ASERTN: A Dynamic Testing Tool For Java Based Software Agent

Suhelah Mohammed Sandokji, Mai Fadel, Fathy Essa

King AbdulAziz University, Faculty of Computing and Information Technology.

e-mail: ssandok99@gmail.com . mfadel3@yahoo.com . fathy55@yahoo.com

ABSTRACT

Testing is an essential activity in software engineering. It is a process of observing the execution of a software system to validate whether it behaves as intended and identifies errors that sometimes can result in large financial losses or bodily harm.

Agents have been recognized as a promising technology to build the next generation of mobility services. They are increasingly used in different application domains, where autonomy, proactivity and cooperation are required. Correspondingly, the demands on the quality of the delivered agents are growing. However, testing remains a challenging activity, in order to ensure a satisfactory level of quality.

This research investigates the applicability of temporal logic-based assertion language as a means for detecting run-time errors of Java-developed software agents. Part of this research is enhancing an existing assertion language, and developing a tool, called ASERTN, as proof of concept. This paper describes the syntax and semantic of the enhanced version of the assertion language and introduces the architecture and implementation of the ASERTN tool.

Keywords: *Agent, dynamic error, assertion language, assertion statements, temporal logic, dynamic testing.*

i. Introduction:

Assertions are a common feature of most modern programming languages. They test the program state at specific locations in the source code. An assertion consists of a Boolean expression that should be true if execution is to proceed. If the expression is not true at runtime, then the system will throw an error. Thus, an assertion verify the programmers' assumption about the behavior of the program, increasing confidence of the probability of the program of being error-free or revealing the existence of errors along with a description [9].

Assertions fall short regarding checking the validity of concurrency. This gap is addressed by combining the use of Temporal logic (TL), which is a formalism used to describe how a program state will change over time [8]. Modeling the future number of states as a linear sequence of events is called linear Temporal Logic (LTL), where as modeling these states as a tree where each node branches out several possibilities is called Branching Tree Logic (BTL). LTL has been used in dynamic testing of concurrent programs, an example can be found in [5] in the AIDA dynamic analyzer. AIDA is designed to test, debug, and specify programs written in the Ada language. It conducts the instrumentation of programs. It also collects, organizes, and reports the results of running the instrumented program. Our approach is an extension of the work done in [5]. We have defined an enhanced version of AIDA's assertion language, in order to be applicable for testing software agent.

Next, we present a brief review of previous work. In Section iii, we describe the syntax of the assertion language that has been developed along with the defined operators. Then, we give an example of the use of one of the language