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, July 13, 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	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master’s thesis	July 13, 2014	24

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 ..

ACM Computing Classification System (CCS):
General and reference → Experimentation
TODO: add rest

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

Lean software development, which was inspired by the Toyota Production System [], is a development approach attempting to eliminate unnecessary work and to create value for the customer. The approach consists of seven principles: eliminate waste, build quality in, deliver fast, optimise the whole, create knowledge, defer commitment and respect people.

TODO: Connect this thesis with the Lean Startup Build-Measure-Learn cycle

IEEE divides the quality of a software component into two components: how well the software meets software requirements specifications, and how useful the software is to the end user.

CMMI 5th stage

A way to shorten the delivery cycles and to automate the delivery process is continuous deployment. It is an extension to continuous integration, where the delivery process is often entirely automated, and software functionality is deployed frequently to customer environment. While continuous integration defines a process where the work is automatically built, tested and frequently integrated to mainline [6], often multiple times a day, continuous deployment adds automated acceptance testing and deployment. Continuous deployment therefore attempts to deliver an idea to users as fast as possible by automating the deployment process.

Bosch et al. introduce an innovation experiment system, where the development process consists of frequently deploying new versions, using customers and customer usaged data in the development process and finally by focusing on innovation and testing ideas with customers to drive customer satisfaction and revenue growth [2]. "First, it frequently deploys new versions focusing on continuously evolving the embedded software in short cycles of 2-4 weeks" [4].

Continuous experimentation attempts to validate that an idea 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 [13]. The process in continuous 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.

In this thesis we conduct a case study to analyse the adoption of continuous deployment and continuous experimentation in B2B domain. The case study consists of a literature review on applications of these practices, an interview of key members in the case company and analysis of the results.

The purpose of this thesis is to

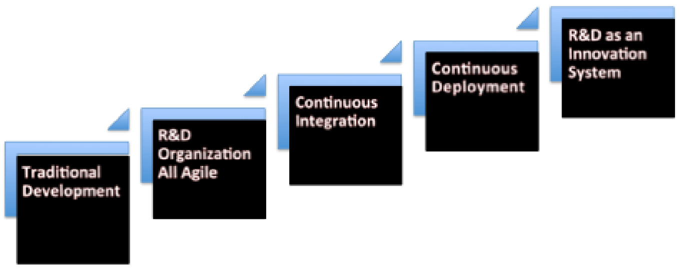


Figure 1: Organization evolution path [12].

The main research questions are as follows.

- **RQ 1: What is the Minimum Viable Process of continuous experimentation? How can the process be simplified without losing the benefits?**
- **RQ 2: What are the differences in continuous experimentation and continuous delivery in B2B as compared to B2C?**
- **RQ 3: Why would continuous experimentation benefit the case company? What are the problems and strengths of the current development process?**
- **RQ 4: How to collect usage data and customer feedback in B2B domain, where user amounts can be very low?**
- **RQ 5: How should continuous experimentation be implemented as a development process?**

The research approach is
This thesis is organized as follows. Chapter II ..

2 Continous delivery and continuous experimentation

Agile software development emerged from the iterative software development methods in the 1980's []. The general principles were

Many agile models, such as Scrum[], Extreme Programming [] and Feature Driven Development [] defined their own ..

Lean manufacturing [], derived from the Toyota Production System [], is a practice that aims to eliminate anything that doesn't create value for the end customer.

Lean startup

TODO: prelude: -Agile software development -Lean manufacturing development -Lean startup -Stairway to heaven -Innovation experiment system

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 [6], 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 [7].

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 [7]. 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. Continuous delivery differs from continuous deployment. Refer

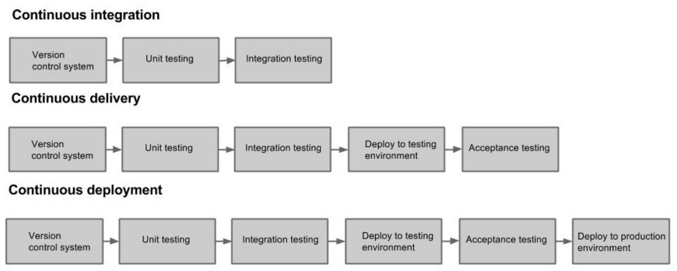


Figure 2: Continuous integration, delivery and deployment.

to Fig. 2 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 [7]. 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.

