Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications (MECSJ)

ISSUE (26), November (2019)

ISSN: **2616-9185**

www.mecsj.com

Software Bugs

Aminah Saad Shalhoob Aldossary

Master of Science in Cloud Computing

University Of Leicester, UK

Lecture at King Faisal University, KSA

Email : aldossary@kfu.edu.sa

Abstract

The software industry has been endeavored to create perfect software for the final consumers, thus this is a crucial challenge for software engineers to introduce effective techniques in order to increase the quality of software. Despite the fact that writing software without bugs is theoretically possible, the majority of software has bugs essentially, indeed there are roughly between 3 % to 20 % bugs every a thousand line of code (NRC,1999 cited in Libicki et al., 2015, p.42). Software bugs are incorrect results or odd behaviours resulted from errors or mistakes in program codes (Linfo, 2017). Bugs have effects on diverse stages of software performance some of these impacts are limited such as nuisance user whereas others have serious impacts which can lead to destroying the whole operating system (ibid). A good example is that an error in Microsoft Windows System can lead to disabling the computer work (ibid). It would seem that since 1950s software engineers have tried to create marbles, notions and methods in order to introduce an accurate software (Lu Luo, p6). Moreover, some researchers consider that there are some techniques which are more beneficial than others. This paper will outline some problems that are encountering programmers which make it impossible to write free- bugs software, then it will describe some forms of software testing techniques for finding and fixing bugs, finally compare techniques with each other through accuracy results and framework.

Key words: Software, Bugs, Quality, White Box, Black Box.

Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications (MECSJ)

ISSUE (26), November (2019)

ISSN: **2616-9185**

www.mecsj.com

الأخطاء البرمجية

الملخص

لقد سعت صناعة البرمجيات لإنشاء برامج مثالية للمستهلكين النهائيين، وبالتالي فإن هذا يمثل تحديًا كبيرًا لمهندسي البرمجيات لتقديم تقنيات فعالة من أجل زيادة جودة البرمجيات. على الرغم من أن كتابة البرامج دون أخطاء ممكنة من الناحية النظرية، فإن غالبية البرامج تحوي أخطاء بشكل أساسي، في الحقيقة يوجد ما يقرب من ٣٪ إلى ٢٠٪ من الأخطاء في كل ألف سطر من الرموز البرمجية. الاخطاء البرمجية هي نتائج غير صحيحة او سلوكيات غريبه او غير متوقعه وبتالي فإن هذه الأخطاء لها تأثيرات على مراحل متنوعة من أداء البرنامج، بعض هذه الآثار محدودة مثل التسبب بإزعاج المستخدم، بينما يكون لبعضها تأثيرات خطيرة يمكن أن تؤدي إلى تدمير نظام التشغيل بالكامل أوقد تؤدي إلى تعطيل عمل الحاسب الآلي بشكل كلي. ومن الأمثلة الجيدة على ذلك ان خطأ واحد في نظام التشغيل مايكروسوفت ويندوز كفيل بتعطيل عمل الحاسب. يبدو أن مهندسي البرمجيات حاولوا منذ خمسينيات القرن العشرين إنشاء مبادئ ومفاهيم وطرق لخلق برنامج من دون اخطاء أو أن يكون دقيق لحد ما علاوة على ذلك، يرى بعض الباحثين أن هناك بعض التقنيات التي هي أكثر فائدة من غيرها. ستوضح هذه الورقة بعض المشكلات التي تواجه المبرمجين والتي تجعل من المستحيل كتابة برنامج خالي تماماً من الأخطاء، ثم ستصف بعض أشكال تقنيات اختبار البرامج للعثور على الأخطاء وإصلاحها، ومقارنة التقنيات مع بعضها البعض من خلال نتائج الدقة والإطار.

الكلمات المفتاحية: البرمجيات، الأخطاء، الجودة، الصندوق الأبيض، الصندوق الأسود.

.

1. Introduction

There are a number of obstacles which face programmers such as difficulty to write software without errors especially currently with a substantial development in the software industry. Firstly, on the program code level, consumers usually demand high functions, therefore, these functions are promoting programmers to use huge and complex codes (Linfo,2017). For example, Microsoft Windows XP has around 40 million of code (ibid). It is clear that the length and difficulty of the codes can lead to difficult in avoiding bugs. Secondly, on the bugs level, according to Linfo (2017), as programs evolve as well as bugs are becoming more complicated to deal with. Furthermore, some types of bugs are difficult to solve (ibid). In the same way, Libicki et al.,(2015, p.54) confirm that it is difficult to eliminate some categories of bugs in application software, thus programmers and developers need to use some techniques in order to decline bugs. A perfect example is that Microsoft always uses some techniques for every new product to make bugs usually limited (ibid). Finally, Spinellis (2006, p.92) implies that even adherence to specific standards in order to make software without errors is going to create negative impacts on the final outputs. This would appear to be correct because negative impacts on the final outputs may lead to reducing the quality of software. The work of Spinellis (2006, p.92) shows that " As Pericles recognized, creating a bug-free artifact is a lot more difficult than locating errors in it". It seems that this view is valid because there are considerable obstacles which prohibit programmers and developers to introduce an accurate software, as a result, they always need to use many types of tests in order to decrease bugs in software.

