in Software Testing, Evaluation Framework, AOST, Multi-Agent System for testing.

1. INTRODUCTION

Delivering high quality software for real-world applications is difficult. A wide range of software engineering paradigms have been recently devised (e.g. object-orientation [7], component ware, design patterns [11][3] and software architectures[3]) either to make the engineering process easier or to extend the complexity of applications that can feasibly be built [17]. Although each paradigm have their own influence in the software engineering field on the support of their proficiencies, due to the exceptional growth of the software industry, researchers continue to strive for more efficient and powerful techniques [7][5][17].

Agents are being identified as a next generation model for engineering complex, distributed systems [5][17][15]. Although there are an increasing number of deployed agent applications [16][10], there is no systematic analysis precisely what makes the paradigm effective [17], when to use it and what type of applications make use of it. Agent-Oriented Software Engineering (AOSE) [7][5][17][15][16] explanations lacking in details that would allow a software tester to decide easily when to shipping to agent-based software testing. Due to these claims, there has been comparatively little work on agent-based computing as a serious software engineering paradigm that can significantly enhance development in wide range of applications. These shortcomings can be rectified by recasting the essential components of agent systems into more traditional software engineering concepts [17] [10].

In this paper, we do not just described the agent-based approach in software testing, also developed an evaluation framework (based on the attributes of complexity) for agent-oriented approach in software testing and its application to develop a multi-agent system for software testing. The multi-agent system illustrated here is on the basis of few basic operational real-world testing techniques, as an attempt to describe how to practice agent-based software testing, which has not previously done. The multi-step evaluation process described in this paper, allow us to keep a trace why (i.e. the needs) of the agent-based approach is needed for software testing.

Defining and classifying a relatively new phenomenon is always a difficult task to face the objections to basic definitions, arguments that important points have been overlooked, or claims that it is really nothing new anyway [17][4][2][8][9][10]. Bringing together agents and other fields of software engineering might be difficult as the advantages of agent technology are still not widely recognized. This paper therefore provides a timely summary and enhancement of agent theory in software testing, which motivates recent efforts

0-7803-8840-2/05/$20.00 ©2005 IEEE

in adapting concepts and methodologies for Agent-Oriented Software Testing (AOST) to complex systems.

This paper is structured as follows. Section 2 describes the Evaluation Framework for agent-based approach in software testing with experimental results. In Section 3, a systematic approach for developing multi-agent testing framework is presented. Finally, in section 4, conclusion and future perspectives are presented.

2. EVALUATION FRAMEWORK

A. Evaluation Scheme

The proposed framework is described with its primary components in the fig. 1. Based on the key characteristics of complexity and the essential concepts and notions of agent-based paradigm, we developed a set of questions (Table. I) that reflects the complexity nature of the software in maximum possible aspects. We used an evaluation scheme that assign three distinct values from 0 (not applicable or null significance) to 5 (highly applicable or absolutely essential) as the answers for each question at three conditions: minimum possibility (Min), most likely-hood (Most), and maximum possibility(Max) of the particular aspect/attribute that covered in that specified question. The optimal value of the particular aspect can be calculated as the weighted average of the three values for each question computed as in the eqn.(1). The eqn.(1) gives heaviest credence to the '*most likely-hood*' estimate.

$$C_{W_i} = (C_{Min_i} + 2C_{Most_i} + C_{Max_i})/4 \quad -(1)$$

where $0 < i \quad n$, n is the number of complexity attributes considered in the evaluation scheme and its value is equal to sixteen in the proposed evaluation scheme. The interpretations offered here concentrate on necessary, rather than sufficient conditions, so they can always be extended. As said in the complexity theory (observation-2), the questions are generated and evaluated with respect to system and product(s). After determining the optimal level of complexity parameters, the next stage is to predict and verify the adequacy of the model for determining the optimum resultant value for complexity. The optimum value of resultant complexity C_r can be determined by using Multi Modal approach [21] as follows.

