**UNIVERSITY OF WATERLOO**

Faculty of Mathematics

**Is continuous integration worth implementing?**

Flynn Inc.,

Mississauga, Ontario

Phat Tran

20619470

2A Mathematics

August 2017

**MEMORANDUM**

To: Alan Zych, Mobile and Web Development Manager

From: Phat Tran

Date: August 25, 2017

Re: Work Report: Is continuous integration worth implementing?

Dear Alan,

I have prepared the enclosed report “Is continuous integration worth implementing?” for my 2A work report and for the Mobile Development Team. This report is part of the four required reports that I need to complete as part of my Bachelor of Computer Science Co-op degree requirements. It has not received an academic credit.

During my co-op term, my job is to develop a web application for the Distributed Storage System (DSS) for the company and composes tests for the Flynn Service App, both web and mobile.

The Faculty of Mathematics requests that you evaluate this report for command of topic and technical content/analysis. Following your assessment, the report, together with your evaluation, will be submitted to the Math Undergrad Office for evaluation on campus by qualified work report markers. The combined marks determine whether the report will receive credit and whether it will be considered for an award.

This report was written entirely by me and has not received any previous academic credit at this or any other institution.

Thank you for your assistance in making this report.

*Phat Tran*

*(signature)*

# **Table of Contents**

[Table of Contents 3](#_Toc488339185)

[Executive summary 4](#_Toc488339186)

[Introduction 5](#_Toc488339187)

[Analysis 6](#_Toc488339188)

[The Traditional Development Process 6](#_Toc488339189)

[What is Continuous Integration? 9](#_Toc488339190)

[Benefits and Costs of Continuous Integration 10](#_Toc488339191)

[Conclusion 15](#_Toc488339192)

[References 16](#_Toc488339193)

**List of Figures**

[Figure 1 - Waterfall Process 6](#_Toc488339176)

[Figure 2 - Github's Outage 11](https://d.docs.live.net/dae0a0658347da83/ws/WORKREPORT-2/firstDraft_20619470_MathWorkReport.docx#_Toc488339177)

[Figure 3 - Failed build is Sent to the People Involved 13](https://d.docs.live.net/dae0a0658347da83/ws/WORKREPORT-2/firstDraft_20619470_MathWorkReport.docx#_Toc488339178)

# **Executive summary**

The purpose of this report is to analyze the overall impact of practicing continuous integration (CI) in software development. To achieve that, the main focal points of the report will be around the immediate cost and the delayed long-term benefits of CI. Finally, the conclusion will be a short recap of the analysis section, which leads to the answer of our main question: is CI worth implementing?

# **Introduction**

With recent advancement in technology, people’s lives have been improved drastically. People spends more time than ever on various social media platforms like Snapchat, Instagram and consumes many technology-required services like Amazon and Uber. Billions of customers across the world that are served daily; thus, adding new features while keeping their high-quality well-tested product bug-free is no longer an advantage. It’s becoming the standard requirement for these tech giants. And, on another note, with the entry barrier being extremely low, these top technology companies need to improve their product faster than the competition. Therefore, an innovative strategy needs to be formed to enhance the speed of product iteration, not just to gain the advantage over other competitors in this highly competitive market, but more importantly, to retain companies’ hard-earned customers’ attention.

In this report, we will examine the pros and cons of CI practice, which will let us determine if your software project should apply CI. Therefore, it is assumed that readers have a basic understanding of traditional development and deployment cycle, though a quick refresh of such process will be given. All advantages and disadvantages will be considered in comparison to the shown example of traditional development process.

# **Analysis**

## The Traditional Development Process

### A Quick Recap

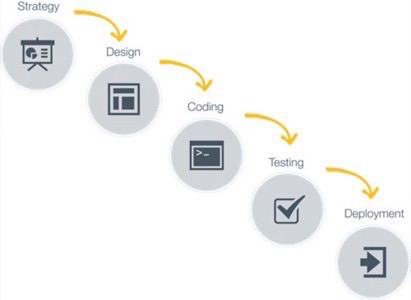
 Software companies have many methods to ship their products to clients. One of them is called Waterflow.

Figure 1 - Waterfall Process

After getting input from clients or end users, the manager comes up with a set of features that are in the final product. Then the software architect designs the architecture of the product, giving developers a big picture, so that developers can program each individual pieces of the program. After all components are implemented, the product will then be tested by QA team and by the tests that developers write. After all the tests pass, the product is deployed, ready to be delivered to clients, or to the end users.

### Problems from the Traditional Process

Despite being implemented for such a long time, there are many disadvantages that are not so obvious.