2.1.1 Deployment pipeline

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 [7]. 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.

Humble and Farley define the deployment pipeline as a set of stages, which cover the path from a committed change to a build [7]. Refer to Fig. 3 for a graphical representation of a basic deployment pipeline. The commit stage compiles the build and runs code analysis, while acceptance stage runs an automated test suite that asserts the build works at both functional and nonfunctional level. From there on, builds to different environments can be deployed either automatically or by a push of a button.

Humble et al. define four principles that should be followed when attempting to automate the deployment process [8]. The first principle states that "Each build stage should deliver working software". As software often consists of different modules with dependencies to other modules, a change to a module could trigger builds of the related modules as well. Humble et al. argue that it is better to keep builds separate so that each discrete module could be built individually. The reason is that triggering other builds can be inefficient, and information can be lost in the process. The information loss is due to the fact that connection between the initial build and later builds is lost, or at least causes a lot of unnecessary work spent in tracing the triggering build.

The second principle states that "Deploy the same artifacts in every environment". This creates a constraint that the configuration files must be kept separate, as different environments often have different configurations. Humble et al. state that a common anti-pattern is to aim for 'ultimate configurability', and instead the simplest configuration system that handles the cases should be implemented. Another principle, which is the main

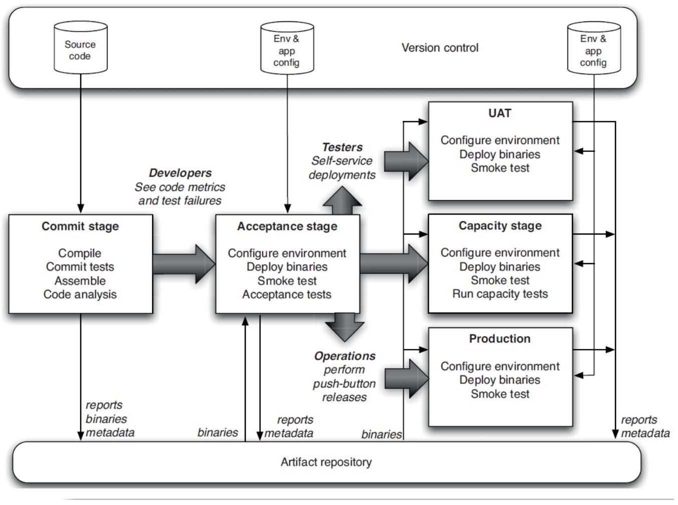


Figure 3: A basic deployment pipeline [7].

element of continuous deployment, is to "Automate testing and deployment". Humble et al. argue that the application testing should be separated out, such that stages are formed out of different types of tests. This means that the process can be aborted if a single stage fails. They also state that all states of deployment should be automated, including deploying binaries, configuring message queues, loading databases and related deployment tasks. Humble et al. mention that it might be necessary to split the application environment into *slices*, where each slice contains a single instance of the application with predetermined set of resources, such as ports and directories. *Slices* make it possible to replicate an application multiple times in an environment, to keep distinct version running simultaniously. Finally, the environment can be smoke tested to test the environments capabilities and status.

The last principle states "Evolve your production line along with the application it assembles". Humble et al. state that attempting to build a full production line before writing any code doesn't deliver any value, so the production line should be built and modified as the application evolves.

2.1.2 Impact on development model

A picture of the typical development process in continuous deployment is shown in Fig. 4. After the team pushes a change to the version control system, the project is automatically built and tests are triggered stage by stage. If a test stage fails, feedback is given and the deployment process effectively cancelled. In a continuous delivery process, the last stages are approved and activated manually, but in a continuous deployment process the last stages are triggered automatically as well.