Let $C = \{ C_{W_1}, C_{W_2}, \ldots\ldots\ldots C_{W_n} \}$ and M_i be the modal value with ith highest frequency f_i. If $f_i = f_j$ for different values and i j, then M_i = maximum value of (f_i, f_j) and M_j = minimum value of (f_i, f_j). Let f_{p_i} - projected frequency for M_i with respect to *n* and it can be calculated as

$$f_{p_i} = \frac{n}{f_t} * f_i \qquad -(2)$$

where $f_t = \sum_{k=1}^{q} f_k$, q is the modal degree of the system.

Modal Degree of System (MDS): If the system 'S' uses the multi modal approach such that the modal values as $m_1, m_2, \ldots\ldots\ldots m_q$, then the Modal Degree of the System is said to be as 'q', i.e. MDS(q), and the corresponding resultant complexity is denoted as C_{r_q}. The value of MDS varies between 1 to the no. of distinct values in the set *C*. The resultant complexity can be calculated as the projected mean as,

$$C_{r_q} = \frac{M_1 * f_{p_1} + M_2 * f_{p_2} \ldots\ldots\ldots\ldots Mq * f_{p_q}}{n} \quad -(3)$$

The final stage is to determine the *Distinguishing Factor* as the Complexity Index. *Complexity Index (CI)* can be defined as the ratio between C_{r_q} to Equilibrium value as in the eqn. (4).

$$\text{Complexity Index }(CI) = \frac{C_{r_q}}{EV} \quad -(4)$$

Equilibrium value (EV) of the evaluation scheme is defined as the median of the scale points and it is equal to 2.5 in this scheme.

Here the CI is the distinguishing factor to define the necessity of agent-based approach in software testing. If CI > 1, then the agent-based approach is essential, and if CI = 1, it is recommended, else the agent based approach is not necessary for software testing.

B. Experimentation and Analysis

Although more number of projects are analyzed using the proposed evaluation scheme, only twelve distinct project samples with different size and their analysis reports are displayed here. The values of C_W and C_r for twelve samples are shown in the Table-II respectively. Experiments can be made to derive two important claims.

Claim-1: The point either MDS(3) or MDS(4) is the end point of the transient state and the starting point steady state of the complexity curve (fig.2, fig.3 and fig.5). i.e. Either MDS(3) or MDS(4) acts as the equilibrium state of the complexity curve.

Claim-2: Standard deviation of the set {MDS (3), MDS (4)} gives the first least value of the deviation among the sets until the same.

i.e. { MDS(3), MDS(4) } < { MDS(1), MDS(2) }, { MDS(2), MDS(3) }

Experiments and analysis shown that the value of { MDS(3), MDS(4) } lies between 0 to $3.2 * 10^{-3}$. Based on these claims, either MDS(3) or MDS(4) can be considered as the optimal value. i.e. then the resultant complexity $C_r = C_{r_3}$ or C_{r_4}. This will avoid the excessive calculations and comparisons in the evaluation scheme, particularly if it is extended. Clearly, this assumption has much to do with complexity and little to do with comprehensibility. The primary advantage of Multi-Modal approach is that the extreme complexity values of minimal parameters don't affect the modal complexity values.

Table-1 : Questions to be answered in the Evaluation Scheme

	Questions Set	C_{Min_i}	C_{Most_i}	C_{Max_i}	C_{W_i}
Q1	Is the deadline of testing the product is unrealistic?				
Q2	Is the testing team for the product lack people with appropriate skills?				
Q3	Does the product need various types of testing techniques?				
Q4	Does the system need various types testing techniques?				
Q5	Is the compartmentalizati●n is necessary for the system?				
Q6	Is it necessary to identify most error influenced (error prone) components in the product?				
Q7	Are there any possible autonomously distributed processing functions in the system?				
Q8	Is the chosen technology (some times more than one) to be changed in the system?				
●9	Is the system needs additional techniques (components)?				
Q1●	Does the system or system components qualify as the public resource center(s)?				
Q11	Does the system need high throughput?				
Q12	Does the system need multi-path inputs?				
Q13	What is the degree of strictness to apply the schedules?				
●14	Does the system or system components need to be automated?				
●15	Is the system size is so high?				
Q16	Can the system make the required effort distributed in nature?				

