Southern New Hampshire University

CS-330: Assignment 1-3: Module 1 Journal

Prepared on: March 9, 2024

Prepared for: Prof. Angelo Luo

Prepared by: Alexander Ahmann

Contents

1 Introduction **1**

2 A look at Software Testing **3**

2.1 Software Testing Process 3

2.2 Discussion: Importance of Software Testing 4

2.3 Discussion: Fluidity in the SDLC 7

3 Conclusion **7**

1 Introduction

The *Software Development Life Cycle*, typically abbreviated to "SDLC," is a model by which software engineers can use to build computer programmes and other kinds of software that is mature enough to be used by the general public, and that can be used to reliably[1] solve real world problems.[2]

The SDLC process can be broken up to six (6) steps: *planning and requirement analysis*, *defining requirements*, *software architecture design*, *building and development*, **testing the software**, and *deployment of the software.* Figure 1 depicts a flowchart of the Software Development Life Cycle.

[1]My use of the term "reliably" is a bit subjective: I am thinking of software that gives consistent results, is robust to an extreme operating environment— amongst other desirable qualities

[2]Paraphrased from Tutorials Point (n.d.).

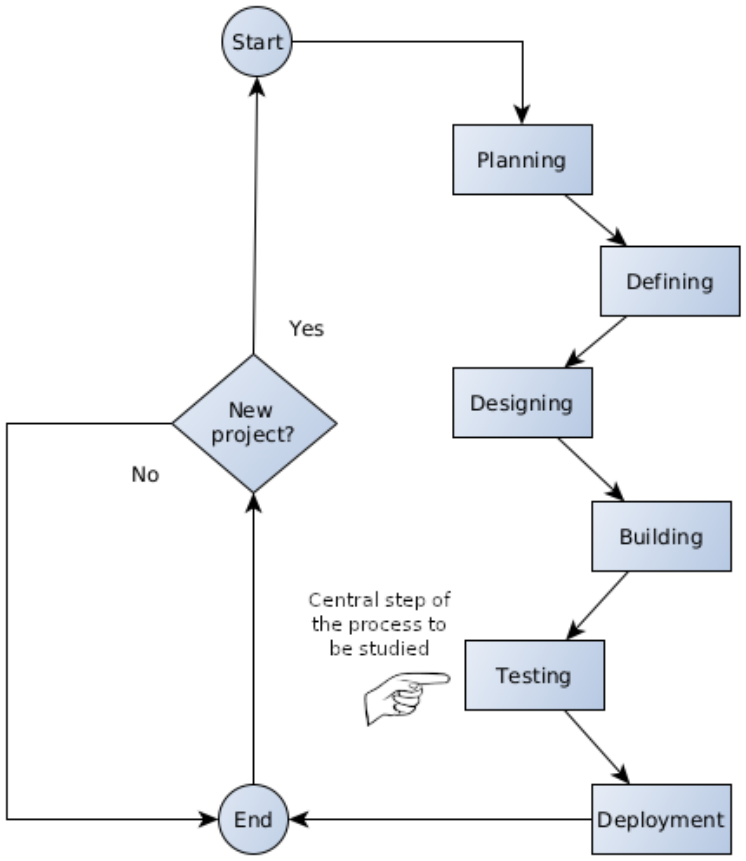

Figure 1: The *Software Development Life Cycle* (simplified); after Tutorials Point (n.d.).

The first two steps[3] involves collaborating with non-technical members of the organisation to deploy the software and determine what kind of problem that the software aims to solve, and then go about devising "blueprints" for the software.[4] The next two steps[5] involves getting more technical with the design of the software and then writing and implementing the software, typically in some programming language. The final step[6] involves releasing the finished software product to the general public — as it is ready for use

[3]Planning and requirement analysis, and defining requirements.

[4]With the exception of the "testing" step, the descriptions for these stages have been paraphrased from Tutorials Point (n.d.).

[5]Software architecture design and building and development of the software.

[6]Deployment of the software.

and consumption by the end user.[7]

2 A look at Software Testing

An organisation *ought not*[8] to deploy their software before they subject it to *software testing*— which the focal point of this paper.[9] Tutorials Point (n.d.) describes the software testing, the fifth step in the SDLC, as:

> "*usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC [... snip ...] it refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the [software requirement specification].*" — quoted verbatim from Tutorials Point (n.d.)

A baseline standard is defined and software testers are, to the best of their ability, test the software product to determine if it can solve the problem that it is intended to solve. Software developers and quality assurance testers typically subject the software product to edge cases and fault injection to determine how it would react to a hostile operating environment. Programmers will then modify their software in accordance with the results from the testing phase to meet the baseline standard.

