

# Continuous Integration of Software Management

Gautam Kumar, Prof. David Socha  
Computing and Software Systems  
University of Washington Bothell  
gautamk@uw.edu, socha@uw.edu

## ABSTRACT

Continuous Integration, A practice where developers integrate frequently[9] is considered an integral part of Agile development. CI is often thought to be a safety net that prevents developers from deploying broken code to production.

A similar process and ideology of continual improvement and integration can be applied to Software Management, The art and science of planning and leading software projects [10].

This paper attempts to explore the process of creating such an integration practice within an Organisation.

*Keywords:* Software Management, Agile development, Continuous Integration

## INTRODUCTION

Martin Fowler in his seminal article[3] on Continuous Integration describes it as a practice where members of a team integrate their work frequently and each integration is verified by an automated build to detect problems quickly.

In the context described above integration is the process of combining the work of all the developers of a project into a cohesive software. On the other hand if considered generically, Integration can be thought of as a practice of combining the work of multiple people and verifying its correctness. Using a similar collective, Continuous integration could be the process if integrating the work of multiple people after specific events or set periods of time.

## CONTINUOUS IMPROVEMENT AND INTEGRATION

The practice of continuous improvement has evolved as a response to improvements in the way software is delivered and used by customers. With the introduction and popularisation of the internet in the early years of the 21st century many consumers and enterprises started using the internet and web technologies as more than just an information directory. Web sites evolved into web apps which lead to the rise of Web 2.0 [7]. This evolution to Web 2.0 required innovations with web server technologies which also introduced significant complexity and risk to the stability of the software systems [7].

Continuous Integration evolved in response to the increased risk to software stability and attempts to act as a safety net [3] to developers. Continuous Integration doesn't actually prevent or fix bugs, it merely makes discovering bugs easier and faster.

## SOFTWARE MANAGEMENT AND CONTINUOUS INTEGRATION

The inherent complexity in software management is best described by the law of leaky abstractions[8] which states that all non-trivial abstractions leak to some degree.

Software management is in essence a process which tries to abstract away the chaos of managing an inherently complex task with the help of developers in an ever-evolving environment. So naturally Software Management tends to be a complex task which needs to be continually improved to adapt to changes in the environment such as new labour laws, new technologies, a new generation of customers and developers.

## CI AND AN EVOLVING ENVIRONMENT

In today's fast changing tech landscape, traditional definitions of Continuous integration and improvement are being challenged to adapt quickly. In the following sections we discuss some potential factors which affect Continuous Integration deployments in an organisation.

### *Evolving Customer*

The previous generation of customers were primarily using a desktop computer with a reliable internet connection and a large display. This customer base is quickly moving towards mobile devices as their primary computing device where displays are much smaller, internet connection is spotty and there are restrictions on power consumption.

These changes require the developers to adapt to the environment which inherently needs an overhaul of the software management process. <sup>[CitationNeeded]</sup>

### *New technologies*

Virtual reality, Internet of Things and Machine learning are the hottest consumer technologies evolving as of this writing and all these new technologies promise a new dimension of interactivity and user experience. The promise of a new user experience and interactivity comes with added complexity and risk, for example Virtual reality requires new hardware and software capabilities which increases the potential chances of failure and development teams need to evolve new techniques of managing this risk. Traditional continuous integration many not work in this scenario. <sup>[CitationNeeded]</sup>

### *Product life cycles and Business models*

Web 2.0 has made a significant impact on the way users and companies buy and sell software. The traditional model of expensive one-time licensing fee has evolved into a small monthly subscription fee and companies are expected to deliver constant updates to their software product.

A move to the subscription model from the licensing model might require implementing a continuous delivery strategy which might involve simplifying and streamlining the CI system to perform a large number smaller builds rather than single large builds.

[CitationNeeded]

### *Open source*

Open source software has had a significant impact on the way companies develop software. Open source software helps developers build software faster and enables tapping into the collective wisdom of a larger community of developers. This translates into large savings for companies which is why many organisations are embracing the trend of using open source components within their technology stack [1] [4].

Introducing and integrating open source software isn't free, though there is no upfront licensing cost for using open source software. Using open source software requires dedicated effort to adapt to the structure of the software library in question. Security is another concern when using open source software so keeping the system up to date with patches becomes a mandate to preserve the privacy of users and also to fulfil certain legal requirements.

### *New generation of developers*

A whole new generation of developers have grown up embracing collaborative development techniques such as Q&A sites, open source software and Hackathons and access or lack thereof to such a community within an organisation may have a significant impact on the productivity of developers [11].

### *Effect on Businesses and Software management*

The aforementioned factors can have a significant impact on the projects, people and incidentally the core business of a company. So to achieve business agility and quickly adapt to an ever changing landscape, companies which rely on software technologies need to be able to quickly adapt their software management process [5].

## TODO: SCOPE OF SOFTWARE MANAGEMENT

### *TODO People Management*

*Hiring:*

*Culture:*

*Expectations:*

*Organisation structure, Flat, Hierarchy or Hybrid:*

### *TODO Project Management*

*Technical choices:*

*Prioritizing projects:*

*Choosing the right people:*

### *Solving the problem right:*

### *TODO Product Management*

*Translating business need to Product requirements:*

*Solving the right problem:*

### *TODO Process Management*

*Need for process:*

*Rigid Process vs Flexible trust: [2]*

## CI IN SOFTWARE MANAGEMENT

[CitationNeeded]

Continuous integration has been well defined from the software development perspective, definitions from the side of Software Management on the other hand are quite sparse.