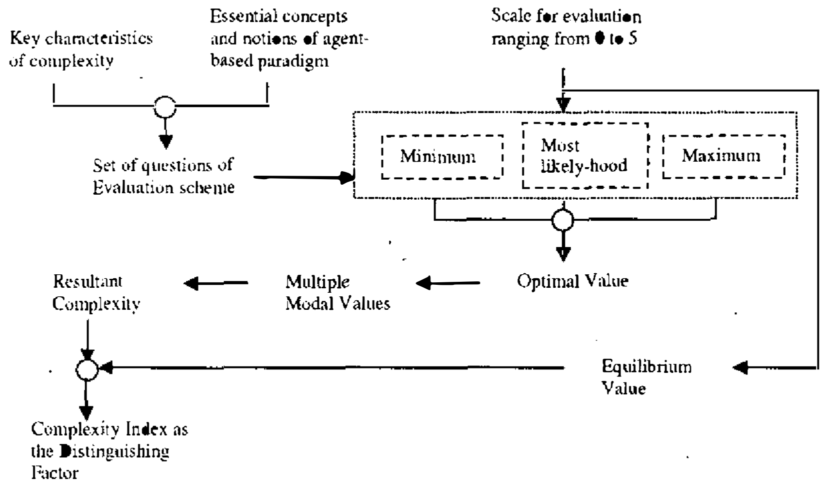

Key characteristics of complexity
Essential concepts and noti●ns ●f agent-based paradigm
Scale f●r evaluati●n ranging from ● t● 5
Set of questions of Evaluation scheme
Minimum
Most likely-hood
Maximum
Optimal Value
Multiple Modal Values
Resultant Complexity
Equilibrium Value
Complexity Index as the Distinguishing Factor

Fig. 1. Evaluation Framework

Table-II : Resultant Complexity values with different MDS

MDS of C_r	C_r values of different samples											
	PL_1	PL_2	PL_3	PL_4	PM_1	PM_2	PM_3	PM_4	PS_1	PS_2	PS_3	PS_4
C_{r_1}	4.75	4	3.25	3.75	3	3	3.25	3.5	2.0	2.25	2.0	1.75
C_{r_2}	4.53	3.9	3.63	3.32	3.33	3.3	3.57	3.31	2.1	2.12	1.75	2.1
C_{r_3}	4.23	3.68	3.8	3.53	3.38	3.42	3.18	3.45	1.92	1.99	1.82	2.13
C_{r_4}	4.18	3.7	3.73	3.61	3.4	3.42	3.27	3.42	1.9	1.9	1.81	2.1
C_{r_5}	4.12	3.49	3.69	3.56	3.31	3.39	3.23	3.42	-	1.93	-	1.98
C_{r_6}	4.07	3.44	3.65	3.48	-	3.33	3.18	3.44	-	-	-	-
C_{r_7}	3.97	-	-	3.48	-	3.29	3.16	3.38	-	-	-	-
C_{r_8}	-	-	-	-	-	-	-	3.31	-	-	-	-

