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[8] 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 [7].

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[2] 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 [5]. 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 [5].

Continuous Integration evolved in response to the increased risk to software stability and attempts to act as a safety net [2] 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[6] 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. *[CitationNeeded]*

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. *[CitationNeeded]*

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] [3].

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.

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 [4].

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.

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.

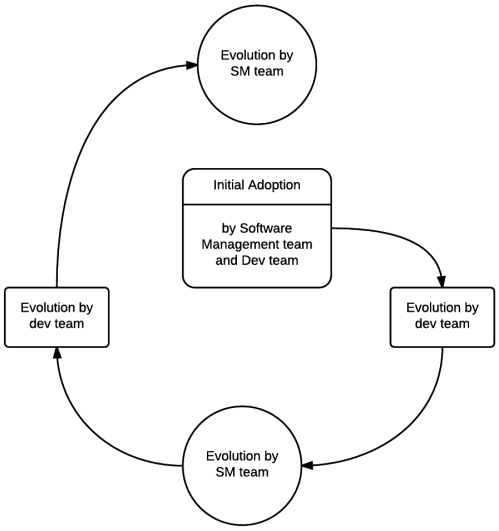

Fig. 1. The Software management team has to evolve the adoption process as the dev team deviates from the initial Software developement model

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.
[2] Martin Fowler and Matthew Foemmel. Continuous integration. page 122.
[3] Mathias Lemmens. Open source success. 22(10):8–9.
[4] Subhas Chandra Misra, Vinod Kumar, and Uma Kumar. Identifying some important success factors in adopting agile software development practices. 82(11):1869–1890.
[5] Tim O'reilly. What is web 2.0: Design patterns and business models for the next generation of software. (1):17.
[6] Joel Spolsky. The law of leaky abstractions - joel on software.
[7] Andrew Stellman and Jennifer Greene. *Applied software project management.* " O'Reilly Media, Inc.".
[8] Daniel Sthl and Jan Bosch. Modeling continuous integration practice differences in industry software development. 87:48–59.