Enhancing The Efficiency Of Regression Testing Through Intelligent Agents

Ms. T.M.S.Ummu Salima
M.E. Student
Thiagarajar College of Engineering
Madurai

Ms. A.Askarunisha
SGLCSE

Dr. N.Ramaraj
Principal
GKM EngineeringCollege
Chennai

Abstract

Software testing is indispensable for all software development. As all mature engineering disciplines need to have systematic testing methodologies, software testing is a very important subject of software engineering. In software development practice, testing accounts for as much as 50% of total development efforts. Testing can be manual, automated, or a combination of both.Manual testing is the process of executing the application and manually interacting with the application, specifying inputs and observing outputs. Manually testing the software is inefficient and costly. It is imperative to reduce the cost and improve the effectiveness of software testing by automating the testing process, which contains many testing related activities using various techniques and methods. In order to automate the process, we have to have some ways to generate oracles from the specification, and generate test cases to test the target software against the oracles to decide their correctness. Today we still don't have a full-scale system that has achieved this goal. In general, significant amount of human intervention is still needed in testing. The degree of automation remains at the automated test script level. 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 to complex system which has not previously done. Agent technologies facilitate the automated software testing by virtue of their high level decomposition, independency and parallel activation[4]. Usage of agent based regression testing reduces the complexity involved in prioritizing the testcases.With the ability of agents to act autonomously, monitoring code changes and generating test cases for the changed version of the code can be done dynamically. Agent-Oriented Software testing (AOST) is a nascent but active field of research . A comprehensive methodology that plays an essential role in Software testing must be robust but easy-to use. Moreover, it should provide a roadmap to guide engineers in creating Agent-based systems. The agent based regression testing(ABRT) proposed here is to offer a definition for encompassing to cover the regression testing phenomena based on agents, yet sufficiently tight that it can rule out complex systems that are clearly not agent based.

1.Introduction

The overall goal of this paper is to increase the efficiency of regression testing. Regression testing is the selective retesting of a software system that has been modified to ensure that any bugs have been fixed and that no other previously working functions have failed as a result of the reparations and that newly added features have not created problems with previous versions of the software. Also referred to as verification testing, regression testing is initiated after a programmer has attempted to fix a recognized problem or has added source code to a program that may have inadvertently introduced errors. It is a quality control measure to ensure

0-7695-3050-8/07 $25.00 © 2007 IEEE
DOI 10.1109/ICCIMA.2007.294

that the newly modified code still complies with its specified requirements and that unmodified code has not been affected by the maintenance activity.

Moreover, it is nearly impossible for users to manually command/control a software testing process that is even mildly complex [2]. Clearly, the need for sophisticated, automatic intelligence must be brought into the development life cycle. An "Agent", is a programmed system that can handle tasks autonomously with intelligence under some prescribed rules. To benefit from autonomous control and reduced running costs, system functions are performed automatically. Agent-oriented considerations are being steadily accepted into the various software design paradigms. Agents may work alone, but most commonly, they cooperate toward achieving some application goal(s).

Agents can be delegated to perform simple tasks or to provide autonomous services to assist a user without being "commanded." Agents play roles in terms of different expertise and functionalities while supporting services to an application[3]. They are responsible for making intelligent decisions to execute such activities and to provide valid and useful results. Agents can work alone, but most importantly, they can cooperate with other agents. Agents are software components in the system that are viewed as many individuals living in a society working together.

2. Problem Definition

2.1 Existing System

The regression testing paradigms have been recently devised using a number of automated test tools, object orientation, component ware either to make the engineering process easier or to extend the complexity of applications that can feasibly be built[4]. Although each paradigm has their own influence in the software engineering field on the support of their proficiencies, due to the exceptional growth of software industry, researchers continue to strive for more efficient and powerful techniques.

2.2 Proposed System

Proposed agent oriented software testing can be identified as a next generation model for engineering complex, distributed system[4]. Although there are an increasing number of deployed agent applications, there is no systematic analysis precisely what makes the paradigm effective, when to use it and what type of applications make use of it. 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 area of applications. These shortcomings can be rectified by recasting the essential components of agent systems into more traditional software engineering concepts.

2.3 Intelligent Agents

The general definition of an intelligent agent is: "a computer system that is situated in some environment and that is capable of flexible autonomous action in this environment in order to meet its design objectives".

