Continuous Integration of Software Management

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

ABSTRACT

Continuous Integration, A practice where developers integrate frequently [19] 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 [20].

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 [9] 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 of merging the work of multiple people after specific events or set periods of time.

EVOLVING ENVIRONMENT

The outcome of a well managed software organisation are products which provide value to the customer and satisfy their need. In today's fast changing technology landscape the needs and expectations of the customers is always evolving. Such an evolution is affected by various factors, some of which are discussed in the following sections.

Changing customer expectations

The first few generations of software customers were primarily educational institutions and enterprises where software was merely a means to an end. Customers today, especially in the consumer software market expect an experience rather than just a tool that solves a problem. This explains the reason why organisations today have roles for professionals who work towards streamlining the way people use software [10] [7].

Customers have also tuned to the idea that organisations which build software systems are expected to support their products for a long period of time when compared to hardware manufacturers who can get away with launching a new version of their product every year and deprecating old products every 2 years. This could potentially explain the meteoric rise in value of customer support services the market for which is projected to be valued at $81 billion [22].

These changes in the customer expectation landscape requires new strategies to be incorporated and verified within the software management process.

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 [1]. 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.

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 [15] [6].

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 [2] [11].

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 [21].

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 [12].

Continuous Improvement

The American Society of Quality define Continuous Improvement to be "The ongoing improvement of products, services or processes through incremental and breakthrough improvements." [8]

Continuous Improvement as a concept has existed for a long time. We can find mentions to continuous Improvement in texts dating back to the mid-19th century [16].

During the Industrial revolution of the mid-19th century workers were often trained not to think but rather just perform just their task. Thus the task of thinking was effectively cut off from the task of doing, which lead to the thinkers not understanding the practical restrictions of the real world and the increasing complexity of the systems [16]. The same concept applies even today with software systems. In large organisations developers are often disenfranchised from the decision making process.

This separation of thinkers from the practical implications of real world lead to the development of Continuous improvement which emphasizes a continuous stream incremental innovation [5].

Continuous Integration

Continuous improvement incentivizes a structure where hundreds of people are collectively working on the same basic problem set [5]. Continuous improvement brings along with it an explosion of ideas and as some of those ideas implemented a surge in the complexity of the system. Such a surge in complexity eventually leads to either the quality or velocity of development to decline [23] .

Continuous integration thus evolved as a mechanism to manage the risk associated with complex software systems [23] and attempts to provide a safety net [9] to developers.

Though continuous integration doesn't actually fix bugs, it offers a lot of benefits which help in

improving the overall quality of the project such as Improved communication, Project predictability and increased developer productivity [18]. So Continuous integration in many ways can be considered a modern method of complexity control [3].

Software Management and Continuous Integration

The inherent complexity in software management is best described by the law of leaky abstractions [17] which states that all non-trivial abstractions leak to some degree.

The Law of leaky abstractions

An abstraction in the context of computers and software refers to a mechanism that tries to hide the complexity of the underlying system while still offering a useful interface to interact effectively with the underlying system. Abstractions are meant to simplify the process of interacting with an inherently complex system by hiding away certain irrelevant details.

Though abstractions vastly simplify interactions with a complex system, the abstractions are still limited by the capabilities of underlying system and thus some serious failures always trickle up through abstractions and surprise the user in unexpected ways.

A classic example of leaky abstractions would be Operating Systems. On the windows operating system any failure in the underlying hardware or its drivers can potentially lead to a BSOD (Blue screen of death) in which the computer displays a cryptic message on a blue screen and prompts the user for action. Though BSOD is a rare occurrence under normal operating conditions, its appearance often perplexes an average user who is unable to make a decision on the next course of action [14].

Abstractions in Software Management

Software Management as a subset of Project management is primarily concerned with balancing the needs of stakeholders with the abilities of developers to deliver value.

Software management practices abstract away the inner workings processes in the organisation. For example Agile methodologies such as Scrum and Kanban are an abstraction which uses best practices from Time management, Team communication, Performance management and Theory of constraints to derive value from developers collaborating within a team.