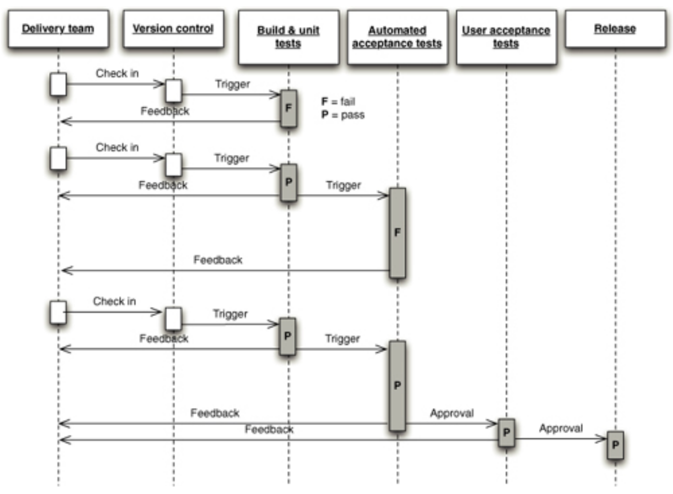


Figure 4: Components of the development process [7].

2.2 Continuous experimentation

-Data driven decisions -Link product development and business aspect -React to customers present needs -Fail fast -Turn hypotheses into facts -Steer development process -systematic! -for a method: what, how, when, who

[4] Applied the continuous experimentation research to embedded software. "Improvement occurs in individual software parts, but the underlying design concept remain mostly unchanged [3]" "The experiment infrastructure allows developers to deploy new software and collect data how it behaves in areal-world settings being used by actual users. The infrastructure support deployment of software experiments and collection of data over-the-air on a scale comparable to the entire customer base". "The infrastructure supports with automated randomisation and factorial designs [6] sufficient to draw statistical conclusions from the experimental scenarios." "The experiment manager architecture, seen in Figure 3, supports the deployment of multiple experimental software parts to the same device and autonomously controls when to run which experiment, even allowing for local A/B-testing. Measurements and analysis is done on-board in real-time. The experiment scenario to be answered is implemented on the embedded device (i.e. how long does it take to . . .)" TODO: picture here

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 [11].

In Lean startup methodology [13] 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.

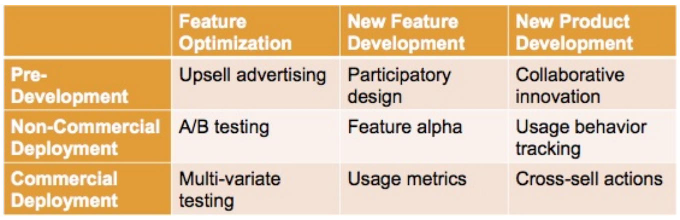
Continuous deployment attempts to deliver an idea to users as fast as possible. Continuous experimentation instead attempts to validate that it is, in fact, a good idea. In continuous experimentation the organisation runs controlled experiments to guide the R&D process. The development cycle in continuous experimentation resembles the build-measure-learn cycle of lean startup. The process in continuous 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.

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.

2.2.1 Experimentation planning

2.2.2 Experimentation stages and scopes

Fig. 6 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.



	Feature Optimization	New Feature Development	New Product Development
Pre-Development	Upsell advertising	Participatory design	Collaborative innovation
Non-Commercial Deployment	A/B testing	Feature alpha	Usage behavior tracking
Commercial Deployment	Multi-variate testing	Usage metrics	Cross-sell actions

Figure 5: Scopes for experimentation[2].

2.2.3 Components in continuous experimentation

Kohavi et al. investigate the practical implementations of controlled experiments on the web [11], and state that the implementation of an experiment involves two components. The first component is a randomization algorithm, which is used to map users to different variants of the product in question. The second component is an assignment method which, based on the output of the randomization algorithm, determines the contents that each user are shown. The observations then need to be collected, aggregated and analyzed to validate a hypothesis. Kohavi et al. also state that most existing data collection systems are not designed for the statistical analyses that are required to correctly analyze the results of a controlled experiment.

The components introduced by Kohavi et al. are aimed primarily for A/B testing on websites. Three ways to implement the assignment methods are shown. The first one is traffic splitting, which directs users to different fleet of servers. An alternative methods is server-side selection, in which API calls invoke the randomization algorithm and branch the logic based on the return value. Last alternative is a client-side selection, in which the front-end system dynamically modifies the page to change the contents. Kohavi et al. state that the client-side selection is easier to implement, but it severely limits the features that may be subject to experimentation. Experiments on back-end are nearly impossible to implement in such manner.