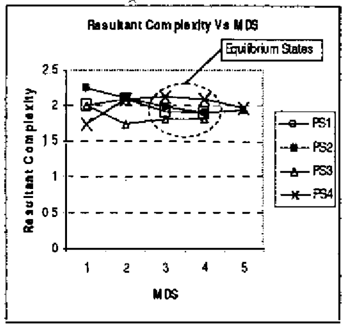


Fig. 2. Equilibrium States

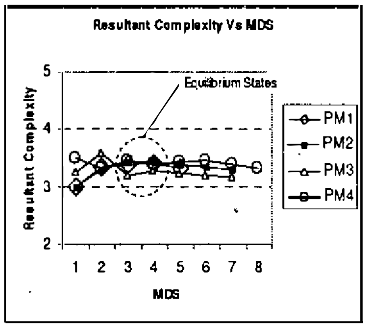


Fig. 3. Equilibrium States

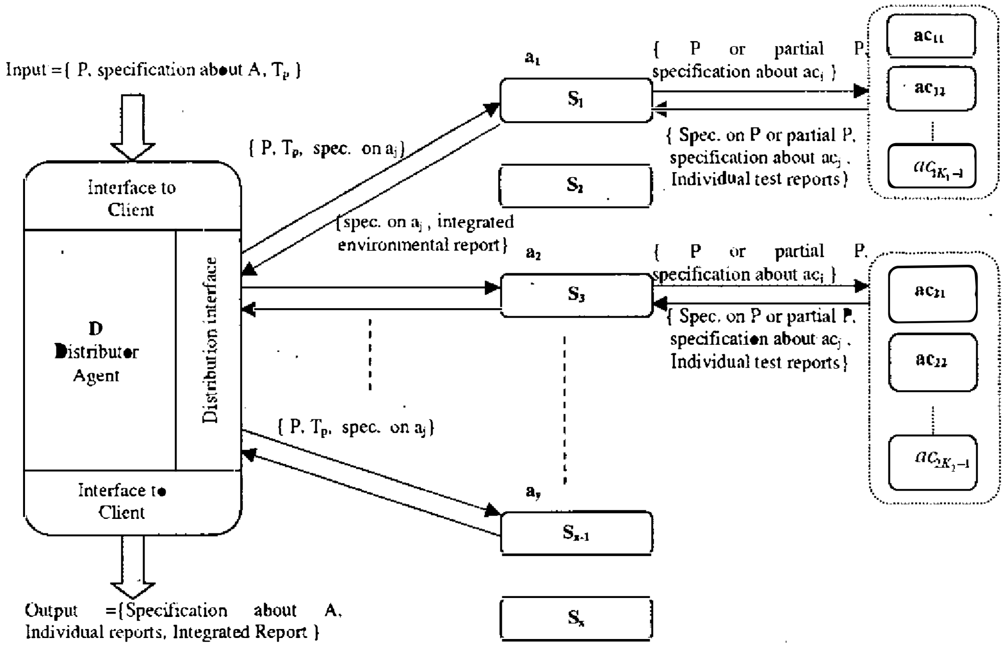
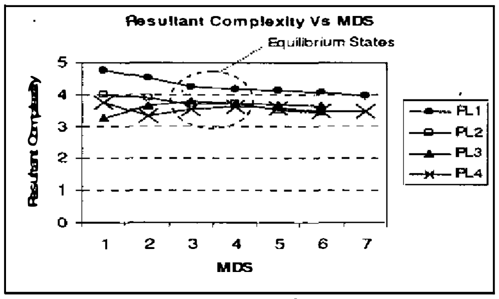

Input = { P, specification about A, T_p }
Interface to Client
D Distributor Agent
Distribution interface
Interface to Client
Output = {Specification about A, Individual reports, Integrated Report }
{ P, T_p, spec. on a_j}
{spec. on a_j, integrated environmental report}
{ P, T_p, spec. on a_j}
a_1
S_1
S_2
a_2
S_3
a_y
S_{x-1}
S_x
{ P or partial P, specification about ac_j }
{ Spec. on P or partial P, specification about ac_j, Individual test reports}
{ P or partial P, specification about ac_j }
{ Spec. on P or partial P, specification about ac_j, Individual test reports}
ac_{11}
ac_{12}
ac_{1K_1-1}
ac_{21}
ac_{22}
ac_{2K_2-1}

Fig. 4. Multi Agent System for Software Testing



3. CONSTRUCTION OF MULTI-AGENT SYSTEM

Let S be the system to be constructed for testing and it can be defined as, $S = \{D, s_1, s_2, ...s_x\}$, where *D* is the distributor agent and *x* is the number of testing agents and also the number of testing techniques available in the system. Let A be the set of agents needed for the product and it can be defined as, $A = \{a_1, a_2, ... a_y\}$, where *y* is the number of testing agents and also the number of testing techniques needed by the product P. Sometimes the input to the distributor agent may also consist of time specification. i.e. T_p is the permitted time to complete the testing processes. In the testing agents in A, there will be the predicted values about the number of test cases to be formed and executed, and the average time for single test case execution (based on the program attributes such as data variables, statements, functions, independent paths, etc).