Harder to debug:

Due to the nature of the traditional development process where there exist a “large time gap between delivery of end-product and requirements elicitation” (UK Essays, 2015), software developers are likely to lose context of the first couple features. If an error occurs near the end of a release cycle, it is extremely difficult for one to trace the bugs and determine which module causes the error as developers are unlikely to remember all details of the code they wrote a couple months ago. When that happens, a junior will get help from other senior engineers; however, it is not an easy task for the senior engineers because of the following reason.

Long code reviews:

Since many new features are packed within one release, a senior engineer has to read through a massive changelog. According to Erik Dietrich – a consultant specializing in code analysis, “getting together in a conference room for days to go over a months’ worth of code was just awful” (Dietrich, 2015). Certain new modules may, for instance, contain different version of dependencies of other previous components of the product, and this could produce an error at runtime. Without reading through all the new features requirements, one may not be able to spot the mistake. And without frequent feedback from senior developers from the beginning, a junior engineer can amass an enormous amount of terribly written code, and this only makes the code reviewing job harder.

Slower feature release:

With the reasons described above, the traditional development workflow leads to slower feature release. In the beginning of a project, there may not be a lot of features, and thus, development may seem fast and efficient. But as time goes on, as the project gets more complex, the time it takes for one to troubleshoot errors will delay the release cycles. More time has to be spent debugging, refactoring, re-designing, or even trashing the old, unmaintainable code. This forces developers to slow down substantially when developing a new feature, and even so, mistakes are still bound to happen.

Non-reliable progress metric:

Last but not least, not merging new code into codebase frequently leads to a false sense of accomplishment. Software programmers have a tendency to mentally check the feature off the to-do list despite not 100% sure that the new code will work in harmony with the old ones. Thus, when reporting about through either scrum meeting or tracking work logs, one may inadvertently report the false progress, leading to false expectation from managers. When bugs came during the code merge, not only does one waste a lot of time to debug, a software engineer also loses reputation, or even the job if the deliverables are vital to the survival of the business.

Therefore, software development needs to replace such inefficient workflow that will solve the mentioned problems, introducing continuous integration.

## What is Continuous Integration?

### Definition of Continuous Integration

There’s no concrete definition of continous integration (CI). Some people may claim that CI means that whenever someone deploy new code changes to the codebase, there exists a system to re-build the software and deploy it. Other may say that CI means that whenever a new change is introduced to the code repository, the system compiles the new code changes against a set of pre-written unit tests to ensure that it’s working as expected. However, the most fundamental idea of CI that it’s a “software development practice where members of a team integrate their work frequently” (Fowler, 2006). Therefore, whether the system compiles the software or executes tests on the new changes, if the newly-added code is introduced into the production branch in a frequent manner, it is considered CI. From now onward, this will be the official definition that we will consider pros and cons for.

### Continuous Integration in real world

Famous software companies such as Google, Facebook, Microsoft, Amazon, Netflix, Adobe all practices CI, and they’re well known for reliable software products with uptime nearing almost 100%. An investment automation company, Wealthfront, has been applying CI for the past 7 years (Perez, 2010). Wealthfront is now valued at 700 million US dollars.

## Benefits and Costs of Continuous Integration

Many technology companies, especially the one with massive userbase or has an enormous valuation deploy this practice. However, not everyone understands the reason behind that strategy. Thus, it is my attempt to consider the cost and benefits of CI, and wish to derive the insight from the analysis in the following sections.

### Cost

Requires infrastructure expertise and knowledge:

In order for a software team to practice CI, a system has to be set up. This is not an easy objective, and sometimes requires a dedicated team to achive such task. There are countless scenarios for one to consider. For instance, some of these questions may be: how to ensure that the new code will not break the existing codebase? Where does the code go while waiting for code review to be done? What if the old codebase changes while one is currently under development? What if two or more developers work on the same feature and create different implementations? These are just some of the potential issues that one needs to think about, and it would requires knowledge in many aspects including, but not limited to, programming workflow and software deployment practice.

Depends on third-party:

To abstract such complex job, many companies provide these plug-and-play infrastructure-as-a-service where it is substantially easier for software engineers to commit their code and do not have to worry about other problems when merging code.