Data collection To collect, aggregate and analyze the observations, raw data has to be recorded. According to Kohavi et al., some raw data could be for example page views, clicks, revenue, render time and customer-feedback selections [11]. The data should also be annotated to an identifier, such that conclusions can be made from it. Kohavi et al. present three different ways for collecting raw data. The first solution is to simply use an existing data collection tool, such as Webmetrics. However, most data collection systems aren't designed for statistical analyses, and the data might have to be manually extracted to an analysis environment. A different approach is local data collection, in which a website records data in a local database or log files. The problem with local data collection is that each additional source of data, such as the back-end, increases the complexity of the data recording infrastructure. The last model is a service-based collection, in which service calls to a logging service are placed in multiple places. This centralizes all observation data, and makes it easy to combine both back-end and front-end logging.

Data analysis To analyze the raw data, it must first be converted into metrics which can then be compared between the variants in a given experiment. An arbitrary amount of statistical tests can then be run on the data with analytics tools in order to determine statistical significance.

2.2.4 Roles in continuous experimentation

Fagerholm et al. define five different roles in continuous experimentation: Business Analyst, Product Owner, Data Analyst, Software Developer and Quality Assurance [5]. The business analyst along with the product owner are responsible for creating and updating the strategic roadmap, which is the basis of the hypotheses. The basis for decisions are the existing experimental plans and results stored in a database. A data analyst is used to analyze the existing experiments and results and to create assumptions from the roadmap. The analyst is also responsible for the design and execution of experiments. The data analyst is in tight collaboration with the software developer and quality assurance, who are reponsible for the development of MVPs and MVFs. The software designers create the necessary instrumentations used to collect the data required by the analyst.

`infra.jpg`

Figure 6: Continuous experimentation infrastructure [5].

2.3 Continuous deployment and continuous experimentation collaboration

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.

2.4 Challenges regarding continuous delivery

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

2.5 Challenges regarding continuous experimentation

-Roles -Environment -Deployment

3 Case study as a research method

3.1 Case study

Case study is a way to collect data through observation and to test theories in an unmodified setting [18]. The case in a case study is the situation, individual, group or organization that the researchers are interested in [14]. Kitchenham et al. found case studies important for industrial evaluation of software engineering methods and tools [9]. Runeson and Höst define it as a suitable research method for software engineering, since it studies contemporary phenomena in its natural context [15]. It might not be possible to generate causal relationships from case studies as compared to controlled experiments. However, they provide a deeper understanding of the unit under study [15]. An opposite type of research would be a formal experiment, with a narrow focus and control on variables.

Case studies can serve different purposes. Robson defines four purposes as Exploratory, Descriptive, Explanatory and Improving [14]. In an exploratory case study, the purpose is to seek new insights, find out what is happening and generate ideas and hypotheses for new research. Descriptive case study attempts to portray a situation or a phenomenom. Explanatory case study attempts to seek and explanation to a problem or situation, mostly in a causal relationship. An improving case study tries to improve an aspect of the studied phenomenom.

Runeson and Höst define the case study structure in five steps [15]. First the case study is designed: objectives are defined and the study is planned. Then procedures and protocols for data collection are defined. Then the evidence is collected by executing the data collection on the studied case. The collected data is then analysed, and finally the results are reported.

Triangulation is a way to increase the reliability and validity of the findings. Triangulation means using different data collection methods or angles, and providing a wider picture of the case. Triangulation is especially important for qualitative data, but can also be used for quantitative data to compensate for measurement or modeling errors [15]. Stake defines four ways to apply triangulation: Data triangulation, Observer triangulation, Methodological triangulation and Theory triangulation [17]. In data triangulation, multiple data sources are used, or the data is collected at different occasions. In observer triangulation more than one observer is used in the study. In methodological triangulation different types of data collection methods are combined, such as qualitative and quantitative methods. In theory triangulation alternative theories or viewpoints are used.

Data collection in case studies tend to lean towards qualitative data that provides a richer and deeper view as compared to quantitative data [15]. As a data collection method, semi-structured interviews are common in case studies [15]. However, generalizing the results of a case study is often a subject of internal validity [10]. It is especially important in a case study to address reliability and validity of the findings. Even with good faith and intention, biased and selective accounts can emerge [14].

