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, June 2, 2014

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

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

Oppiaine — Läroämne — Subject

Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master’s thesis	June 2, 2014	13

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

Structure: parts

Background

Literature review

Case study

Structure:

1 Background information - case study context - results - limitations - summary, future work 2 Related work - experimental setup - results - summary

-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 Background and related work

2.1 Case study

Case study is a way to collect data through observation and to test theories in an unmodified setting [10]. An opposite type of research would be a formal experiment, with a narrow focus and control on variables. Case studies are particularly important for industrial evaluation of software engineering methods and tools [6]. Generalizing the results of a case study is often a subject of internal validity [**?**].

2.2 Qualitative research

Qualitative research attempts to answer to questions Why and How instead of What, Where and When as compared to quantitative research. The most common way to collect qualitative data is via an interview, which can be structured, unstructured or semi-structured [**?**].

3 State of the practice

Steeri is a mid-sized company of 80 employees, focusing in managing and improving customer data usage. This includes CRM systems, business intelligence solutions, customer dialog and data integration. Steeri has created Customer data management (CDM) and Customer dialog (Dialog) products, which are developed by two different teams.

These two teams develop in an agile manner, but the continuous integration process [4] and short feedback cycles aren't achieved yet.

The development isn't at all distributed, and no external party affects the development process.

3.1 iSteer Contact backlog tool

3.1.1 Overview

3.1.2 Customer backlog

Work items are first added to customer backlogs in iSteer Contact. In this phase they might be just initial skeletons and do not need to contain all needed information. The work items are added by product or business owners or project managers. At this phase the created backlog items are not yet visible in team backlogs in iSteer Contact. The idea is to list them in the customer backlog as early as possible and start refining the requirements collaboratively both offline and online with the help of tools such as Chatter. Chatter can be used to discuss a single story or the complete backlog.

When the backlog item is ready for the development team to start working on it, it should contain at least the following information: A descriptive name Specifications that explain both the technical side and the business side Agilefant link four hour reporting

The backlog item is added to team backlog by selecting the “show in team backlog” -option.

3.1.3 Team backlog

Team backlog is a view to all stories assigned to one selected team. It is a tool to collect and organize stories from all customer/project backlogs without cloning details into multiple places. It’s just a view of all “in team backlog” stories that are not yet completed and are assigned to the selected team.

Completed stories related to one team can be found from the report (link in team detail page) Team backlog items (stories) are prioritized and estimated once a week (Tuesday) and new stories will be moved to Trello when accepted by the development team.

Stories can be moved to Trello by project managers or product owners but they must be checked by team leader (Juha) who then moves them forward into the sprint backlog or prioritized list columns.

3.2 Using Trello

3.2.1 Overview

Trello is a project management application that uses Kanban to control the production chain from development to release. Kanban attempts to limit the work currently in progress by establishing an upper limit of tasks in the backlog, thus avoiding overloading of the team. Trello consists of multiple boards, each representing a project or a development team. A board consists of a list of columns, and each column consists of cards. Columns each contain a list of tasks, and cards progress from one column to the next when each task has been completed. A card is essentially a task, which is added by the Backlog Owner, and can be checked out by a developer. In Steeri, the columns used are Sprint Backlog, In Progress, Review, Ready, Verified and Done.

3.2.2 Sprint Backlog

The Backlog Owner (Juha) moves the cards that have the highest priority into the sprint backlog. The sprint backlog should always contain enough cards so that whenever a developer completes a task something new is available in the backlog.

3.2.3 In progress

The actual development work is done in this stage. The actor here is the developer. The checklist to move a card into review stage is: · All tasks are complete · Changes are pushed to a feature-specific branch · Unit tests are written and pass in the CI server · Feature is well-written and does not need refactoring · Pull request is created and a link is added to the comment field · Feature has been documented as needed If the feature needs refactoring a task list must be created and the card moved back to In Progress column.

