مجلة العلوم البحثة والتطبيقية
Journal of Pure & Applied Sciences
www.Suj.sebhau.edu.ly ISSN 2521-9200
Received 10/04/2017 Revised 15/06/2018 Published online 18/06/2018

The Correlation between software difficulty and care chargeS

Ahmed Mohamed Ibrahim Alabbasi
Faculty of Technical Sciences ,Computer Department, Sebha University, Libya
Corresponding author: aammbbsi@gmail.com

ABSTRACT As software becomes more and more difficult due to increased number of module size, process size, and dividing difficulty, software care prices are often on the increase. Consider a software such as Windows 8 operating systems with 80 million Source lines of code (SLOC) and with 3000 designers [1], there is no disbelief that such a large and complex software will require large amount of money, common and environmental factors to keep it. It has been estimated that over 70% of the total costs of software development process is spent on maintenance after the software has been distributed. This paper studies the relationship between software complexity and maintenance cost, the reasons responsible for software difficulty and why maintenance costs increase with software complexity. The results show that there is a strong correlation between software complexity and maintenance costs. That is, as lines of code increase, the software becomes more complex and more bugs may be presented, and therefore the cost of keeping software rises.

Keywords: Software, Software Maintenance, Software Evolution, Maintenance Costs.

الارتباط ما بين صعوبة البرمجيات و تكلفة العناية

احمد محمد ابراهيم العباسي
كلية العلوم التقنية–قسم الحاسوب– جامعة سبها، ليبيا
للمراسلة: aammbbsi@gmail.com

الملخص في سياق هندسة البرمجيات، يشير مصطلح صيانة البرمجيات إلى التعديلات التي تجرى على منتج برمجي بعد التسليم بهدف تصحيح العيوب أو تحسين أداء البرمجية أو أي خاصية من خصائصها. وأن هذه المرحلة هي الأطول في حياة النظام البرمجي لبقاء النظام قادراً على مواكبة التطورات والمعدات الحديثة و جزء من هذه المرحلة يكون في تصحيح الأخطاء، والجزء الآخر يكون في التطوير وإضافة تقنيات جديدة. فأن تطوير وصيانة مشاريعِ الحاسب الضخمة ذات النوعيةِ العالية عملية صعبة ومعرضة للأخطاء و مكلفة ايضا. وبالنظر الى نظام التشغيل ويندوز 8 لديه 80 مليون سطر من التعليمات البرمجية (SLOC) و 3000 مبرمج فلا يوجد شك من ان هذه البرمجيات الضخمة و المعقدة ستتطلب مبالغ كبيرة و عوامل بيئية معينة للحفاظ عليها. و قد قدر ان اكثر من 70% من التكلفة الاجمالية لتطوير العمليات البرمجية صرفت لصيانة البرمجيات التي تم توزيعها. وهذه الدراسة تشير الى وجود علاقة قوية بين هذه البرمجيات و تكلفة صيانتها. وعليه كلما ازدادت التعليمات البرمجية كلما اصبح البرنامج اكبر حجما واكثر تعقيدا و اكثر عرضة لضهور الفيروسات و بذلك تكلفة الحفاظ أو الابقاء عليها في ارتفاع.