3.2 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. Seaman defines interviews as a way to collect historical data, opinions and impressions about something [16]. The interview can be either structured, unstructured or semi-structured. In a structured interview the interviewer asks all of the questions, and the objectives are very specificed. The focus of a structured interview is to find relations between constructs, and the objective is descriptive and explanatory [15]. In an unstructured interview the topic is broadly defined, and questions are asked also by the interviewee. In an unstructured interview open-ended questions are typical, and unforeseen types of information can be gained. The focus of an unstructured interview is on how the interviewee qualitatively experience the phenomenom, and the objective is exploratory [15]. A semi-structure interview is a mixture of both open-ended and specific questions. Semi-structured interviews focus on how individuals qualitatively and quantitatively experience a phenomenom, and the objective is both descriptive and explanatory [15].

Qualitative data analysis [14]

Seaman has explored qualitative research in software engineering context, and he defines different ways to analyse qualitative data. [16] -Coding, extracting values for quantitative variables from qualitative data -Coding subjective data, select reference points for views, evaluate relability -Generation of Theory (GoT), extract from a set of field notes a statement or proposition supported in multiple ways by the data - – for use as hypotheses -GoT cross-case analysis: data divided into “cases”: two groups based on some attribute and examine similarities and differences (devs / management?) -Confirmation of Theory: target: building up “weight of evidence"; triangulation; anomalies in the data -qualitative data is richer than quantitative data. use of qualitative methods increases – amount of information – diversity of information – confidence in results

The basic objective of data analysis is to form conclusions and theories from the data based on a chain of evidence. Qualitative data analysis can be divided into two different parts, hypothesis generating techniques and hypothesis confirmation techniques [15]. Both of these techniques can be exploratory and explanatory case studies [15].

Robson introduces common features of qualitative data analysis in a sequential list [14]. The first step is to code the initial set of data. Then, comments and reflections (referred to as ’memos’) are added. After this the data is processed, similar phrases, patterns, themes, relationships and differences between sub-groups are identified. These identified patterns are then used to focus the next data collection phase. Gradually a set of generalizations are formed, that cover the consistencies discerned in the data. These generalizations are then linked to a formalized body of knowledge in the form of constructs or theories.

In the sequential list, hypotheses are identified after the data has been coded. The hypotheses are then used to guide the following data collection process in an iterative approach. During the iterations, some generalizations can be formed, which eventually form a formalized body of knowledge as a final result.

Hypothesis generating techniques are for example "constant comparison method" and "cross-case analysis" [16]. In the constant-comparison method the field notes are coded, text pieces are grouped into patterns and propositions that are strongly supported by the data are made. The propositions are then validated against any new data. In the cross-case analysis method data is divided into cases, such as two groups based on the same attribute. Pairs of cases are then compared to determine validations and similarities.

4 Current development process of the case company

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 [6] and short feedback cycles aren’t achieved yet.

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

4.1 iSteer Contact backlog tool

4.1.1 Overview

4.1.2 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.

4.1.3 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.

4.2 Using Trello

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.

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.

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.

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.

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.

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. 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. 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.

4.3 After Trello

Thnergefwdqa

[12]

-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.

5 Research study

5.1 Objective

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.

An interview is used to analyze and identify the pain points continuous deployment and continuous experimentation encounter in the B2B domain. After the interview, results are analyzed and addressed... TODO: continue

5.2 Case

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 development process of the team is elaborated in detail in the chapter "State of the practice".

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. The purpose of the interview is to identify the pain points in the development process regarding continuous deployment and continuous experimentation.

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

This case study builds on existing research on both continuous deployment and continuous experimentation. The chapter State of the Art

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

RQ 1: What is the Minimum Viable Process of continuous experimentation? How can the process be simplified without losing the benefits?

Continuous experimentation process with multiple different roles is quite heavyweight, and not quickly adoptable. The purpose of this research question is to attempt to investigate whether only the core process of continuous experimentation can be adopted and used in practice. A way to do this is for example to avoid creating new roles only for the purpose of continuous experimentation. This way it is quick for companies to test whether the process suits their needs. This research question is answered based on literature review.

RQ 2: What are the differences in continuous experimentation and continuous delivery in B2B as compared to B2C?

In B2B domain the connection with end users might not be established properly, and the customer feedback might only be received from certain key members. Running experiments on customer environments also requires a new deployment of the software product. TODO: question: contractual issues? The purpose of this research question is to identify the differences in the software development process and the product in B2B and B2C environments, and it is answered based on both literature review and interview.