Let C_j is the total number of test cases for a_j, $0 < j$ y and t_{g_j} is the average test case generation time and t_{e_j} is the average test case execution in a_j. The basic components of Multi-agent based testing model is shown in the fig. 4. In consists of two primary components as testing agent and distributor agent with one optional component as clones of testing agent.

A. Distributor Agent

Distributor agent D depends on the testing agents and their clones for the testing to be performed, or the testing resource to be made available. D can get the input as the combination of P, specification about A and an optional piece as T_p. Based on the specification about A, D can select the components of A from S and distribute the service based assignment to all of them. This decomposition allows one to apply alternative coordination mechanisms (such as cloning by testing agents, direct supervision of testing agents over their corresponding clones) and generating the decision parameters (such as C_j, K_j for clones generation and load scheduling) in order to achieve a literal Multi-Agent system. The message from D to a_j is a set of $\{P, T_p$, specification about $a_j\}$. The response from a_j to D, will consists of {Environmental Integrated Reports (number of test cases, number of faults detected), specification about a_j (type of testing technique)}.

The output of D consists of {specification about A (such as types of testing techniques) + environmental integrated reports (number of test cases for each technique, number of faults detected, techniques based performance) + Integrated Test Reports (total number of test cases, effort spent, total number of faults detected)}.

B. Testing Agent

The testing agents depend on the distributor agent to get the assignment with specifications. Also, they depend on their clones for the task to be performed within a precise time period that depends on the type of the task. The testing agents have independent options to be operated either in the Testing mode or in the Distributor mode.

Testing Mode: In this mode, the testing agents will be performed to execute the test cases and at the end, send the technique specific Integrated Test Report to D. In this mode, no need to generate the clones and have the control over them. This is explained in the step-3.2. of the cloning algorithm. In this mode, the total time to be spent in a_j can be calculated as,

$$T_j = C_j * (t_{g_j} + t_{e_j}) \quad - \quad (5)$$

Distributor Mode: The clones of any agent must be controlled by their respective parent agents. i.e. all agents (except clones) must be capable enough to act as a distributor and as a load scheduler for their respective clones. The message from a_j to its clones might be a set of {P or part of P + specification about ac_j}. After the completion of assigned task in the clones, the response transmission to the parent agent will consist of {Reference to Partial P + specification about ac_j + Individual test Results}. In this mode, the testing agent a_j can be defined as, $a_j = \{ a_j, ac_{j1}, ac_{j2}, ... ac_{jK_j-1} \}$. Here K_j indicates the total number of agents in the specific testing environment. K_j -1 denotes the number of cloning agents for testing purpose only and the remaining one a_j (testing or parent agent) will act as a Environmental Distributor (distributor only for specific testing technique) not as D. Then the time to be spent for testing can be calculated by using the eqn. (6) as follows:

$$T_j = (C_j / (K_j - 1)) * (t_{g_j} + t_{e_j}) \quad - \quad (6)$$

where K_j is the total number of agents in the environment j.

4. CONCLUSION

The multi-agent system presented here is systematic and it does illustrate its effectiveness in selecting the appropriate assignment based on requirements. This methodology rests on the idea of building a conceptual model that is incrementally refined and it can be extended from other existing models of other fields of software engineering. The arguments and results support that the agent models fit better for testing the complex software systems. The described framework has much to do with complexity and little to do with comprehensibility. The interpretations offered here concentrate on necessary, rather than sufficient conditions, so that they can be extended. Other related work includes developing distributed algorithms for reorganizing when goals are not being met by the agents in the systems.