1. Introduction:

There is no disbelief that software is becoming complex due to technological development, organizational request, need for ease of use. The first operating system that was produced is the batch operating systems in the late 1940s. These operating systems could only support a limited jobs being processed in groups. Ever since then, operating systems have continued to become more and more complex supporting networking, real-time processing, multi-processing, multi-programming, and a lot of other events. It is a well-known fact that after the first release of Windows Operating System (WOS) in the 1990s, WOS has become more and more complex, developing from Windows 3.0 to Windows 8. The lines of code (LOC) have also increased due to the fact that developers of WOS are responding to customer demands, environmental factors. As a result, the costs of maintaining the WOS have also increased due to the increase in human work needed to develop and maintain it as it grows from one version to the next. This case is not atypical to only WOS but to all other software that are in use especially those in high demand such as application software [1] [2].Maintenance plays an important role in the life cycle of a software product. It is therefore necessary to study the connection between software complexity and maintenance costs so as to know the factors responsible for software complexity and why complexity increases the costs of maintenance. Therefore, there is need to estimate the costs of maintaining software. [3] As noted in [4] [5], A system that is more complex may be harder to specify, harder to design, harder to implement, harder to verify, harder to operate, risky to change, and/or harder to predict its behavior. Complexity affects not only human understandability but also "machine understandability". Larger and complex software projects require significant management control. Also, the maintenance of large software systems requires a large number of employees.Therefore, management must find ways to reduce the costs of software maintenance by ensuring that the right people are employed to keep them to avoid more complication of the software.

2. Background of Software Maintenance

Software maintenance is defined by The Institute of Electrical and Electronics Engineers (IEEE) as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment. Software maintenance is the general process of modifying a system after it has been delivered to the organization or user requesting the software. Therefore, software maintenance and evolution are important concepts in the software life cycle because organizations are now completely dependent on their software systems and have invested a lot of money in these systems.

2.1. Software Complexity

The (IEEE) defines complexity as the degree to which a system or component has a design or implementation that is difficult to understand and verify [6] . As seen from these definitions, complexity is not necessarily measured on an abstract scale, complexity is relative to the viewer, what appears complex to one person might appear simple to the other person. Complexity in software is not entirely subjective because it can be measured and since it can be measured, it has some determined values. Therefore, the growth in complexity of a system is the growth in risk.

The complexity of software increases as the size increases. That is, as software grow in size, it becomes necessary to determine the complexity. Therefore, increased software complexity means that maintenance and enhancement of projects will take longer time to be completed, will costs more, and will result in more errors[7].

2.2. Causes of Software Complexity

Several factors can be traced to the causes of software complexity. These include:

2.2.1 Lack of documentation:

Most software projects are lacking in documentation, they are often incomplete or insufficient for maintainers to fully understand the software. Most often, the people that understand the software usually leave or retire from the organization without being replaced. Also, sometimes, they die and their knowledge dies with them. Thus software becomes increasingly complex. What this simply means is that documentation is an essential activity in software development process as it helps programmers to better understand the software during analysis and design stages of maintenance process [8].

2.2.2 Presence of dead code:

The term dead code means unnecessary, inoperative code that can be removed without affecting program's functionality. These include functions and sub-programs that are never called, properties that are never read or written, and variables, constants and enumerators that are never referenced and user-defined types that are never used. These dead codes often increase the size of software by increasing the Executable of (EXE or DLL) file size by hundreds of kilobytes thereby making it complex and therefore difficult to maintain. Thus the older and larger the system is, the more dead code in the system [9].

Dead code in a program often cause increased memory utilization, slower execution of programs, more code to read and maintain, increased effort and time in trying to comprehend the code. The effect of all these is that there will be increased in complexity and costs of maintenance. Thus there is need for programmers and software developers to remove dead code from their software as much as possible before releasing it to customers to help future maintainability.

2.2.3 Presence of bugs/faults:

Faults in a software program are often called bugs. A software bug is failure or mistake in a program that produces undesired or incorrect results. It is an error that prevents the application from functioning as it should. Software faults manifest themselves only under particular conditions. However, a single software fault can give rise to system errors or failures. This may happen until the bug has been identified and corrected.

According to a study conducted by the US Department of Commerce, in a typical software development project, 80% of the cost of software development is spent on identifying and correcting software faults [10] .

A huge time and money is often spent to verify that a system is fault free. Figure 1 shows the costs of fixing bugs in a typical software life cycle. As seen in the figure, the cost of fixing bugs in the maintenance stage appears to be much higher than all the other stages. Therefore there is need to properly develop software by having good design and ensuring that the software is well tested to remove bugs as much as possible before the software is released to the market or customers to reduce maintenance cost.