Question: selecting the correct OEC in b2b is harder. for example revenue. question: in B2C people develop their own product, but in B2B the feature ideas might come only from customer.
RQ 3: Why would continuous experimentation benefit the case company? What are the problems and strengths of the current development process?
To rationalize the decision to adopt continuous experimentation in a company, the actual benefits to the business have to be identified. This question is answered by comparing problems and benefits of literature review to the results of the interview.

One idea is also to aid marketing by selling the MVPs to customers. This way we can pivot the product with customers, they can test it in practice and then decide whether to pay for the product or not.
RQ 4: How to collect usage data and customer feedback in B2B domain, where user amounts can be very low?
As compared to the B2C domain where the product is often deployed in cloud with a lot of users, B2B applications might only have a handful of users. This question is answered by collecting ideas from the literature review and interview results.
-Legal issues -Technical issues
RQ 5: How should continuous experimentation be implemented as a development process?
When the benefits and requirements are identified and the process is approved by a company, it has to be adopted in practice. This requires an answer to questions what, how, who and when. This question is answered by collecting ideas from the literature review and interview results.
-Where do the feature ideas come from? -What roles are required?

5.5 Methods

The primary source of information in this research are semi-structured interviews performed within the Development Integration team and its management. TODO: business people? The interview consists of pre-defined themes as follows: (1) current development process, (2) current deployment process, (3) current interaction with customers, (4) problems and strengths in the current development process, (5) software product, (6) future ways with continuous deployment and continuous experimentation. Data triangulation will be implemented by interviewing multiple individuals. Methodological triangulation will be implemented by collecting documentary data regarding the development process.

The interview was chosen as a data collection method because of the nature of the research questions. As the case study doesn't include a technical implementation, quantitative measurements to properly measure the effects before and after implementation isn't an option.

5.5.1 Data collection

The interview has a standardized set of open-ended questions TODO: validate. Leading questions are avoided on purpose, and different probing techniques such as "What?"-questions are used. Interviews were performed in the native language of the interviewee if possible, otherwise in English.

The interviews were performed once with every subject. Semi-structured interview with open questions allow deep exploration of studied objects [15]. The interview begins with a set of background questions, used in coding the interview data per subject. After the introductory questions, the main interview questions make up most of the interview. The interview session is structured based on areas under study rather than based on a specific model. The interviews are recorded in audio format, and then transcribed into text.

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

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

5.5.2 Data analysis

In this case study the data analysis is performed with tabulation [15], where coded data is arranged into tables to get an overview of the data. The data is organized by ..TODO.

The data is analysed based on

TODO: Analysis is done [15] Coding Interview results are added to spreadsheet, with the participants grouped based on their teams. TODO: find analysis methods

"One example of a useful technique for analysis is tabulation, where the coded data is arranged in tables, which makes it possible to get an overview of the data. The data can, for example be organized in a table where the rows represent codes of interest and the columns represent interview subjects." [15] TODO: Tabulationia käytetään.

-roles (devs, business, PO's)

5.6 Interview questions

5.6.1 Background

ID	Question	Reason	Notes
1	Name of the interviewee	Background	-
2	Team of the interviewee	Background	-
3	Position of the interviewee	Background	-
4	Years in the company	Background	-
5	Years of experience in the industry	Background	-
6	What is the software product you're working with?	Background	

5.6.2 Current development process

ID	Question
a	Describe your personal daily work routine
a	Describe a normal week with your team
a	What is the current development model in your team?
a	How has the development model evolved during your stay in the company?
a	Where do the development ideas come from? Are they mostly requirements from customers?