REFERENCES

1. Anna Perini, Paolo Brescian, P. Giorgini, F.Giunchigila, and J.Mylopoulas (2001). "Towards and Agent-Oriented approach to Software Engineering," In Proceedings of Woa: Dagli Oggetti Agli Agenti, Modena, Italy, September-2001.
2. Barbara A. Kitchenham, Shari Lawrence Pfleeger, David C. Hoaglin, and Jarrett Rosemberg (2002). "Preliminary Guidelines for Empirical Research in Software Engineering", IEEE Transactions on Software Engineering, vol.28, No. 8, August-2002.
3. G. Booch (1994), "Object-Oriented Analysis and Design with Applications", Addison-Wesley, 1994.
4. Carolyn B. Seaman (1999), " Qualitative Methods in Empirical Studies of Software Engineering", IEEE Transactions on Software Engineering, vol. 25, No.4, August-1999.
5. Charles Petrie (2001). "Agent-Based Software Engineering", Lecture Notes in Artificial Intelligence, vol. 1957, Springer-Verlag, 2001, pp. 58-76.
6. Chavez A., Mouka A., and Maes P. (1997), "Challenger: A Multi-agent System for Distributed Resource allocation", *Proceedings of the First International Conference on Autonomous Agents'97*, Marina Del Ray, California, 1997.
7. P. CianCarini and M. Wooldridge (ed) (2001), "Agent-Oriented Software Engineering", vol. 1957, LNCS, Springer-Verlag, 2001.
8. Dag I.K. Sjoberg, Bente Anda, Erik Arisholm, Tore Dyba, Magne Jørgensen, Amela Karahasanovic, Espen F. Koren and Marek Vokac (2002),"Conducting Realistice Experiments in Software Engineeing", International Symposium on Empirical Software Engineering (ISESE'02), IEEE Computer Society, 2002.
9. David Avison, Francis Lau, Michael Myers, and Peter Axel Nielsen (1999), "Action Research", Communications of the ACM, vol.42, No.1, January-1999.
10. Dhavachelvan.P. and G.V.Uma (2004), " Multi-agent based Integrated Framework for Intra-class testing of Object-Oriented Software", Accepted for publication in International Journal on Applied Soft Computing, Elsevier, 2004.
11. E. Gamma, R. Helm, R. Johnson, and J. Vlissides (1995), "Design Patterns", Addison-Wesley, 1995.
12. Henry. S. and Goff. R.(1988), "Complexity Measurement of a graphical programming language", Software Practice and Experience, 19(11), pp. 1065-1088, 1988.
13. Hetzel, William C. (1998), The Complete Guide to Software Testing, 2nd ed. Publication info: Wellesley, Mass: QED Information Sciences, 1998. ISBN:0894352423.
14. Hong Zhu, Patrick A.V. Hall and John H.R. May (1997), "*Software Unit Test Coverage and Adequacy*", ACM Computing Surveys, Vol.29, No. 4, December-1997, pp. 367-427.
15. N.R. Jennings, M. Wooldridge(2000), "Agent-Oriented Software Engineering", in: J. Bradshaw (Ed), Handbook of Agent Technology, AAAI/MIT Press, 2000.
16. N.R. Jennings, M. Wooldridge (Eds) (1998), "Agent Technology: Foundations, Applications and Markers", Springer, Berlin, 1998.
17. Nicholas R. Jennings (2000), "On Agent-Based Software Engineering", International Journal on Artificial Intelligence, 117 (2000), Elsevier, pp. 277-296.
18. Joe W. Duran, Simeon C. Ntafos (1984), "An Evaluation of Random Testing", *IEEE Transactions on Software Engineering*, vol. SE-10, No. 4, July-1984, pp. 438-443.
19. Jones.B.F., Sthamer.H.H., and Eyres.D.E. (1996), "Automatic Structural Testing Using Genetic Algorithms", *Software Engineering Journal-11*, 5(1996), pp. 299-306.