2.1 Software Testing Process

Like the SDLC, software testing has its own process.[10] It starts with a *planning* stage, then a *monitoring and control* stage, then an *analysis* stage, then a *design* stage, then an *implementation* stage, then an *execution* stage, and finally a *completion* stage. This process is "branched out" depending on the results of each stage of the process and external influences. Figure 2 illustrates the software testing process as it "branches out."

[7] This is a bit misleading: the deployment process is dynamic and "interdependent," as it requires input from the end user to help the software developer fix issues and improve the end product.

[8] This is a normative rule, not an objective description of organisational behaviour. Technically, organisations are free to deploy their software products without testing it.

[9] I skipped the testing phase in the last paragraph because it will be elaborated on more in the rest of this paper.

[10] After Hambling et al. (2019, p. 24).

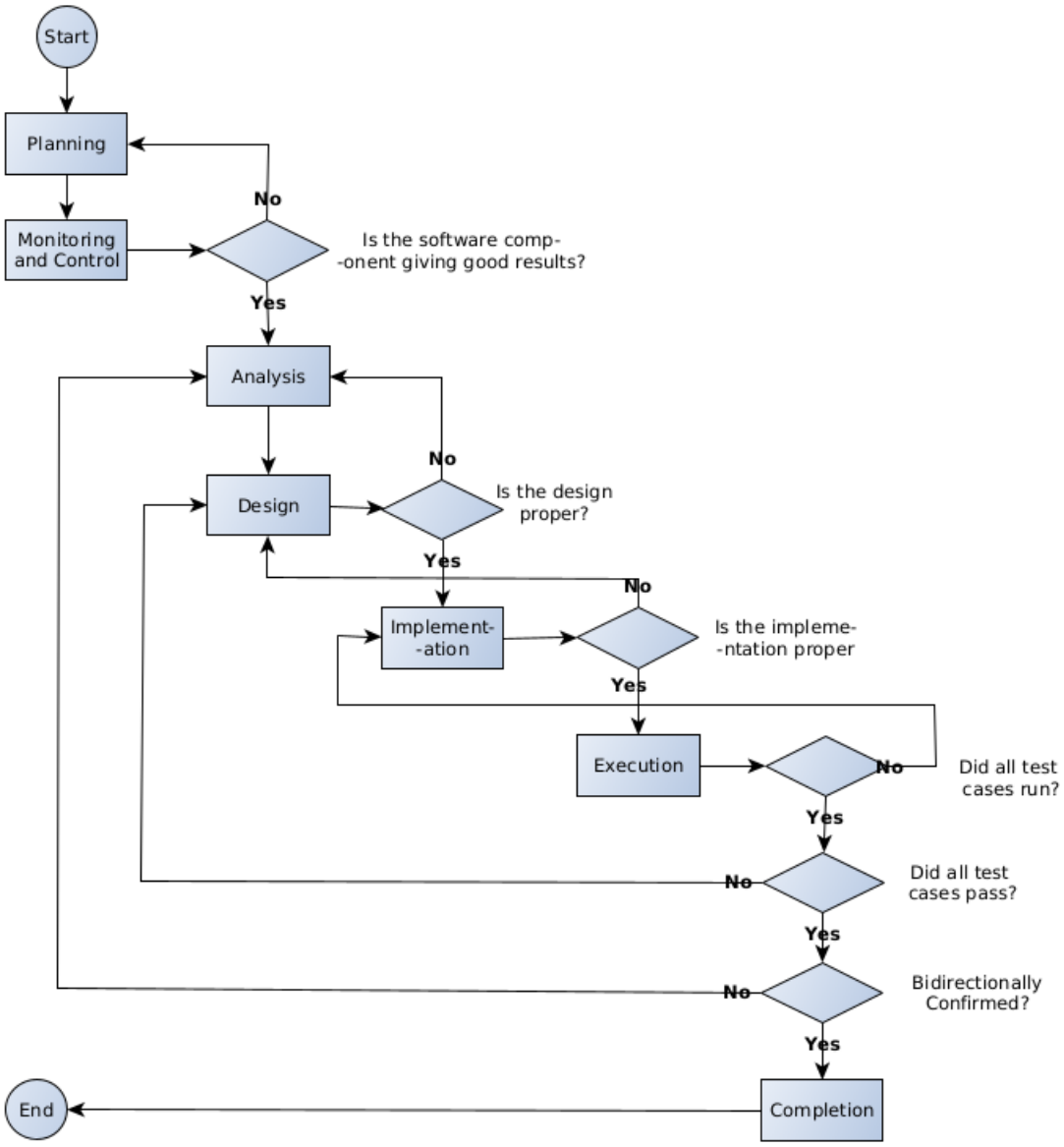

Figure 2: The *Software Testing Process*; after Hambling et al. (2019, p. 24).

