CI in the context of business

Isac Haglund, isacha@kth.se

May 31, 2022

1 Introduction

Software testing has been around since the first program code was written in the 1940s making the computations of a calculator. At the time, testing was mainly done by debugging the code until the teams converted to realistic simulated environments for testing. This step did opened up for a new view on what the testing contributes for software development and it made testing a natural part of the software project, which also gave birth to a new era with new ideas, methods, tools and automation.[6] However, historically businesses have struggled with lagging or neglected software testing, leading to expensive project failures and vulnerabilities within different types of organizations. In 1996, the space aircraft Ariane 5 flipped around shortly after takeoff due to a floating number variable error, the variable did not allocate enough memory. This did result in the self-destruction of the aircraft, when its thrusters exploded. The disaster did end up with a 370 m $ bill, as well as a four year delay for research on the earth's magnetosphere for the research team[1]. Further problems that have been caused by bad testing practices are security vulnerabilities, where the question arises around testing, should it occur in the CI/CD process or wait until the first stable version of a product? Continuous integration is becoming an essential part of software development and has the power to eliminate such errors. Further, security breaches extend the risk of losing large amounts of money to demands from intruders. For instance CNA Financial paid 40 m $ to a ransomware in order to regain full control of their systems in 2021[2]. Moreover, according to IBM, 1.1 trillion in assets is the total cost of bad software in 2016 alone, with a total of 4.4 billion affected users.[6] This shows that there is a need for change within the software development sector.

2 Research question

This essay aims to answer the question: How does CI contribute to a software project? And how can CI increase business performance?

3 Method

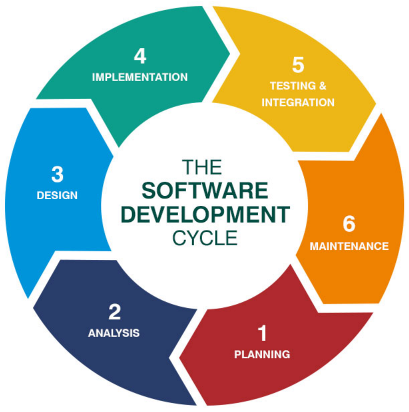
In order to answer the research question of this essay, the methodology was built on gaining multiple references stemming from credible sources such as version control producers, and other prominent companies within DevOps as well as 3 research articles. The data is gathered and assembled into a background section, which then provides material for a discussion and finally a conclusion.

4 Background

4.1 SDLC

To get a better understanding of how testing is used in a software development project we will look into a commonly used model called software development life cycle(SDLC).

SDLC is a software development process that is used in order to secure quality of the software developed according to the requirements of the customer or company, as well as keeping to the timeline and given budget from the customer or company. The development process defines what to do at specific steps of the development process which ensure a standard for developers to adhere to. For instance it tells the developer how it should handle maintenance of code, how to exchange code for new code improvements or how to change existing code. The SDLC enhances the grade of which the code is completed as well as the quality of the process itself and consists of the following 6 steps: Requirements, Design, Development, Testing, Deployment and Maintenance. The SDLC is developed for professional standards and has got an official standard name ISO/IEC 12207.[3] According to the Harness article [3], one of the biggest obstacles for a team is to deliver code faster. Using the SDLC, teams will have a better idea of what is the fastest route to deliver software and updates to the end user.



4.2 Testing

The software testing discipline ensures that applications and other software behave like intended, thereby reducing time, effort and cost while developing a properly functioning and efficient product. Testing can be divided into many different categories with different goals, depending on the requirements of the software.[6]

According to IBM, historically testing has been divided from the development in the SDLC process, pushing testing to the end of the life cycle. Meaning a tester has very little time to test the software just ahead of releasing a product, essentially giving testing a low priority overall. In contrast to this approach, Continuous Testing promotes testing continuously throughout the development process, finding bugs and errors along the way. In this way the development team avoids finding bugs at the last minute, and are able to resolve them as they go through the SDLC rather than at the end. This makes bugs cheaper to resolve, since re-coding will not affect large parts of the code base, as it could possibly do in the old fashioned testing approach.[6]