5.6.3 Current deployment process

ID	Question	Reason
a	What is the current deployment process like?	Continuous dep
a	Who decides when to deploy?	Continuous dep
a	Who decides what to deploy?	Continuous dep
a	How often is new version released ?	Continuous dep
a	How often are the new versions deployed to customer??	Continuous dep
a	How are the deployment dates chosen?	Continuous dep
a	Which parts of the deployment process are manual?	Continuous dep
a	Which parts of the deployment process are hard to measure automatically?	Continuous dep
a	Is the customer involved in the deployment process?	Continuous dep
a	Does the customer have to do something when a version is deployed?	Continuous dep
a	What are the strenghts in the current deployment process?	Continuous dep
a	What are the problems in the current deployment process?	Continuous dep

5.6.4 Current interaction with the customer

ID	Question	Reason
a	Who is responsible for interacting with the customer?	Genera
a	From your point of view, what are the challenges with the customer interaction?	Genera
a	From your point of view, what are the strengths with the customer interaction?	Genera
a	From your point of view, how could the interaction with the customer be improved?	Genera
a	Is it common for customers requirements to change?	Genera
a	Is the development team aware of the customers present requirements?	Genera
a	How is feedback collected from the customer?	Contin
a	What are the B2B specific challenges in feedback collection?	Contin
a	From your point of view, how could the feedback collection be improved?	Contin
a	How is the customer feedback used?	Contin
a	From your point of view, how could the feedback usage be improved?	Contin

5.6.5 Software product

ID	Question
a	How many end-users does the product have?
a	How is usage data collected?
a	How could usage data collection be improved?
a	What technical challenges would real-time deployment have?
a	What technical challenges would plugged-in data collection instruments have?
a	If data were to be collected in customer environment, what challenges would be faced storing

5.6.6 Problems and strenghts in the current development process

ID	Question	Reason	Notes
a	What are the strenghts in the current development process?	General	RQ 3
a	What are the problems in the current development process?	General	RQ 3

5.6.7 Future ways with continuous deployment and continuous experimentation

ID	Question	Reas
a	Could your product be instrumented with a data collection service? If not, why?	Cont
a	In your product, could experiments be quickly deployed to the customer environment?	Cont
a	In your product, could the development process be guided by A/B testing?	Cont
a	Does your team have a data analyst?	Cont

6 Research results

7 Analysis

(1) current development process, (2) current deployment process, (3) current interaction with customers, (4) problems and strengths in the current development process, (5) software product, (6) future ways with continuous deployment and continuous experimentation.

7.1 Current development process

7.2 Current deployment process

7.3 Current interaction with the customer

7.4 Problems and strenghts in the current development process

7.5 Software product

7.6 Future ways with continuous deployment and continuous experimentation

RQ 1: What is the Minimum Viable Process of continuous experimentation? How can the process be simplified without losing the benefits? Based on the literature review

The interview

RQ 2: What are the differences in continuous experimentation and continuous delivery in B2B as compared to B2C?

The literature review revealed

RQ 3: Why would continuous experimentation benefit the case company? What are the problems and strengths of the current development process?

RQ 4: How to collect usage data and customer feedback in B2B domain, where user amounts can be very low? RQ 5: How should continuous experimentation be implemented as a development process?

The simplest experiment cycle: 1.oec 2. 3.impl 4.analysis 5.steer

8 Conclusions

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] Eklund, Ulrik and Bosch, Jan: *Architecture for large-scale innovation experiment systems.* In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 244–248. IEEE, 2012.

[5] Fagerholm, Fabian, Guinea, Alejandro Sanchez, Mäenpää, Hanna, and Münch, Jürgen: *Building blocks for continuous experimentation.* In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (RCoSE 2014), Hyderabad, India*, 2014.

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

[7] 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.

[8] Humble, Jez, Read, Chris, and North, Dan: *The deployment production line.* In *Agile Conference, 2006*, pages 6–pp. IEEE, 2006.

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

[10] Kitchenham, Barbara A, Pfleeger, Shari Lawrence, Pickard, Lesley M, Jones, Peter W, Hoaglin, David C., El Emam, Khaled, and Rosenberg, Jarrett: *Preliminary guidelines for empirical research in software engineering.* Software Engineering, IEEE Transactions on, 28(8):721–734, 2002.

[11] 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.

[12] 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.

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

[14] Robson, Colin: *Real word research.* Oxford: Blackwell, 2002.

[15] Runeson, Per and Höst, Martin: *Guidelines for conducting and reporting case study research in software engineering.* Empirical software engineering, 14(2):131–164, 2009.

[16] Seaman, Carolyn B.: *Qualitative methods in empirical studies of software engineering.* Software Engineering, IEEE Transactions on, 25(4):557–572, 1999.

[17] Stake, Robert E: *The art of case study research.* Sage, 1995.

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