However, these days, several tools and techniques have been developed to help automatically find bugs

.

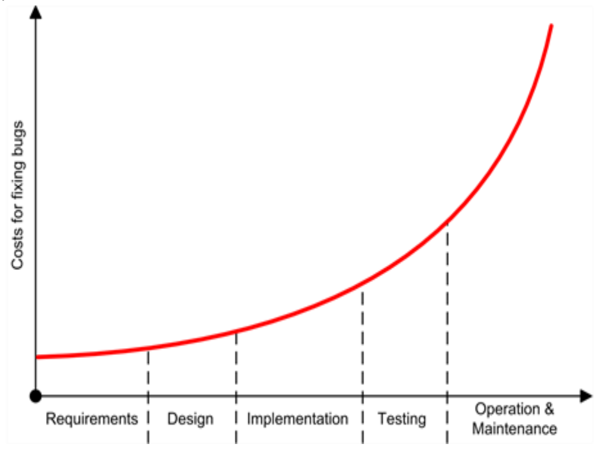

Figure 1. Costs of fixing bugs [5] .

2.2.4 User's changing requirements:

Software users often demand for other requirements when software has just been delivered to them. They often ask the developer to provide other functionalities that eventually make the software to become complex at the end and difficult to maintain.

2.2.5 Bad design:

Bad design can also lead to software complexity. Software that is not properly designed can cause a lot of wastages during development and maintenance stages[3]. During the design phase, there should be a breakdown of the design model into smaller modules which is referred to as detailed design. then the software can easily be developed and will not be too complex for human comprehension.

2.2.6 Environmental changes:

Most often, Software developed for a particular environment or organization when taken to another environment or organization might require some modifications to the software before it can be used or adapted to that environment. In trying to adapt the software to the new environment, maintenance must be carried out. However, this could lead to further complexity of the software making it difficult for future maintenance[11].

2.3. Factors Affecting Software Maintenance Costs

Software maintenance costs are significantly affected by software complexity, measured in three dimensions: Program size Modularity, and The use of branching.

2.3.1 Program size

It is a general belief that the more the size of a program increases, the more the program becomes more complex. Large systems require more maintenance effort than do smaller systems. This is because there is a greater learning curve associated with larger systems, and larger systems are more complex in terms of the variety of functions they perform. less maintenance is needed when less code is written. the length of the source code is the main determinant of total cost during maintenance as well as initial development. Also, older systems are more difficult to maintain than newer systems because with usage and frequent changes to the older software, they become less organized and less understandable with staff turnover[1].

2.3.2 Modularity

Modularity is the degree to which system's components may be separated or recombined. for large modules to be meaningful and useful in a program, they must be broken down into smaller modules called sub-modules. it is clear that there is no firm or organization that has been able to solve the problem of software complexity. even a firm with deep expertise in software development, like Microsoft, can still suffer from a complexity disaster resulting from a system's lack of modularity[9].

2.3.3 The use of branching

Branching plays a major role in the development process of large software. The use of branching in software development can have some negative effects on the code by making the program more complex. Furthermore, a build break on a branch often affects the team working on that branch and not on the entire development team. Branches are meant to provide a level of isolation for development teams to work on parts of the code base without having to worry about affecting others. Branches may introduce a false sense of safety, as changes made in different branches will eventually be merged together. increasing the chances of introducing regression failures and making it difficult to maintain the code base [8].

3. Case Study of Windows Operating Systems Software

This section provide estimated costs for development and maintenance for a large project such as Microsoft WOS certain percentage of the costs for each version of the software. As in data in Table 1.