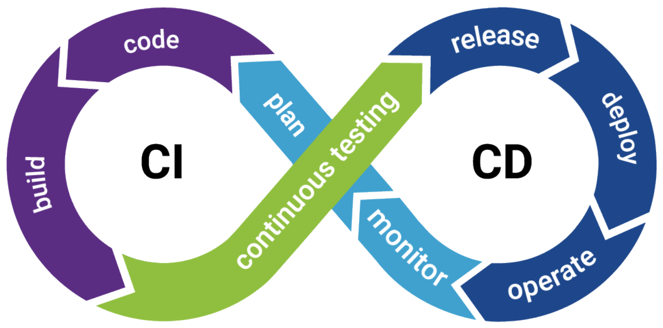
4.3 Testing and businesses

According to IBM, the process of testing has a price, however organizations can get away with a fraction of the cost in the event of proper testing and quality analysis. They also mention using proper testing management can speed up the pace of product to market.[8] Further IBM states that brand reputation is at risk if testing is neglected, due to the severeness of late delivery and poorly working applications.[6]

According to Hamilton[7] a software malfunction was the issue when a 1.2 billion $ military satellite crashed. Further, Starbucks had to temporarily shut down 60 percent of their stores in the U.S and Canada and Nissan cars had to recall 1 million cars when they intercepted an airbag malfunction, both incidents caused by a software bug. Also China Airlines plane crashed and went down, killing hundreds of people, caused by a software bug. The author makes it clear that neglecting testing can lead to dangerous and expensive consequences for humans and companies such as these. He further states that benefits from testing are lower costs, reliable applications, well functioning products and happy customers. [7]

4.4 Continuous integration

Atlassian on Continuous integration [4], Continuous integration(CI) means changing the software workflow into an automatic process of continuously putting together the work of developers into one piece of code in a main development branch for a version control system. The main branch will then run a set of tasks, trying to build the code as well as running tests, if the build is not fulfilling all predefined requirements and test cases it will be rejected by the pipeline. The task of integrating is originally made by one developer at a single workstation, while CI delegates that work on to a server instead of a human resource. Further, using CI helps avoid the issue of late discovery of bugs as well as mitigate the issues that arise when integrating the work of all developers at the end of the project. The process of CI helps teams reach requirements and code quality through solving problems that arise continuously, not waiting until the end of a project.[4] CI is also a prerequisite for Continuous Deployment or CD, which automatically deploys code on the main branch. In combination with CD, this also makes it easy to, in a fast manner, deliver new builds for your customers, which is always in a working state, continuously integrated and ready to go, which means that your next release is always a functioning piece of software. CI tools also let you track key measurements of your software's performance which is often displayed via dashboard plugins available in the tool itself. These key characteristics does visualize and clarify what problems your application is having at the moment.[15] Teams can even see positive effects from CI when it comes to morale in the development team since they can focus on coding rather than eliminating bugs. [10] However, according to Elazhary et. al, CI is not a magic formula for success and there has to be vigilance and attention to details while implementing CI to avoid CI becoming a bottleneck rather than a useful tool.[15]



4.4.1 Jenkins

Jenkins is a CI software tool for automating parts of your SDLC and chains together your different code bases into one build, further it runs your tests and automatically deploys the application. It was made by a programmer who felt it was too complex to handle multiple programmers integration at the same time, and felt the need for an automation tool when their processes were too complex. According to an article by Mohsienuddin, The tool in question is one of the most widely used automation server [16] with 1600 different plug-ins that makes Jenkins easy to use in any sort of automation project and supports in large, most tech stacks. The way to set up a jenkins project is to create a Jenkinsfile that lets you define your pipeline with a given set of parameters which tells Jenkins what to execute. [9]

4.5 Security testing

Cyber security as an industry is growing fast in the next decade as it is predicted to have an annual growth of 10 percent and to reach 376 billion $ by 2029. [12] This means the industry is reacting to the increasing amount of technology and vulnerabilities in the world of software, and security is becoming more relevant day by day.

Normally security is not a given part of CI/CD pipelines, it is all about developing, testing deploying and monitoring. Typically, security is left outside the CI/CD and implemented as a standalone feature of the software project which according to Cyberpedia makes it difficult to manage security, for instance securing the confidentiality, integrity and availability of the application in question.[14] The article also mentions three main methods for securing the CI/CD workflow which is through:monitoring and automatic scrutinizing of code, vulnerability checks and compile time security. The main purpose of doing so is saving time for the development team, avoiding big re-configurations once a problem or vulnerability surfaces.