Intelligent agents possess certain key characteristics:

- They perform tasks on behalf of the user
- Ability to communicate with other agents
- Can operate without direct user intervention

• Monitor environment and can directly affect surroundings

2.4 Agent communication languages

JADE (Java Agent Development Framework) is a software framework fully implemented in Java. It allows reducing the time-to-market for developing distributed multi-agent applications by providing a set of ready and easy-to-use functionalities that comply with the standard FIPA specifications and a set of tools that supports the debugging and monitoring phases. FIPA is the international standardization body for software agent technologies. JADE has been widely used over the last years by many organizations (both industrial and academic) in a variety of contexts: research projects, industrial prototyping, tutorials, and as a teaching support for agent-related courses in many Universities all over the world. JADE is extremely versatile and its features and footprint can also fit the constraints of environments with limited resources.

3. General approach and methodology

The process has been broken down into five steps :

1. Finding a suitable software package for the design
2. Determining the types of jobs that will be done by agents
3. Determining the number of agents required and designing agent algorithms
4. Implementing user interface, agent design
5. Individual agent testing.

4. Design procedures

The procedures involved in the designing of agents are as follows. Table 1 lists the various agents and its functionalities.

Table 1.List of agents and its description

Module Name	Description
Maintenance Agent	Execute test cases in priority order to achieve 100% maintenance of the code
Monitor agent	Monitors the code base for changes
Prioritization agent	Prioritize the test cases based on code coverage metric called APFD
Generating agent	To automate the Test case generation for the changed code and its affected parts.
Impact agent	Determine the impact of any software changes

5. Implementation of an Agent using JADE

```
import jade.core.Agent;
import jade.core.behaviours.*;
import jade.lang.acl.ACLMessage;
……..
public class monitor extends Agent {
private monitorGui myGui;
protected void setup() {
myGui = new monitorGui(this);
myGui.show();
DFAgentDescription dfd = new DFAgentDescription();
dfd.setName(getAID());
ServiceDescription sd = new ServiceDescription();
sd.setType("monitoring");
sd.setName("JADE-monitoring");
dfd.addServices(sd); }
public void updateCatalogue() {
addBehaviour(new OneShotBehaviour() {
public void action() {
for(i=0; i<myGui.os.getModel().getSize();i++)  {……..
String str=(String)myGui.os.getModel().getElementAt(i);
String str1=(String)myGui.os1.getModel().getElementAt(i);
try {
FileReader fr     = new FileReader(str);
FileReader fr1     = new FileReader(str1);
BufferedReader br = new BufferedReader(fr);
BufferedReader br1 = new BufferedReader(fr1);
while (((record = br.readLine()) != null)   &&   ((record1 = br1.readLine())
!= null)) {
recCount++;recCount1++;
if(record.compareTo(record1)!=0){
 r=str1;}}
System.out.println(r+ "has some modification");
} //try…
public void updateCatalogue1() {
 addBehaviour(new OneShotBehaviour() {
 public void action() {
for(i=0; i<myGui.os.getModel().getSize();i++)
{
String str=(String)myGui.os.getModel().getElementAt(i);
String str1=(String)myGui.os1.getModel().getElementAt(i);
try {……
while (((record = br.readLine()) != null)   &&   ((record1 = br1.readLine())
!= null)) {
recCount++;recCount1++;
if(record.compareTo(record1)!=0)
System.out.println(recCount1+ "th line has changed in" +str1);
}// while
```