2.2 Discussion: Importance of Software Testing

The software testing textbook meticulously describes each step of the process (Hambling et al., 2019, pp. 26–30), so it would be repetitive for me to do so here. Instead, I would rather discuss the importance of software testing and

how the process may change depending on how it is influenced.

Software developers or, more commonly, business people who are interested in the increase of profits and reduction of costs, may ask "*why spend time, resources and money on software testing?*" The software testing textbook (Hambling et al., 2019, pp. 10–11) lists some of the consequences for failure to engage in software testing: which include *loss of profits*, *loss of business reputation*, and even *injury and death*. Regarding how bad software can lead to material and physical harm, I present the following real life examples:

- An early example was the *Patriot Missile Failure* (Arnold, 2000). In February 25, 1991, the Patriot missile was tasked to intercept an Iraqi missile. But a software programming mistake in the Patriot missile, involving the lack of precision in floating-point numbers, caused a kinematics miscalculation— resulting in the death of 28 American soldiers and injuring about a hundred other people.

- In 2010, the *Stuxnet* computer virus infected the computer systems of to nuclear facilities. Some target systems were airgapped and isolated from the internet (Holloway, 2015). Regarding the software-side of things, the worm exploited zero-day and n-day software vulnerabilities to propagate, amongst other techniques for self-spreading.

- Around May 2015, an *Airbus A400M* plane experienced a failure and crashed. It was confirmed that a software misconfiguration caused the problem (Gallagher, 2015). This is more of an example of a software misconfiguration as opposed to a programming error, but is still a good example on how misused software can lead to damages.

- In 2017, the *WannaCry ransomware* infected hospital computer systems worldwide and put patients lives at greater risk (Collier, 2017). WannaCry infected networked computers through its worm component exploiting a software bug in a Microsoft NetBIOS service. When the worm component successfully exploited the service, it injected code into the target system that instructed it to install WannaCry.

The impact of software on the material world has become very apparent, which has led to scholars and practioners to investigate how to approach potential issues in the future. For example, self-driving cars are now a "hot topic" in the technology world.[11] Bonnefon et al. (2015) discuss a trolley

[11]The following is a "popular science" article discussing the situation (MIT review, Retrieved on Mar. 9, 2023): https://www.technologyreview.com/2015/10/22/165469/why-self-driving-cars-must-be-programmed-to-kill/

problem applied to self-driving cars. They survey the social science literature to work out the *vox populi* regarding the ethics of self-driving cars and discuss the results, along with the importance of these ethical questions. Likewise, the importance of *software testing* is important to the development of technology with good results. A lack of software testing usually implies that said software will not function as it is intended to, or may function in an unreliable and unstable manner.

In addition to the harm that untested or unreliable software would bring to society, it will also increase costs on the organisations developing and maintaining the software product. Hambling et al. (2019, p. 20-21) posit that the cost of software testing increases by orders of magnitude[12] as it gets addressed further into the software development life cycle. Figure 3 illustrates the cost.