3.2.4 Review

Here other developers review the new code and deploy it to a development environment. Checklist for moving the card into ready stage is: · Pull request is reviewed and by at least two (2) persons · Pull request is merged to the development branch · Feature is deployed to a development environment · Source code quality has to be good enough! After you have reviewed the pull request leave yourself as an assignee. The second person who reviews the pull request is responsible for cleaning all the assignees and moving the feature to Ready column. If there is a major problem in the pull request the feature should be moved back to Sprint Backlog and the yellow "Boomerang" tag added. Person who created the pull request is responsible for implementing the necessary remarks. If only small fixes are needed, they should be implemented within the Github pull request workflow. The second person who has reviewed and accepted the pull request is responsible for deploying the feature to a development environment.

3.2.5 Ready

Here the product owner verifies the new functionality in the development environment. The card can be moved to Verified if: · Feature has been verified by the Product Owner in the development environment Product owner is responsible for moving the feature to the Verified column. If the verification for the feature fails the Product Owner should move the feature back to Sprint Backlog column with the highest priority. In addition the Product Owner should add a yellow “Boomerang” label with a comment describing the results in the feature.

3.2.6 Verified

Here the backlog owner collects the timestamps and trello flow data. The timestamps depicts the duration it took from a card to process through the whole chain. The data is then used to analyze which columns the card spent the longest time in, and to identify the pain spots.

3.2.7 Done

This column simply states that the task has been completed, and should eventually be archived. There’s currently no general validation required from the customer, as the customer projects each have a different schedule and process for builds.

3.2.8 Prioritized lists

The backlog owner adds tasks to prioritized lists from team backlog as soon as the tasks meet the required criterias, contain the required information and are inspected by both the stakeholders and the backlog owner.

3.3 After Trello

Thnergefwdqa

[8]

-current state at steeri -short summary

Be sure to specify as much of the industrial context as possible. In particular, clearly dene the entities, attributes, and measures that are capturing the contextual information.

4 State of the art

4.1 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 [4], 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 [5].

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 [5]. 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 [3]. 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 [5]. 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 [5]. 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.

4.2 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" [2]. Most organizations have many ideas, but the return-on-investment for many may be unclear and the evaluation itself may be expensive [7].

In Lean startup methodology [9] 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" [2]. 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 [2]. 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.

4.3 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 [9]. The process in continuos experimentation is to first form a hypothesis based on a business goals and customer "pains" [2]. 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, integrating experiments to the deployment pipeline should be considered. This requires changing 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.

4.4 Challenges respecting to continuous delivery

-Technical implementation -Education on the subject -Active participation of employees

4.5 Challenges respecting to continuous experimentation

-Participation of every key member

5 Case study implementation: Research method

5.1 Objective - what to achieve?

The study is an exploratory case study, which aims to explore how continuous deployment and continuous experimentation can be integrated to the development process of the company in question. The study specifically aims to identify the main requirements, problems and key success factors with regards to these approaches. Integrating these approaches to the development process requires a deep analysis of the current development process, seeking the current problems and strenghts. Adopting both continuous deployment and continuous experimentaton also requires understanding the requirements of continuous deployment and continuous experimentation.

Existing documented applications of continuous experimentation are primarily executed in the B2C domain, often with a SaaS product. Examples are the Microsoft EXP platform [1] and Etsy []. The focus of this study is in the B2B domain, with an application that is not used as SaaS. As the development process greatly varies in other teams inside the company, the focus is on a single team to narrow the scope.

5.2 The case — what is studied?

Kitchenham et al. (Kitchenham et al., 2002) state that the experimental context needs the three elements: background information, discussion of research hypotheses, and information about related research. The two former will be discussed here, and the latter in the section "Frame of reference".

The company in question is Steeri Oy, which is a medium-sized company specializing in managing, analyzing and improving the usage of customer data. In this research, the unit under the study is the Development Integration team, which is split into two sub-teams: Dialog and CDM.