Another example of abstraction in Software management in teams would be embracing diversity which is an abstraction over Psychology and Economics to help teams take varied view points on any particular problem.

CI in Software Management

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. As with any non trivial abstraction, software management is leaky and implementing continuous integration into Software management practices could potentially offer benefits similar to software development.

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: [4]

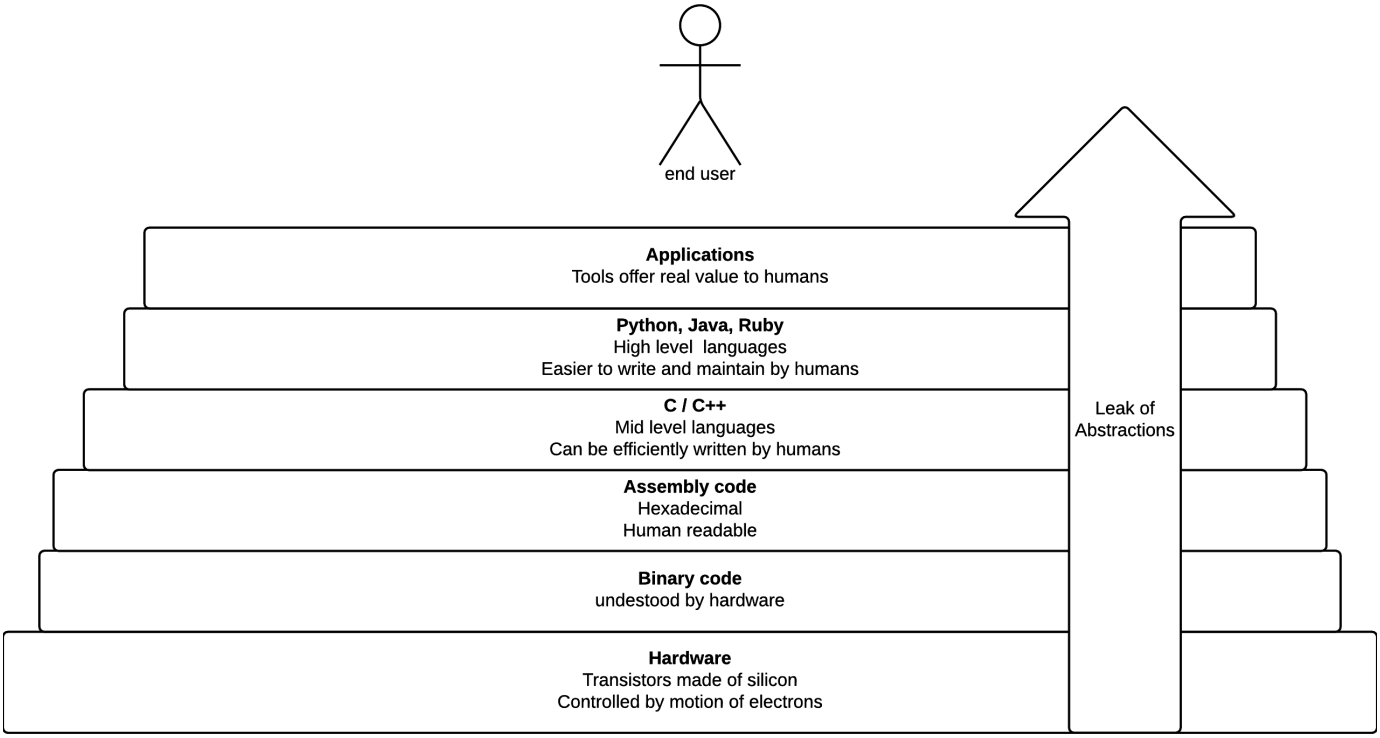

Fig. 1. The law of leaky abstraction: All non-trivial abstractions, to some degree, are leaky.

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 [13].

