

Failures of GUI Tests on Different Computer Platforms

Lee White and Baowei Fei
EECS Dept., Case Western Reserve University
{LJW, BXF18}@cwru.edu

1. Introduction

There has been anecdotal evidence that failures from testing may be dependent upon the software/hardware platform used to test the target software. We have been involved in the research of testing Graphics User Interfaces (GUIs) [1,2,3], and have established that GUI failures do indeed depend upon the computer platform. The objective of this paper is to evaluate the computer platform parameters that are responsible for different failures observed for different computer platforms for the same tests and GUI software. Another factor is that despite thorough testing, failures may not be manifested in any observable way. References [2], [3] showed that without the use of memory tools, 30% and 85% of the failures detected would have been missed, respectively.

The *CIS approach* is used here for GUI testing [1,2,3]; it involves obtaining all *responsibilities*, i.e., observable effects in the environment of the GUI system produced by use of one or more GUI objects. For each responsibility, the corresponding *complete interaction sequences (CISs)* are identified; each CIS consists of the sequence and selections of GUI objects that deliver the responsibility. Each CIS is then modeled by a *finite-state model (FSM)* to generate the tests for the CIS. One approach utilizes the *design* of the CIS to model the transitions in the FSM; this leads to *design tests*, which assure that this CIS is implemented as designed. Another utilizes the *implementation* of the CIS to model the transitions in the FSM; this requires checking every selection in each GUI object in the CIS; the resulting new paths are then added to the design tests, resulting in *implementation tests*. For brevity, we will only report faults found by implementation tests, as considerably more faults are detected than by design tests. In the CIS testing method, a failure is observed by the tester, and identified as either due to a *defect*, where the GUI specification is

violated, or a *surprise*, which exhibits an undesirable behavior not addressed in the specification. The set of defects plus the set of surprises will define the set of *faults*.

2. Empirical Studies

We investigated the factors of operating system, CPU speed and memory for three different hardware platforms. PC3 described below was evaluated with both Windows 98 and 2000 operating systems. All three were DELL, X86-based, with L1, L2 caches; Windows 2000 operating system, version 5.0.2195 Build 2195.

PC1: Pentium IV 1.8 GHz, Mem: 1.8 GB, L1: 8 KB, L2: 512 KB, HardDisk: 40 GB.

PC2: Pentium III 800 MHz, Mem: 256 MB, L1: 32 KB, L2: 256 KB, HardDisk: 9.35 GB.

PC3: Pentium II 400 MHz, Mem: 256 MB, L1: 32 KB, L2: 256 KB, HardDisk: 4.90 GB.

The target GUI software used for these empirical studies was RealOne, a music playing subsystem of RealNetwork, with specifications: RealOne Player, Version 2.0, Build 6.0.11.818. Responsibilities: 102, CISs: 950, Design Tests: 595, Implementation Tests: 950.

Table 1. Total number of faults detected by implementation tests

	Surprises	Defects	Faults
Windows 98	96	35	131
Windows 2000	37	24	61

2.1 Operating System Effects on Failures Detected by GUI Testing

PC3 was first used to test the RealOne GUI system with Windows 98 Second Edition (SE) (Version

4.10.2222A); subsequently the Windows 2000 operating system was used, again with the same tests. Table 1 shows the results of this empirical study for the implementation tests applied; Win98 has many more detected faults, both defects and surprises, than that of Win2000. The 37 surprises are a subset of the 96; there are 18 defects in common between the two operating systems; so Win98 has 17 additional defects, whereas Win2000 has 6 different defects. Probably the major reason for this difference is that Win98 is designed for a single process, so that the CPU must be shared between all the demands of the software, which causes an increased number of faults. Win2000 is designed to handle multiple software resource demands. Yet recall that the Win2000 platform still had six more defects than Win98, illustrating that different faults can occur with different platforms, even those with improved capabilities.

Table 2. Total number of faults detected by implementation tests

	Surprises	Defects	Faults
PC1	31	19	50
PC2	34	19	53
PC3	37	24	61

2.2 CPU Speed Effects on Failures Detected by GUI Testing

Next consider PC1, PC2 and PC3 as platforms for testing with Win2000, and the same implementation tests were applied to the RealOne GUI system. In PC1, the 31 surprises comprise a subset of the 34 for PC2; also the 34 surprises for PC2 comprise a subset of the 37 for PC3 (Table 2). PC1 and PC2 share the same 17 defects, but each has 2 others. PC3 has 19 of the defects from PC1 with 5 others, and 18 of the defects from PC2 with 6 others. There is a trend with increased faults with CPU speed in the three platforms; yet since PC1, PC2 and PC3 also have Pentium IV, III and II processors, respectively, it is not possible to separate the effects of the different processors; more research is needed here.

2.3 Memory Effects on Failures Detected by GUI Testing

To examine the effect of memory, we selected PC3 with Win98 using 256, 192 and 128 MB to

correspond to three different platforms. Table 3 shows the effects of this memory change on the faults detected. The 131 faults for 256 MB constitute a subset of the 135 and 139 faults. All additional surprises in Table 3 are due to noise in the music. The new defect in 192 MB is different than the 3 new defects in 128 MB; clearly there is a trend for increased faults as memory is decreased, but it is very subtle.

Table 3. Number of faults detected by implementation tests

	Surprises	Defects	Faults
PC3 (256 MB)	96	35	131
PC3 (192 MB)	99	36	135
PC3 (128 MB)	101	38	139

3. Conclusions and Future Research

We have shown that the factors of operating system, CPU speed and type, and memory all affect the total number of faults detected, but in decreasing magnitude in order of factor listed. With the observability problems documented in previous studies, our confidence in these results is also in the order of the listed factors. Clearly larger and more careful empirical studies should be undertaken with different computers and software in order to establish greater generality of these results, as well as to account for the difficult problem of observability.

References

- [1] Lee White and Husain Almezen, "Generating Test Cases for GUI Responsibilities Using Complete Interaction Sequences", *Proc. of Int. Symp. on Software Reliability Engineering*, pp. 110-121, San Jose, CA, Oct. 2000.
- [2] Lee White, Husain Almezen and Nasser Alzeidi, "User-Based Testing of GUI Sequences and Their Interactions", *Proc. of Int. Symp. on Software Reliability Engineering*, pp. 54-63, Hong Kong, Nov. 2001.
- [3] Lee White, Husain Almezen and Shivakumar Sastry, "Firewall Regression Testing of GUI Sequences and their Interactions", accepted by the *Int. Conf. on Software Maintenance*, Amsterdam, The Netherlands, Sept. 2003.