Delayed Failures in Software Using High Volume Automated Testing

Jeremy Gardiner
Cranfield University
Defence College of Management and Technology
Department of Information Systems
jgardiner@woomerang.com

Abstract

The research described studies delayed failures in software using high volume automated testing (HVAT) and investigates the effectiveness of different HVAT techniques; such techniques include genetic algorithms, model-based testing, penetration testing, robustness testing, and random (stochastic) testing. A delayed failure is a failure that occurs some time after the conditions that lead to the failure are applied. There appear to be no studies of delayed failures of software in the literature and no comparative studies of the effectiveness of different HVAT techniques; therefore research in this area can make an important contribution. Delayed failures in software are unlikely to be revealed by conventional testing techniques; a HVAT technique that systematically reveals delayed failures could lead to improved reliability of software and reduced costs. Experimental work is in progress using the MySQL database server as the software under test.

Keywords: automated testing, database testing, delayed failures, high volume, HVAT, software testing, testing techniques.

1. Introduction

Most software testing teams today are likely to use automated test execution tools, however these are less effective than they might be because test cases are still manually generated. The high volume automated testing (HVAT) approach combines automated test execution with automated generation of test cases, increasing the number of test cases produced in a given time by several orders of magnitude. The objectives of the HVAT approach are to increase the number of software faults found and to reduce the cost of finding faults.

The automated generation of test cases requires both automatic generation of inputs to the software under test and automatic generation of the expected responses. Since software is often sequential in nature, that is, a response depends not only on the input but also on the current software state, predicting the expected responses of the software under test requires a state model of the software. Creating and maintaining the state model can be a time consuming and costly undertaking. An alternative approach is to use software behaviour that is not state-dependent as the basis for generating test cases.

One approach based on state-independent behaviour is to use HVAT with randomly generated valid input values; this is often known as stochastic testing. The software under test should not crash or hang regardless of the sequence of valid input values executed. Another approach based on state-independent behaviour is to use HVAT with randomly generated invalid input values; this is often known as penetration or robustness testing. The software under test should reject all invalid input values and should not crash or hang.

2. Research problem

The research described studies a type of software failure that is not revealed by either stochastic or penetration testing techniques on their own, namely delayed failure. A delayed failure is a failure that occurs some time after the conditions that lead to the failure are applied; in contrast, an instantaneous failure occurs immediately after the conditions that lead to the failure are applied.

Delayed failures have been studied in the context of materials science, but not apparently in the context of software engineering, although Kaner, Bach & Pettichord [1] do mention what they call delayed-fuse bugs: "wild pointers, memory leaks or stack corruption that eventually leads to stack overflow".

The following HVAT technique for revealing delayed failures is proposed: a sequence of invalid
input values is first applied to the software under test until an invalid value is accepted. If the software under test does not immediately crash or hang, then a sequence of valid input values is executed. If the software under test subsequently crashes or hangs then this could be a delayed failure. The research problem is to show that a delayed failure can result from the software accepting an invalid input value. The effectiveness of different HVAT techniques in revealing both instantaneous and delayed failures will also be investigated.

As there appear to be no studies of delayed failures of software in the literature and no comparative studies of the effectiveness of different HVAT techniques, research in this area can make an important contribution.

3. Importance of the research

The trend towards increasing software complexity, more rapid software production and lower costs has not been matched by similar developments in software testing techniques; HVAT is a promising approach for overcoming this deficiency, but more research into different HVAT techniques is needed.

Delayed failures in software are unlikely to be revealed by conventional testing techniques. A HVAT technique that systematically reveals delayed failures could lead to improved reliability of software and reduced costs.

4. Related work

The term High Volume Automated Testing (HVAT) was introduced by Kaner, Bond & McGee [2] to cover a number of related software testing techniques that have usually been considered in isolation from one another. These techniques have in common the automatic generation of large numbers of test cases combined with automatic test execution and automated comparison of results; such techniques include genetic algorithms, model-based testing, penetration testing, robustness testing, and random (stochastic) testing. These HVAT techniques may be better than more conventional software testing techniques at finding certain types of fault such as delayed-fuse bugs. Kaner [3] has observed that the literature of HVAT is fragmented, with non-overlapping discussions of related techniques; this fragmentation makes a comparison of HVAT techniques difficult.