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

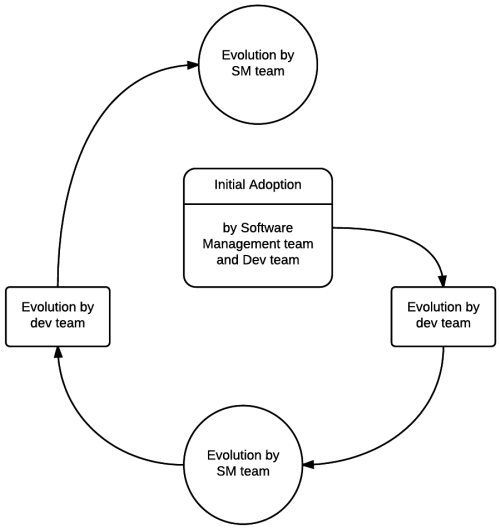

Fig. 2. The Software management team has to evolve the adoption process as the dev team deviates from the initial Software developement model

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.

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] Angel Bachvarov, Stoyan Maleshkov, Polina Häfner, and Jurica Katicic. Design-by-the-customer through virtual reality. pages 561–566, 2009.

[2] Howard Baldwin. 4 reasons companies say yes to open source, January 2014.

[3] Kent Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[4] 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.

[5] J. Bessant, S. Caffyn, J. Gilbert, R. Harding, and S. Webb. Rediscovering continuous improvement. *Technovation*, 14(1):17–29, February 1994.

[6] Christina Di Valentin, Tobias Weiblen, Anton Pussep, Markus Schief, Andreas Emrich, Dirk Werth, and Peter Loos. Measuring Business Model Transformation. In *Proceedings of the 9th European, Mediterranean and Middle Eastern Conference on Information Systems (EMCIS), Munich, Germany*, pages 1–9, 2012.

[7] Matthew Dixon, Karen Freeman, and Nicholas Toman. Stop Trying to Delight Your Customers, July 2010.

[8] American Society for Quality. Continuous Improvement Model - Learning Resources| ASQ.

[9] Martin Fowler and Matthew Foemmel. Continuous integration. *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, page 122, 2006.

[10] Jon Kolko. Design Thinking Comes of Age, September 2015.

[11] Mathias Lemmens. Open Source success. *GIM Int*, 22(10):8–9, 2008.

[12] Lars Mathiassen and Jan Pries-Heje. Business agility and diffusion of information technology. *European Journal of Information Systems*, 15(2):116, 2006.

[13] 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.

[14] Scott ROSENBERG. The Law of Leaky Abstractions. *Technology Review*, 110(1):46–47, February 2007.

[15] Markus Schief, Amir Bonakdar, Tobias Weiblen, and others. Transforming Software Business Models into Business Processes. In *ICEIS (3)*, pages 167–172, 2012.

[16] Dean M Schroeder and Alan G Robinson. America's Most Successful Export to Japan: Continuous Improvement Programs.

[17] Joel Spolsky. The Law of Leaky Abstractions - Joel on Software, November 2002.

[18] Daniel St\a ahl and Jan Bosch. Experienced benefits of continuous integration in industry software product development: A case study. In *The 12th IASTED International Conference on Software Engineering,(Innsbruck, Austria, 2013)*, pages 736–743, 2013.

[19] Daniel Ståhl and Jan Bosch. Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, 87:48–59, January 2014.

[20] Andrew Stellman and Jennifer Greene. *Applied software project management*. " O'Reilly Media, Inc.", 2005.

[21] Bogdan Vasilescu, Stef van Schuylenburg, Jules Wulms, 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.

[22] Stefan Wuyts, Aric Rindfleisch, and Alka Citrin. Outsourcing customer support: The role of provider customer focus. *Journal of Operations Management*, 35:40–55, May 2015.

[23] Yury V. Zaytsev and Abigail Morrison. Increasing quality and managing complexity in neuroinformatics software development with continuous integration. *Frontiers in Neuroinformatics*, 6:31, 2013.