The Dialog team focuses on developing a multiple-channel online marketing automation, iSteer Dialog. It is a software product designed to allow effective marketing on multiple channels, such as e-mail and websites, and to automate repetitive tasks. The product can be integrated to existing CRM solutions, and the data stored in CRM can then be effectively used for marketing purposes. The software is configured and integrated to the customer environment as project work. This deployment doesn't require additional code as per customer, but only different configuration files.

The CDM team focuses on building a Master Data Management (cite) solution, which integrates the customers data sources such as CRM, ERP and billing system to create a single point of reference. The product also manages the data by removing duplicate records, matching same records and cleaning and validating the data with the help of external data such as the resident registration database. The product is deployed manually to a customer, and certain custom specific configurations and rules have to be implemented in the code.

The unit of analysis is the development process of the team. The whole development process consists of the development framework used, but also of the interaction with customers, tools used in the current development process and the roles of individuals in the process. The unit of analysis is studied by focusing on interviewing individual members at different positions in the organisation.

TODO: describe organisation structure

The organisation of Steeri Oy is of a divisional type, with each business area forming independent teams based on the products and projects. The sub-teams under study have a common team leader, but different product owners and middle management.

Continuous deployment and continuous experimentation haven't been previously used by the case company.

In the beginning of the study, feedback was only collected in the form of bug reports from the customers. The bugs were then prioritized according to the importance, and added into the Trello workflow.

Under analysis is also the company's ways to interact with the customer.

5.3 Theory - frame of reference

TODO: open this chapter

Based on the existing research in Continuous Experimentation and Continuous Deployment:

Crook et al: Seven pitfalls to avoid when running controlled experiments on the web

Eric Ries: The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses

Kohavi et al: Practical guide to controlled experiments on the web: listen to your customers not to the hippo

Fagerholm et al: Building Blocks for Continuous Experimentation

Jan Bosch: Building products as innovation experiment systems

Olsson et al: Climbing the" Stairway to Heaven"–A Multiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software

Humble et al: The deployment production line

Humble et al: Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation

Microsoft EXP platform

The researcher is a part of the development team of the company, but the viewpoint ...

5.4 Research questions - what to know?

Research questions—what to know? these fulfill the objective

What are the problems in the current development process?

What are the strenghts in the current development process?

How is the current interaction with the customer?

subquestion: What are the current roles and stakeholders?

What are the differences in continuous experimentation and continuous delivery in B2B as compared to B2C?

How to collect feedback in a B2B domain?

subquestion: Who should use the collected feedback?

subquestion: Who is responsible for collecting the feedback?

How to collect feedback without a SaaS product?

subquestion: Where should the data be stored?

subquestion: How should the data be accessed?

How to shorten the time between the idea and implementation? subquestion: What are the roles?

How should continuous experimentation guide the development process? subquestion: Where do the development ideas come from? Customer, team, management?

How should continuous delivery be integrated in the development process? subquestion: How to keep the product in a stage where it can be released at all times?

What tools are necessary for continuous deployment?

What tools are necessary for continuous experimentation?

If an impact is done right now, when will the effects be visible?

The whole pipe between the idea and implementation isn't yet completely perceived. Some parts are visible, but the entirety is still unclear. How do we validate whether a feature is succesful or not?

5.5 Methods - how to collect data?

TODO: what, how, why?

The primary source of information in this research are semi-structured interviews performed within the Development Integration team and its management. The interview consists of pre-defined themes as follows: (1) current development process, (2) current interaction with customers, (3) problems and strengths in the current development process, (4) future ways with continuous deployment and continuous delivery. Data triangulation will be implemented by interviewing multiple individuals. Methodological triangulation will be implemented by collecting documentary data regarding the development process.

Data regarding the development process is also collected from the internal documents. Trello, documents

5.6 Selection strategy - where to seek data?

Interview data will be primarily sought from the developers of the development teams and the managers of the team. Process data will be sought from the process documents made by the team. Quantitative data, such as the Trello ticket flow time, is sought from Trello.