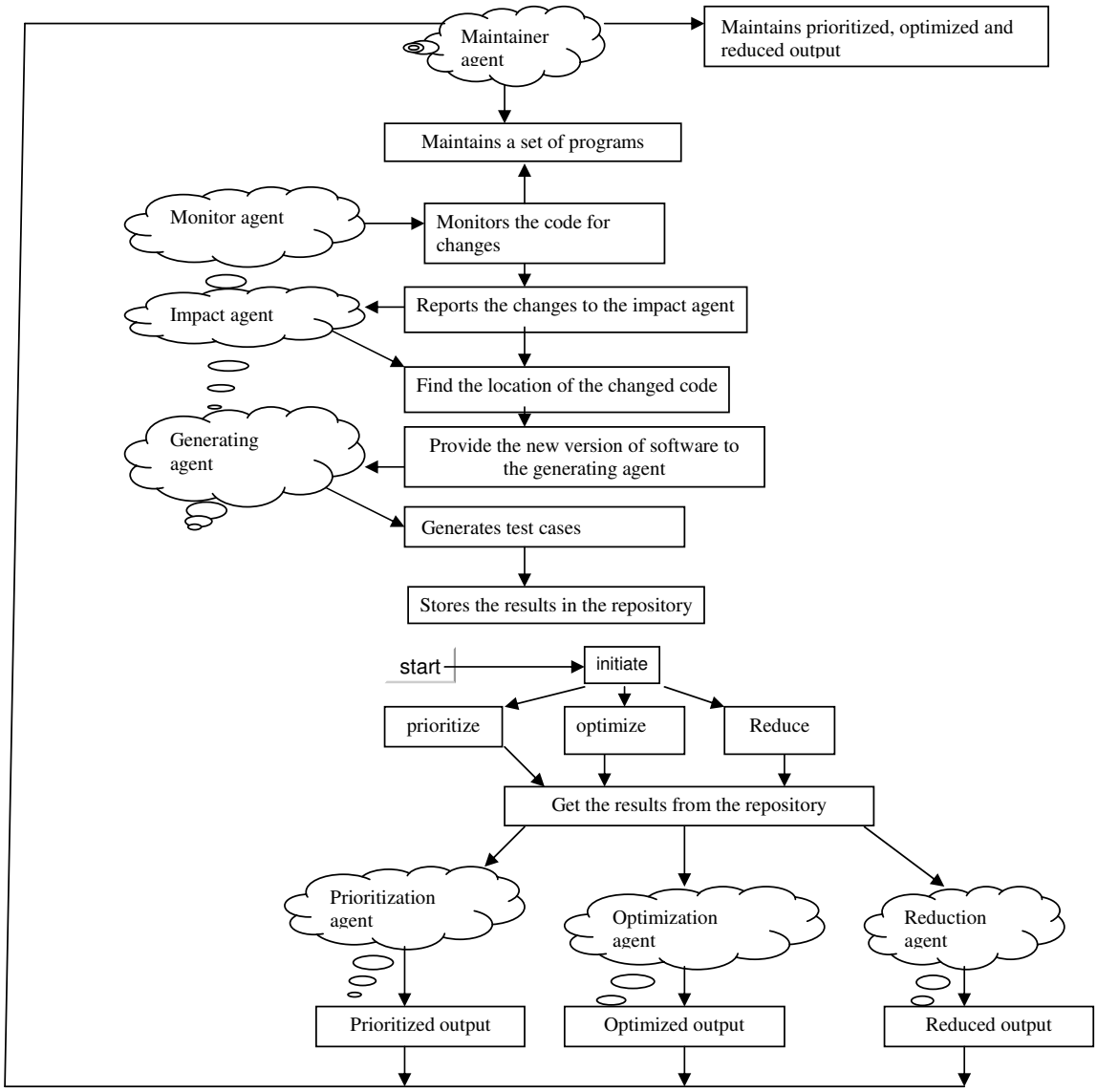

Fig.1 Flowchart for Agent Based Regression Testing

6. Conclusion

Software Agent technology has drawn much attention as the preferred architectural technique for the design of many distributed software systems. Agent-based systems are often featured with intelligence, autonomy, and reasoning. Such attributes are quickly becoming alluring to both legacy and new systems. Agents are building blocks in these software systems, while combinations of attributes are composed to form the software entities. The more complex an Agent-based system is, the more sophisticated the methodology to design such systems must be. At present there are no consensus standards on how to create Agents or model them in the development process. A study of proposals for creating Agent-based systems is under way to gain insights on what attributes are useful in leading to better design methodologies.

In this paper, we propose an agent based technique for automating the regression testing. To make a test case more adaptable to the change of application, we define an agent based technique to generate test cases which facilitates test case creation, execution and repair. A prototype multi-agent system is under development. Evaluations for using proposed technique in real world applications will be carried out.

References

[1] Amit Sharma and Miriam A. M. Capretz , "Application Maintenance Using Software Agents", IEEE , Florence, Italy, 2001, pp. 55-64

[2] Zunliang Yin, Chunyan Miao, Yuan Miao and Zhiqi Shen," Actionable Knowledge Model for GUI Regression Testing",IEEE, Melbourne, Australia,2005,pp.165 - 168

[3] Chia-En Lin, Krishna M. Kavi, Frederick T. Sheldon, Kris M. Daley and Robert K. Abercrombie," A Methodology to Evaluate Agent Oriented Software Engineering Techniques",IEEE, Waikoloa, HI,2007,pp 60

[4] P. Dhavachelvan ,G.V.Uma ,"Complexity measures for software systems: Towards multi-agent based software testing" ,IEEE, pg 359- 364

[5]. David Kung," An Agent-Based Framework for TestingWeb Applications", IEEE Proceedings of the 28th Annual International Computer Software and Applications Conference 2004

[6]. Mauro Pezz`e, Michal Young,"Testing Object Oriented Software", IEEE Proceedings of the 26th International Conference on Software Engineering,2004

[7] Chia-En Lin, Krishna M. Kavi, Frederick T. Sheldon, Kris M. Daley and Robert K. Abercrombie," A Methodology to Evaluate Agent Oriented Software Engineering Techniques", Proceedings of the 40th Hawaii International Conference on System Sciences - 2007