Extending the "Development Pipeline" Towards Continuous Deployment and Continuous Experimentation: A Case Study in the B2B Domain

Olli Rissanen

Master's thesis
University of Helsinki
Department of Computer Science

Helsinki, April 15, 2014

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Laitos — Institution — Department Department of Computer Science

Tekijä — Författare — Author
Olli Rissanen

Työn nimi — Arbetets titel — Title
Extending the "Development Pipeline" Towards Continuous Deployment and Continuous Experimentation: A Case Stud

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year April 15, 2014	Sivumäärä — Sidoantal — Number of pages 4

Tiivistelmä — Referat — Abstract

Currently more and more software companies are moving to lean practices, which often include shorter delivery cycles and thus shorter feedback loops. However, to achieve continuous customer feedback and to eliminate work that doesn't generate value, even shorter cycles are required. In continuous deployment the software functionality is deployed continuously at customer environment. This process includes both automated builds and automated testing, but also automated deployment. Automating the whole process minimizes the time required for implementing new features in software, and allows for faster customer feedback. However, adopting continuous deployment doesn't necessarily mean that more value is created for the customer. While continuous deployment attempts to deliver an idea to users as fast as possible, continuous experimentation instead attempts to validate that it is, in fact, a good idea. In a state of continuous experimentation, the entire R&D process is guided by controlled experiments and feedback. In it's core continuous experimentation consists of a design-execute-analyse loop, where hypotheses are selected based on business goals and strategies, experiments are executed with partial implementations and data collection tools and finally the results are analyzed to validate the hypothesis. In this paper we're ..

Avainsanat — Nyckelord — Keywords
Continuous delivery, Continuous experimentation, Development pipeline

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

It’s hard to argue that Tiger Woods is pretty darn good at what he does. But even he is not perfect. Imagine if he were allowed to hit four balls each time and then choose the shot that worked the best. Scary good. – Michael Egan, Sr. Director, Content Solutions, Yahoo (Egan, 2007)

1 Introduction

-GOAL OF THE THESIS -motivation -research question -approach

>analyze state of the practice >specify problem and goals >analyze state of the art >state hypotheses >derive solution idea

2 Related work

-continuous delivery -continuous experimentation -state of the art practices -short summary

2.1 Continuous delivery

Continuous deployment is an extension to continuous integration, where the software functionality is deployed frequently at customer environment. While continuous integration defines a process where the work is automatically built, tested and frequently integrated to mainline [3], often multiple times a day, continuous deployment adds automated acceptance testing and deployment. The purpose of continuous deployment is that as the deployment process is completely automated, it reduces human error, documents required for the build and increases confidence that the build works [4].

An important part of continuous deployment is the deployment pipeline, which is an automated implementation of an application's build, deploy, test and release process [4]. A deployment pipeline can be loosely defined as a consecutively executed set of validations that a software has to pass such before it can be released. Common components of the deployment pipeline are a version control system and an automated test suite.

In an agile process software release is done in periodic intervals [2]. Compared to waterfall model it introduces multiple releases throughout the development. Continuous deployment, on the other hand, attemps to keep the software ready for release at all times during development process [4]. Instead of stopping the development process and creating a build as in an agile process, the software is continuously deployed to customer environment. This doesn't mean that the development cycles in continuous deployment are shorter, but that the development is done in a way that makes the software always ready for release.

It should also be made clear that continuous delivery differs from continous deployment. Refer to Fig. **??** for a visual representation of differences in continuous integration, delivery and deployment. Both include automated deployment to a staging environment. Continuous deployment includes deployment to a production environment, while in continuous delivery the deployment to a production environment is done manually. The purpose of continuous delivery is to prove that every build is proven deployable [4]. While it necessarily doesn't mean that teams release often, keeping the software in a state where a release can be made instantly is often seen beneficial.

2.2 Experimentation

An experiment is essentially a procedure to confirm the validity of a hypothesis. In software engineering context, experiments attempt to answer questions such as which features are necessary for a product to succeed, what should be done next and which customer opinions should be listened to. According to Jan Bosch, "The faster the organization learns about the customer and the real world operation of the system, the more value it will provide" [1]. Most organizations have many ideas, but the return-on-investment for many may be unclear and the evaluation itself may be expensive [5]. I

In Lean startup methodology [6] experiments consist of Build-Measure-Learn cycles, and are tightly connected to visions and the business strategy. The purpose of a Build-Measure-Learn cycle is to turn ideas into products, measure how customers respond to the product and then to either pivot or persevere the chosen strategy. The cycle starts with forming a hypothesis and building a minimum viable product (MVP) with tools for data collection. Once the MVP has been created, the data is analyzed and measured in order to validate the hypothesis. To persevere with a chosen strategy means that the experiment proved the hypothesis correct, and the full product or feature can is implemented. However, if the experiment proved the hypothesis wrong, the strategy is changed based on the implications of a false hypothesis.