Year Released	Product Version	Dev. Team Size	SLOC (Million)	Dev. Estimated Costs (Billion US Dollars)	Maint Estimated Costs (Billion US Dollars)
Jul-93	Windows NT 3.0	200	4-5	10	28
Aug-95	Windows '95	450	7-8	18	47
Jun-98	Windows '98	800	11-12	23	58
Feb-00	Windows 2000 Professional	1400	29+	32	72
Oct-01	Windows XP	1800	45	40	86
Apr-03	Windows Server 2003	2000	50	48	102
Nov-06	Windows Vista	2500	56	60	128
Oct-09	Windows 7	2700	67	75	161
Oct-12	Windows 8	3000	80	83	192

Table 1. Versions of Microsoft Windows operating system[1].

As seen in Table 1, as these operating systems evolve; source line of code SLOC increases ,so that the costs of development and maintenance through these versions will be on the rise. The table also shows that in each release of Windows version, the size of the development team is on the increase. This is because as more functionalities are needed by users, so there is pressure on the software development team to release new products that will meet user's demands and in the process of trying to meet deadlines, new maintainers will be employed and the more people are involved, the more bugs are introduced into the software. Thus when the software is eventually released to the user, there will be further need to maintain the software since those latent errors will start to manifest themselves with usage of the software. Therefore, having a large and complex program will eventually make the program more costly and difficult to maintain. This is graphically shown in Figure 2.

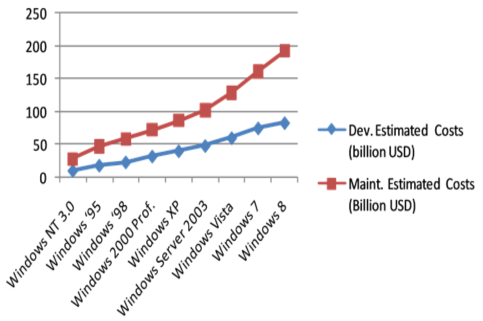

Figure 2. Windows operating system and estimated development and maintenance costs.

5. Conclusions

On conclusions, The results shows that there is a direct connection between software complexity and maintenance costs. That is, as lines of code increase, the software becomes more complex and more bugs may be introduced, and hence the cost of maintaining such software increases. The estimated costs of software maintenance are high enough to justify strong efforts on the part of software managers to monitor and control complexity.

References:

[1]- M.Glinz,(2003),"Estimating Software Maintenance", Requirements Engineering Research group, University of Zurich.

[2]- Lyu, M.R. (1996) Handbook on Software Reliability Engineering. McGraw-Hill, USA and IEEE Computer Society Press, Los Alamitos, California, USA.

[3]- Ogheneovo, E.E. (2013) Software Maintenance and Evolution: The Implication for Software Development. West Africa Journal of Industrial and Academic Research, 7, 34-42.

[4]- Banker, R.D., Datar, S.M., Kemerer, C.F. and Zweig, D. (1993) Software Complexity and Maintenance Costs. Communications of the ACM, 36, 81-94

[5]- Grubb, P. and Takang, A.A. (2003) Software Maintenance: Concepts and Practice. 2nd Edition, World Scientific Publishing Company, Singapore.

[6]- Guth, R.A. (2005) Code Red: battling Google, Microsoft Changes How It Builds Software. The Wall Journal, A1. Factiva, MIT Libraries, Cambridge.

[7]- Shihab, E., Bird, C. and Zimmermann, T. (2012) The Effects of Branching Strategies on Software Quality. Proceedings of ESEM'12, Lund, 17-22 September 2012, 301-310.

[8]- Appleton, B., Berazuk, S., Cabrera, R. and Orenstein, R. (1998) Streamed Links: Branching Patterns for Parallel Software Development, 2002.

[9]- Mall, R. (2009) Fundamentals of Software Engineering. 3rd Edition, PHI Learning Private Ltd., New Delhi, 404-411.

[10]- Dvorak, D.L., Ed. (2016) NASA Study on Flight Complexity Final Report, Systems and Software Division, Jet Propulsion Laboratory, California Institute of Technology.

[11]- R. Spiewak, K. McRitchie. (2014) "Using Software Quality Methods to Reduce Cost and Prevent Defects", 23-27 .