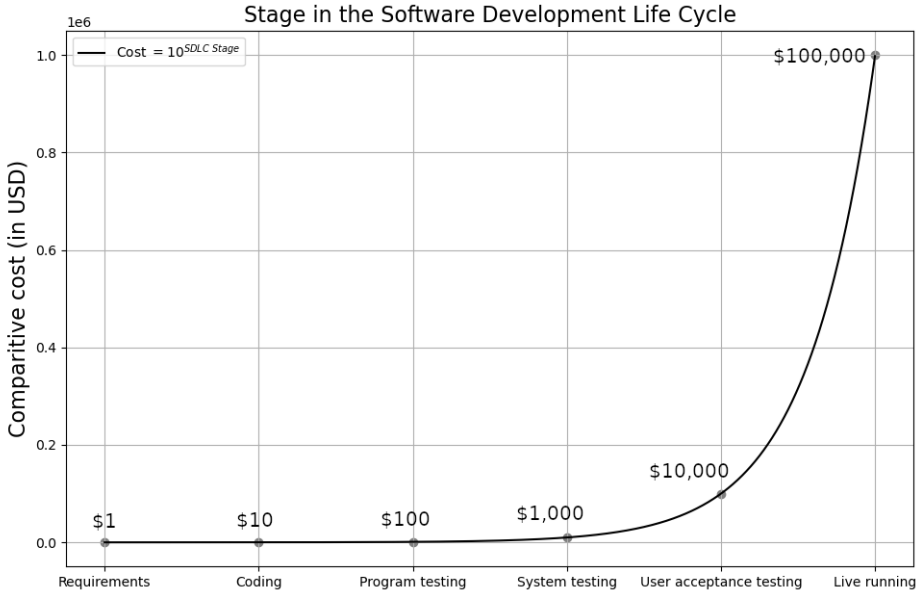

Figure 3: The *Costs of fixing software given the stage in the software development life cycble*; after Hambling et al. (2019, p. 20–21).

Observe how the cost is much larger in the last two (2) stages of software testing when compared to the first four (4) stages. This assessment shows that from a financial standpoint, it is important to identify and eliminate as many software errors in the earlier stages.

[12]Specifically, the formula for estimating cost is $\text{Cost}(\text{SDLC Stage}) = a10^{\text{SDLC Stage}}$, where $a = 1$ for illustrative purposes.

2.3 Discussion: Fluidity in the SDLC

There may be instances where software testing can be made to work in other steps of the software development life cycle:

- During the planning, defining and designing stages, some basic software testing can be used to work out the results of prototypes, and then requirements and design can change based on further information.
- During the building stage, coders and programmers can use basic software testing and debugging to make sure that certain functions work properly. Testing done by workers in the "building" stage may not be as comprehensive as a systematic software test.
- During the deployment stage, end users may send in requests for features and support tickets documenting software failures, and prototype software updates will be sent to workers in software testing for quality assurance purposes.

These software frameworks and development guidelines are not "set in stone." They serve as a guide to help software developers and engineers write better code, but because of the complexities of the real world, engineers may need to "bend the rules" a tad in order to solve problems in both an efficient and elegant manner.

3 Conclusion

The following key points are to be taken from this paper:

- The testing phase involves ensuring that the software product is solving the problem that it is intended to solve in a reliable and valid manner[13]
- The testing stage is vital to good software development because it reduces the costs that come from fixing software errors and bugs, and to prevent any damage to the greater society[14]
- The SDLC and testing methodologies are not "set in stone," and may be changed "on the fly" as demanded by the current situation.[15]

[13]This answers the question "[w]hat occurs during the testing stage of the SDLC?" from the assignment guidelines (CS-320, n.d.).

[14]This answers the question "[w]hy is the testing stage vital to a successful SDLC?" from the assignment guidelines (CS-320, n.d.).

[15]This answers the question "[a]re there any exceptions in which the testing stage would occur earlier or later than it typically does in the SDLC?" from the assignment guidelines (CS-320, n.d.).

References

Arnold, D. N. (August 23, 2000). *The Patriot Missile Failure*. University of Minnesota. Retrieved on Mar. 9, 2024 from: https://www-users.cse.umn.edu/ arnold/disasters/patriot.html

Bonnefon, J.-F., Shariff, A., & Rahwan, I. (2015). *The social dilemma of autonomous vehicles*. arXiv. https://doi.org/10.48550/ARXIV.1510.03346

Collier, R. (2017). NHS ransomware attack spreads worldwide. *In Canadian Medical Association Journal (Vol. 189, Issue 22, pp. E786–E787)*. CMA Joule Inc. https://doi.org/10.1503/cmaj.1095434

CS-320 (n.d.). *Module One Journal Guidelines and Rubric*. Southern New Hampshire University.

Gallagher, S. (Jun. 1, 2015). *Airbus confirms software configuration error caused plane crash*. Ars Technica. Retrieved on Mar. 9, 2024 from: https://arstechnica.com/information-technology/2015/06/airbus-confirms-software-configuration-error-caused-plane-crash/

Hambling, B., Morgan, P., Samaroo, A., Thompson, G., & Williams, P. (2019). *Software testing : An istqb-bcs certified tester foundation guide - 4th edition*. BCS Learning & Development Limited.

Holloway, M. (July 16, 2015). *Stuxnet Worm Attack on Iranian Nuclear Facilities*. Stanford University. Retrieved on Mar. 9, 2024 from: http://large.stanford.edu/courses/2015/ph241/holloway1/

Tutorials Point (n.d.). *SDLC - Overview*. Retrieved on Mar. 9, 2024 from: https://www.tutorialspoint.com/sdlc/sdlc_overview.htm