According to Greenfield[11] and Mittal et. al, adapting to CI/CD, is making the SDLC a fast paced process but it does leave the matter of security behind, where software continuously evolves, so do the security risks. Therefore the author argues that security has to be part of the CI/CD, in other words updating security when updating software and thereby always being equipped for the latest threats towards the organization. However she also states that security should not hinder a release and that the level of security has to match the level of risk accepted by the business overall. [11]

5 Discussion & Reflection

There are many factors that can impact a company's well being, which have a major part of their revenue streams coming from software or integrated software, since the software itself has a direct connection to their customers. Firstly a company's employee retention is maybe not the most obvious, but definitely an important question when speaking of keeping talented personnel. Highly skilled programmers have no problem getting work elsewhere, which makes them a high priority for a business to satisfy in the means of a good working climate. You probably do not want to give unsatisfactory work tasks to creative and highly skilled employees. Therefore, not testing your code continuously is something that could impact how the employee feels about working at the company, detecting big problems at the end of the software process would probably be very frustrating for any programmer, since it leads to reworking, or changing big parts of your code which can be really time consuming and frustrating. This could be avoided easily by using continuous testing through a CI/CD pipeline.

Since software testing failure has such an obvious impact on both customers, company and brand reputation, and ultimately company revenue, any company should know about its importance to your business. Especially as society grows more and more dependent on software technology the cost, damage and repercussions of bad software can become extremely severe. One could imagine the impact of world wide spanning systems of societal magnitude with bad testing practices.

As Jenkins does not seem to have any competitors for CI with any obvious features that makes it better than the other, one would be safe to say that one of the most widely used (Jenkins) would be a safe bet for any company looking into working with CI. As Jenkins is one of the most used CI tools and could therefore probably be a really good asset to any company not currently working with CI and continuous testing, especially as the world of software becomes more complex by the minute. Leading development teams to have more flexible development, where cutting costs becomes an automatic effect in the long run, making it a "no brainer" for any team, no matter if the goal is being fast or cutting costs. However there is also a requirement of expertise to be able to implement a successful CI, which requires some additional resources.

Security has been a neglected subject within the CI/CD, but seems to be emerging in importance. Looking at the state of the world's political landscape, there is an urgent need not only for businesses but also for nations and governments to implement security within their software teams and in their DevOps. Also regarding security testing as hacker attacks get more normalized and the cyber security threat is real, with growing numbers having a precautionary stance on cyber security has become essential. Not being prepared could possibly lead to security breaches and seriously expensive consulting fees for hiring response teams at a potential breach, and in times of war catastrophic data leaks could be a fact.

6 Conclusion

Businesses have little to lose in the long term regarding continuous testing and CI, it results in quality products, keeps customers loyal and gives a stainless brand image, ultimately resulting in increasing revenue and even saving human lives. The key takeaway of this essay is that Continuous integration should be part of every software process, if no obvious reasons states otherwize, making the use of CI a natural path to take for development teams. However, CI should be implemented with attention to details. Further, security should be considered as a natural part of the CI/CD pipeline in order to cope with, and be up to date with current cyber security threats and vulnerabilities. For future research it is recommended to have an even more credible reference list with sources originating mainly from scientific articles and reports.

7 References

[1] https://www.bugsnag.com/blog/bug-day-ariane-5-disaster
[2] https://www.cnet.com/personal-finance/crypto/a-timeline-of-the-biggest-ransomware-attacks/
[3] https://www.tutorialspoint.com/sdlc/sdlc$_o$*verview.html*
[4]*https : //www.atlassian.com/continuous − delivery/continuous − integration*
[5]*https : //www.atlassian.com/continuous − delivery/continuous − integration/how − to − get − to − continuous − integration*
[6]*https : //www.ibm.com/topics/software − testing*
[7]*https : //www.guru*99*.com/software − testing − introduction − importance.html*
[8]*https : //www.ibm.com/products/ibm − engineering − test − management*
[9]*https : //www.infoworld.com/article/*3239666*/what−is−jenkins−the−ci−server−explained.html*
[10]*https : //hackernoon.com/best − cicd − tools − to − consider − in −* 2022
[11]*https : //devops.com/continuous − security − the − next − evolution − of − ci − cd/*
[12]*https : //www.fortunebusinessinsights.com/industry − reports/cyber − security − market −* 101165
[13]*Mittal, K, Sharma, M, Gupta, M, Dr.Sheoran; DevSecOps : ABoontotheITIndustry*2021.
[14]*https : //www.paloaltonetworks.com/cyberpedia/what − is − the − ci − cd − pipeline − and − ci − cd − security*
[15]*Elazhary.O, Werner.C, Li.Z, Lowlind.D, Ernst.NandStorey.M. : UncoveringTheBenefitsAndChallengesOfContinuousIntegrationPractices*
[16]*Mohsienuddin, S : ContinuousIntegrationandAutomation. InternationalJournalofCreativeResearchThoughts(IJCRT), ISSN :* 2320−2882*, Volume.*4*, Issue*3*, pp.*938− 945*, July*2016.