The value of Continuous integration can only be seen in an organisation that values continuous improvement of the organisation as a whole as opposed to merely using CI as a tool for software development.

The easiest area to define Continuous Integration within the scope of Software management would be Process Management. CI in terms of Process Management could be thought of as a way to verify that all internal processes within an organisation comply with the expectation and solve the intended problem. Some potential implications of a meta-process of Continuous Integration could be weeding out of complicated processes which affect productivity, Simplifying processes to reduce bottlenecks and potentially automation to simplify implementation and verification. There are many tools especially in the BPMN space which offer a simple way to automate and integration internal processes with existing infrastructure.

## IMPLEMENTING CI IN SOFTWARE MANAGEMENT

Continuous Integration and Improvement by nature is an iterative process thus implementing within the context of Software management requires the management team to follow a disciplined process of continuous evaluation and an evolution.

### *Adoption*

The initial adoption process of any new Software management model would be significantly different from integrating new team members into an existing model. For example, some the factors which affect the success of adopting Agile methodologies in an organisation are Customer collaboration, satisfaction and commitment, Communication and negotiation skills of the team and Cultural factors [6].

These success factors differ when brining in new team members. For example a team member fresh out of college might adapt to an existing process at an organisation with ease while a seasoned developer who is loyal to the waterfall model might have a hard time adapting to the rapid pace of an agile organisation.

As teams being to understand the expectations of their customers and their organisations the previously defined software process evolves thus as time goes on each team / organisation starts building a unique variation of the agile process.

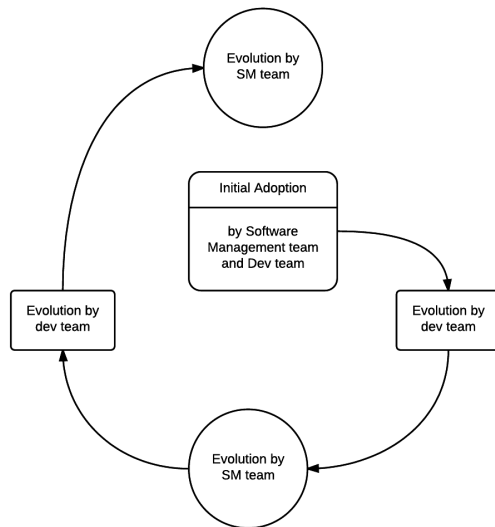


Fig. 1. The Software management team has to evolve the adoption process as the dev team deviates from the initial Software development model

In this context initial adoption is usually the easiest task as the whole team is collectively working towards making sure that Agile adoption succeeds. Adoption by new team members on the other hand depends on the new team member, the Software management team and the culture of the organisation.

It is the responsibility of the software management team to enable new developers to adopt and adapt to the unique process of the dev team they are going to be working with.

### Success criteria

The software management team needs to have a clear and evolving success criteria which monitors the each new hire and how well they are able to adapt to their new teams. This differs from Definition of Done (DoD) based on the fact that DoD is merely a set of activities which need to be performed to get the user story to a shippable state while Success criteria is a broader set of expectations from the candidate which are recorded based on subjective observations.

### REFERENCES

- [1] Howard Baldwin. 4 reasons companies say yes to open source, January 2014.
- [2] Mary J. Benner and Michael L. Tushman. Exploitation, exploration, and process management: The productivity dilemma revisited. *Academy of management review*, 28(2):238–256, 2003.
- [3] Martin Fowler and Matthew Foemmel. Continuous integration. *Thought-Works*) <http://www.thoughtworks.com/Continuous Integration.pdf>, page 122, 2006.
- [4] Mathias Lemmens. Open Source success. *GIM Int*, 22(10):8–9, 2008.
- [5] Lars Mathiassen and Jan Pries-Heje. Business agility and diffusion of information technology. *European Journal of Information Systems*, 15(2):116, 2006.
- [6] Subhas Chandra Misra, Vinod Kumar, and Uma Kumar. Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, 82(11):1869–1890, November 2009.

- [7] Tim O’reilly. What is Web 2.0: Design patterns and business models for the next generation of software. *Communications & strategies*, (1):17, 2007.
- [8] Joel Spolsky. The Law of Leaky Abstractions - Joel on Software, November 2002.
- [9] Daniel Ståhl and Jan Bosch. Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, 87:48–59, January 2014.
- [10] Andrew Stellman and Jennifer Greene. *Applied software project management*. ” O’Reilly Media, Inc.”, 2005.
- [11] Bogdan Vasilescu, Stef van Schuylenburg, Jules Wolms, Alexander Serebrenik, and Mark G. J. van den Brand. Continuous integration in a social-coding world: Empirical evidence from GitHub. \*\*Updated version with corrections\*\*. *arXiv:1512.01862 [cs]*, pages 401–405, September 2014. arXiv: 1512.01862.