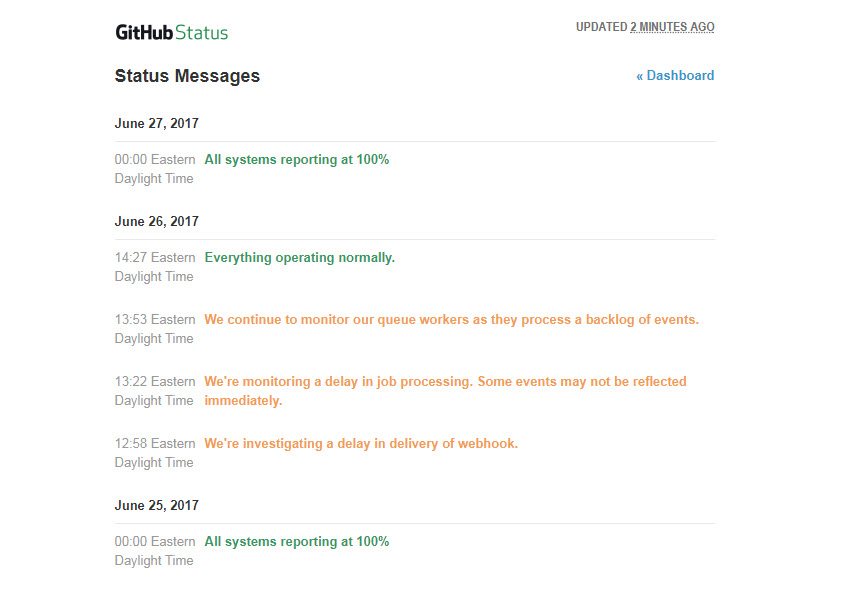


Figure 2 - Github's Outage

However, this also implies that the software project not only costs more to produce, but also has to depend on third-party for support when problems arise. In addition, team’s productivity is affected by the availability of infrastructure service.

Slowed productivity at the beginning:

Last but not least, people have the tendency to resist changes, even though these people are in the technology industry - an industry that changes the most frequent out of all the sectors. Since the code-merging job is no longer reserved for senior engineers, every developer is expected to know how to merge their code without breaking others’. Thus, it is normal for a team to slow down during the first few weeks of implementing CI, partly due to the resistance to constant changes, another part due to time spent reverting typical mistakes such as clearing commit history and losing unsaved code. Anika, a mobile and web developer co-op at Flynn, claims that she spends more time learning Git and CircleCI, which are the tools used to integrate new code into existing project, than contributing something of use for the first few weeks of her summer intership (Morshed, 2017).

### Benefits

These benefits that listed below are based upon the assumption that everything is set up correctly. There should be a code versioning service, a build automation service, and a testing infrastructure that will alert relevant members of the project if anything happens to the product.

Smaller code review:

When practicing continuous integration, one can expect to see a smaller code review blob (< 400 lines), instead of a typical amount of code for a sprint, which can range up to tens of thousands of lines. This will allow the code reviewer to be more effective, finding around 70% to 90% of existing defects (Cohen, 2011).

Detect failures sooner:

When a developer pushes code to a centralized code versioning system, a build will be triggered. The new changes that one made will be tested immediately before applying to the final product. If the new changes create some sort of errors or conflicts, the contribution will be rejected, with an email message notifying all people that are involved in the project.