2. Testing techniques

In fact, there are significant techniques for software testing in order to reduce the number of bugs. Broadly, software prevalent techniques could be classified into four major testing techniques: Correctness testing, Performance testing, Reliability testing and Security testing (Khan, 2010, pp. 24-26). Firstly, correctness testing tries to maintain the minimum requirement of software through observing software attitude and how can it deal with bugs (ibid). Furthermore, there are three categories of correction testing: White box, Black box and Gray box, all of these boxes have a function to ensure the requirement of software (ibid). In fact, white box focuses on the internal structures of software, thus white box tests inputs data and observes appropriate process which software chooses in order to make filters on these inputs to create more accurate outputs (Khan,2010, pp. 24-26).

In the contrary black box focuses on the analysis of one aspect of software without regarding the internal structures in order to find out the relationship between this element and others in the software environment (ibid). It would seem that black box examines inputs and outputs either if they are within acceptable range or not.

Khan (2010, p.26) observes that gray box works to integrate between white box and black box by comparing between a piece of software against its stipulations. It is clear that the gray box tests to what extent the results of white box and black box are correct. Secondly, performance testing. The aim of performance testing is to identify to what extent the objects of software match with performance criteria (Khan, 2010, p.26-27). According to Pan (1999), software is evaluated for their performance by three main criteria resource usage, throughput, stimulus-response and time, typically this test has been done by two main methods load testing or stress testing. However, Khan (2010, p.27) questions whether performance testing can rely on it completely by software engineers when they are looking for bugs. Thirdly, Reliability testing, as Khan(2010, p.28) points out, reliability testing is a significant method because it finds out the bugs and deletes them before publishing software through choosing an effective sampling to measure its accuracy, as a result, the developers' decisions depend on the results of reliability testing. It seems that reliability testing works to cancel the bugs without working to fix them.

Finally, Security testing, according to Pan (1999), bugs can be created by users through opening security holes, therefore, software engineers and developers need to apply security testing. Security testing gives a guarantee that no one can access the program and its functions only the persons who allowed to do this (ibid). It seems that when developers and programmers only access the software this will protect the program codes from any attempt to tamper, hence protecting software from any external bugs, as a result, the kinds of bugs which must be encountered by developers and programmers will decrease to just internal bugs. According to Khan(2010, p.28), the purpose of security testing is to find and fix the major vulnerabilities which can damage the software, as a result, the software can run for a long term without substantial problems. Software tests aim to find and fix the software problems that does not necessarily lead to improving the quality and reliability of the software.

For instance, in California in 1991 the telephone did not work completely, the reason was changing three lines of codes (Pan,1999). It would seem that software tests can find and fix bugs.

However, in some cases, software tests can lead to creating a new problem because the tests themselves have some weaknesses aspects.

3. compare between software testing

There are two main level to compare between software testing. To begin with, correctness testing and performance testing. In fact, correctness testing has more accurate results than performance testing because during performance testing there are some significant mistakes such as ignoring of bugs in input and wrong analysis,

although even correctness testing usually misses codes which they already have deleted (khan ,2010, pp.25-26). It would seem that correctness testing excels on performance testing on the level of the accuracy of results. On the other hand, the work of Khan (2010, P.28) implies that reliability testing is more different than security testing on the level of the framework because security testing focuses on fixing the problems, in the contrary reliability testing is limited to deleting bugs without processing them.

4. Conclusion

This paper has focused on some obstacles which face programmers and developers. It has also described four major techniques in order to reduce the number of bugs. Finally, it has compared these techniques with each other. From the practical side, it is impossible to write software bug-free, therefore, software engineers and programmers tend to use various techniques to reduce bugs and introduce an accurate software in order to have final user satisfaction. Software sector is racing against time to introduce software bug-free through using many techniques, in addition, some of these techniques are more beneficial and more accurate than others. Software industry must focus on how to improve these techniques in order to achieve accurate results. Moreover, developers and software engineers should pay their attention to the security field and try to improve it in order to prohibit external bugs entering into the software, hence reduce the types of bugs encountered by programmers.

References:

1. Khan, M., 2010. Different Forms of Software Testing Techniques for Finding Errors. International Journal of computer science, 7(3), pp.24-29.

2.Libicki, M., Ablon, L., and Webb, T., 2015. The defender's dilemma. Santa Monica : rand corporation.

3.Linfo.org. 2017. Bug definition by The Linux Information Project (LINFO). [online] Available at: http://www.linfo.org/bug.html [Accessed 23 Nov. 2017].

4.Luo, L., Software Testing Techniques. Technology Maturation and Research Strategy, pp.4-8.

5.Pan, J., 1999. Software Testing. Dependable Embedded Systems.

6.Spinellis, D., 2006. Bug busters. IEEE Software, 23(2), pp.92-93.