6 Needs, problems and challenges

6.1 Problems

TODO: Research problem: "High-level slogan" How to adopt CD and CE in the case company?

Research questions:

Larger context, narrow context Subproblems More concrete (Who is having a problem?)

-Collecting feedback in B2B domain In B2B domain the connection with end users might not be established properly, and the feedback can only be received from the other company's key members. However, the feedback from end users is especially important.

Without the data from end users, no usage data is collected all.

Pilot approach: Prioritize bugs based on the amount of occurrences

-Collecting feedback without a SaaS product In a SaaS platform it is easier to plug in measurement tools to the software, as the environment is that of the company responsible for the service. Without a SaaS environment, the data has to be collected and stored in customer environment.

Thus, the collection of the data takes unnecessary effort, and the whole usage data is totally missing. The product development decisions aren't therefore based on data, and are potentially less effective than decisions based on measurable, quantitative data.

-The time between the idea and implementation is too long Currently the ideas are not tested at all, but the effects are planned in a waterfall style and implemented thoroughly before a release. Making partial implementations could however allow the use of quantitative or qualitative data to measure whether an idea is worth implementing at all.

Management can also better convince other internal stakeholders that a feature might be of importance

-Time dimension. If an impact is done right now, when will the effects be visible? Adding the correct measurement metrics to a feature also make the impact more visible.

-Conventions for production deployment. The whole pipe between the idea and implementation isn't yet completely perceived. Some parts are visible, but the entirety is still unclear. Forming the development pipeline to include both experiments and automated deployment makes the entirety clear from the idea to the release.

-How do we validate whether a feature is succesful or not? In B2C this is usually done by observing shifts in sales, but in our product there's no such thing. The measurement metric has to be chosen such that a decision can be made to judge if the feature is succesful or not.

-How to use the collected data to improve decision making? TODO

-Trello flow time? Occasional deploying takes time. Speeding it up would eventually improve flow time.

-Possible thing we can analyze is user experience, but is it enough?

6.2 goals

Requirements should be testable

-Continuously collect feedback from customers. Subgoal: Define the type of feedback to be collected Subgoal: Define a mechanism for feedback collection Subgoal: figure out how to collect feedback Subgoal: figure who is responsible for analyzing feedback Req: Feedback can be continuously collected from customers Req: The person responsible for collecting feedback can be identified -Reduce the length of the delivery cycle. Subgoal: Identify the components in the delivery cycle Subgoal: Identify what takes too long Req: The delivery cycle is completely perceived Req: The length of the delivery from idea to deployment is shortened -Be able to measure the effectiveness of a feature. Subgoal: Implement measurement tools Subgoal: Implement data analysis tools Subgoal: collect data whenever a feature is deployed to measure the performance Req: Data can be collected from an implemented feature Req: Collected data can be analyzed to form an answer

6.3 solutions

6.4 solution idea

7 Hypothesis

8 Methodology

Adopt the solutison Implement test

9 Results

10 Analysis

11 Conclusion

References

[1] *Experimentation platform*, 2014. `http://www.exp-platform.com/`.

[2] Bosch, Jan: *Building products as innovation experiment systems.* In *Software Business*, pages 27–39. Springer, 2012.

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

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

[5] 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.

[6] Kitchenham, Barbara, Pickard, Lesley, and Pfleeger, Shari Lawrence: *Case studies for method and tool evaluation.* IEEE software, 12(4):52–62, 1995.

[7] 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.

[8] Olsson, Helena Holmström, Alahyari, Hiva, and Bosch, Jan: *Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the*

transition from agile development towards continuous deployment of software. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 392–399. IEEE, 2012.

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

[10] Zelkowitz, Marvin V and Wallace, Dolores R.: *Experimental models for validating technology.* Computer, 31(5):23–31, 1998.

	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[2].