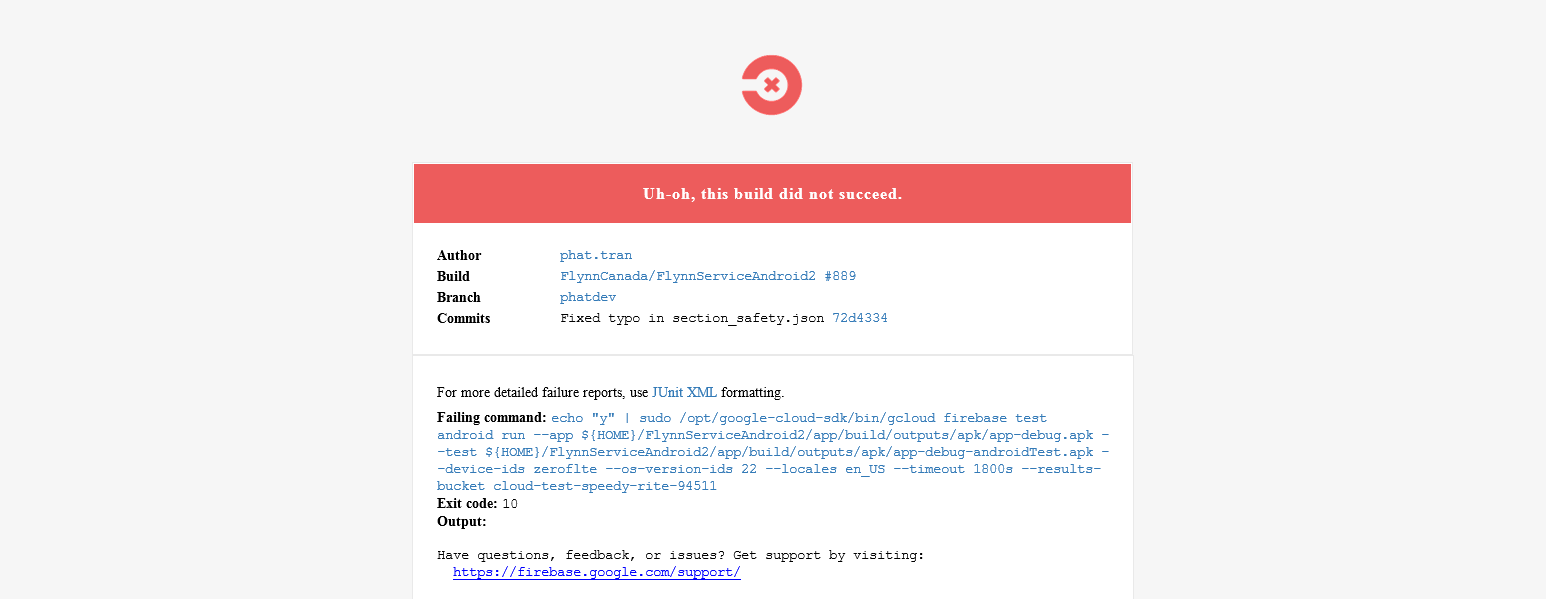
If they pass, the product incorporates the new changes, which leads to the next point.

Figure - Failed build is Sent to the People Involved

Shorten feature-release cycle:

When changes are being filtered and added automatically, it speeds up the feature cycle. Developers no longer have to finish a set of features, wait for it to compile, and check the status of the build. Continuous integration encourages developers to push individual feature immediately as the integration and testing process is automated. This will allow developers to release more features in a shorter time span, comparing to the traditional waterfall process. Besides, small feature is always easier to manage than a large set of features, which plays a huge role in why the release cycle is shorter.

More profit in the long run:

Lastly and most importantly, consumer, businesses will always go to the first available products that solve their problems. When a product improves faster than the competition, the company that deliver the product sooner or release a desirable feature sooner will have a massive advantage over other slightly differentiated products from other companies. With that being said, it is in the company’s control to assure the quality of its product and provide adequate customer support to retain the client base it builds from being first on the market, from practicing continous integration.

# **Conclusion**

Despite the enormous technical knowledge and resources required up front when setting up the build and testing system, CI has proved its worth through faster product iteration speed, which ultimately and eventually leads to tremendous upside in terms of monetary values and market opportunities. Therefore, investing time and effort in a proper CI system will pay enormous dividends in the long run. However, startup’s software product may not be experiencing the same benefit due to the limited amount of paying clients and the huge immediate cost that one must pay in the beginning. Therefore, in order to get the best of both worlds, tech startups have several options: they can pitch and score investment from venture capital fund or other investors, or it can develop a smaller-scaled CI system and only apply on certain core features, the ones that are absolutely required and are being paid for by clients.

# **References**

Cohen, J. (2011, Jan 25). *11 proven practices for more effective, efficient peer code review*. Retrieved July 20, 2017, from IBM Developer Works: https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review/index.html

Dietrich, E. (2015, October 8). *What to avoid when doing code reviews*. Retrieved July 8, 2017, from SmartBear: https://blog.smartbear.com/code-review/what-to-avoid-when-doing-code-reviews/

Fowler, M. (2006, May 01). *Continuous Integration*. Retrieved June 5, 2017, from MARTINFOWLER.COM: https://martinfowler.com/articles/continuousIntegration.html

Morshed, A. (2017, July 11). Mobile and Web Developer Co-op. (P. Tran, Interviewer)

Perez, P.-L. (2010, September 9). *DevOps Cafe on Continuous Deployment and Operations Dashboards at kaChing*. Retrieved June 25, 2017, from Wealthfront Engineering Blog: http://eng.wealthfront.com/2010/09/09/devops-cafe-on-continuous-deployment-and-operations-dashboards-at-kaching/

UK Essays. (2015, March 23). *The Problems Of Traditional Software Development Methods Information Technology*. Retrieved July 8, 2017, from UK Essays: https://www.ukessays.com/essays/information-technology/the-problems-of-traditional-software-development-methods-information-technology-essay.php?cref=1