Jan Bosch has widely studied continuous experimentation, or innovation experiment systems, as a basis for development. The primary issue he found is that "experimentation in online software is often limited to optimizing narrow aspects of the front-end of the website through A/B testing and inconnected, software-intensive systems experimentation, if applied at all, is ad-hoc and not systematically applied" [1]. The author realized that for different development stages, different techniques to implement experiments and collect customer feedback exist. Bosch also introduces a case study in which a company, Intuit, adopted continuous experimentation and has increased both the performance of the product and customer satisfaction.

Fig. 1 introduces different stages and scopes for experimentation. For each stage and scope combination, an example technique to collect product performance data is shown. As startups often start new products and older companies instead develop new features, experiments must be applied in the correct context. Bosch states that for a new product deployment, putting a minimal viable product as rapidly as possible in the hands of customers is essential [1]. After the customers can use the product, it is often not yet monetizable but is still of value to the customer. Finally, the product is commercially deployed and collecting feedback is required to direct R&D investments to most valuable features.

2.3 Continuous experimentation

Continuous deployment attempts to deliver an idea to users as fast as possible. Continuous experimentation instead attempts to validate that it is, in fact, a good idea. In continuous experimentation the organisation runs controlled experiments to guide the R&D process. The development cycle in continuous experimentation resembles the build-measure-learn cycle of lean startup [6]. The process in continuos experimentation is to first form a hypothesis based on a business goals and customer "pains" [1]. After the hypothesis has been formed, quantitative metrics to measure the hypothesis must be decided. After this a minimum viable product can be developed and deployed, while collecting the required data. Finally, the data is analyzed to attempt to validate the hypothesis.

As the experiments are run in a regular fashion, it is only natural to attempt to integrate experiments to the deployment pipeline, and to change the development process in such fashion that functionality is developed based on some actual data. The components required to support continuous experimentation include tools to assign users to treatment and control groups, tools for data logging and storing, and analytics tool for conducting statistical analyses.

2.4 State of the art

2.5 State of the practice

-current state at steeri -short summary

3 Hypothesis

Proposal for solution

4 Methodology

Adopt the solution Implement test

5 Results

6 Analysis

7 Conclusion

References

[1] Bosch, Jan: *Building products as innovation experiment systems.* In

Software Business, pages 27–39. Springer, 2012.

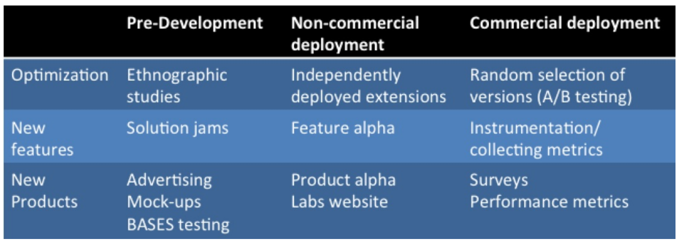
[2] Cockburn, Alistair: *Agile software development*, volume 2006. Addison-Wesley Boston, 2002.

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

[4] Humble, Jez and Farley, David: *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Addison-Wesley Professional, 1st edition, 2010, ISBN 0321601912, 9780321601919.

[5] Kohavi, Ron, Henne, Randal M, and Sommerfield, Dan: *Practical guide to controlled experiments on the web: listen to your customers not to the hippo.* In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 959–967. ACM, 2007.

[6] Ries, Eric: *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses.* Random House LLC, 2011.



	Pre-Development	Non-commercial deployment	Commercial deployment
Optimization	Ethnographic studies	Independently deployed extensions	Random selection of versions (A/B testing)
New features	Solution jams	Feature alpha	Instrumentation/ collecting metrics
New Products	Advertising Mock-ups BASES testing	Product alpha Labs website	Surveys Performance metrics

Figure 1: Scopes for experimentation[1].