Related work in HVAT has tended to concentrate either on stochastic techniques using valid input values or on penetration techniques using invalid input values. As a result the type of delayed failures studied in this research have not been addressed, since a combination of HVAT techniques is required to reveal this type of failure.

Related work has employed HVAT techniques using invalid input values to find robustness faults in operating systems, using tools such as Fuzz [4] and Ballista [5]. Penetration testing techniques using tools such as PROTOS have been used to find security flaws in protocol-based software [6]. Other work has employed stochastic HVAT techniques using valid input values to find crashing faults in applications such as database systems using tools such as Random Generation of SQL (RAGS) [7].

For database systems, the output value (result) produced generally depends on both the input value and on the current state of the database, and this state may change in response to each test case. The result of testing thus depends on the order of execution of test cases; conventional testing tools and techniques are not usually designed to cope with this situation [8] [9]. This problem can be addressed by considering intensive (state-based) behaviour of the software under test as well as extensive (externally observable) behaviour [10].

5. Research hypothesis

A software specification implicitly defines a set of valid software states and a set of valid transitions between these states as a result of valid inputs. However, transitions to other (invalid) software states may be possible. Transitions to some of these invalid states correspond to software failures; these states could be called failed states. For example, in software that must execute continuously, a terminal state (a state from which no further transitions to another state are possible) is a failed state. An instantaneous failure may be interpreted as a transition from a valid state to a failed state and a delayed failure may be interpreted as a transition from a valid state via a sequence of invalid states to a failed state, as shown in Figure 1 (the figure shows just one invalid state, but the sequence might be of any length).

The research hypothesis is that the software under test enters an invalid state as a result of accepting an invalid input value, although the software continues to function apparently normally for a period of time while in the invalid state, and that this leads to a delayed failure when the software later executes a valid input value.
Figure 1. State transitions for instantaneous and delayed failures.

6. Proposed solution

Experimental work is in progress designed to test the research hypothesis and to perform a comparative study of the effectiveness of different HVAT techniques, using the MySQL database server as the system under test. MySQL has been chosen because it is an open-source software product with freely available documentation of known faults, so that failure data can be published and compared to known fault reports. Small-scale pilot experiments have already demonstrated the feasibility and performance of a PC-based test harness for MySQL. Techniques for test input generation that are to be investigated include random generation of invalid input values, grammar-based techniques, random sequencing of valid input values, graph theory techniques for sequencing valid input values, and the use of genetic algorithms to generate invalid input values that are accepted by the software under test.

The proportion of invalid input values that are accepted by the software under test provides a comparative measure of the effectiveness of different generating techniques for invalid input values. The number of valid input values executed, before either an instantaneous failure or a delayed failure occurs, provides a comparative measure of the effectiveness of different generation techniques for valid input values.

7. Expected contributions of the research

The expected contributions of the research are:

(1) To provide published experimental data on both instantaneous and delayed failures for the MySQL database server.

(2) To determine if this data supports the hypothesis that delayed failures occur as a result of the software entering an invalid state that results from the software accepting an invalid input value.

(3) To show if the proposed HVAT technique is effective in revealing delayed failures.

(4) To provide comparative experimental data on the relative effectiveness of different HVAT techniques in revealing both instantaneous and delayed failures.

8. Further research

The proposed HVAT technique for revealing delayed failures may allow measurements of software reliability, robustness and fault tolerance to be developed. Further research might consider other types of delayed software failure and might explore the use of HVAT techniques for characterising software components in terms of Quality of Service properties such as security and reliability; this could have applications in areas such as grid computing architectures.

9. References


[8] Chays, D, Deng, Y, Frankl, P, Dan, S, Vokolos, F, Weyuker, E; “An AGENDA for testing relational database applications”; SOFTWARE TESTING,