Information and Software Technology 57 (2015) 21–31

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

On the journey to continuous deployment: Technical and social challenges along the way

Gerry Gerard Claps [a], Richard Berntsson Svensson [b], Aybüke Aurum [c,*]

[a] School of Information Systems, Technology and Management, University of New South Wales, Sydney, Australia
[b] Chalmers | University of Gothenburg, Gothenburg, Sweden
[c] 11 Barry Street, Clovelly, Sydney, NSW 2031, Australia

ARTICLE INFO

Article history:
Received 26 July 2013
Received in revised form 11 May 2014
Accepted 11 July 2014
Available online 23 August 2014

Keywords:
Continuous deployment
Agile software development
Lean software development
Challenges and mitigation strategies

ABSTRACT

Context: Continuous Deployment (CD) is an emerging software development process with organisations such as Facebook, Microsoft, and IBM successfully implementing and using the process. The CD process aims to immediately deploy software to customers as soon as new code is developed, and can result in a number of benefits for organisations, such as: new business opportunities, reduced risk for each release, and prevent development of wasted software. There is little academic literature on the challenges organisations face when adopting the CD process, however there are many anecdotal challenges that organisations have voiced on their online blogs.
Objective: The aim of this research is to examine the challenges faced by organisations when adopting CD as well as the strategies to mitigate these challenges.
Method: An explorative case study technique that involves in-depth interviews with software practitioners in an organisation that has adopted CD was conducted to identify these challenges.
Results: This study found a total of 20 technical and social adoption challenges that organisations may face when adopting the CD process. The results are discussed to gain a deeper understanding of the strategies employed by organisations to mitigate the impacts of these challenges.
Conclusion: While a number of individual technical and social adoption challenges were uncovered by the case study in this research, most challenges were not faced in isolation. The severity of these challenges were reduced by a number of mitigation strategies employed by the case study organisation. It is concluded that organisations need to be well prepared to handle technical and social adoption challenges with their existing expertise, processes and tools before adopting the CD process. For practitioners, knowing how to address the challenges an organisation may face when adopting the CD process provides a level of awareness that they previously may not have had.

1. Introduction

Today, software is developed in fast-changing and unpredictable markets, with complex and changing customer requirements with the added pressure of shorter time-to-market. To address this situation, agile practices have increased the ability for software development companies to handle complex and changing customer requirements, and shifting market needs [23]. However, while agile methods and practices are attractive to many software developing companies, there are few companies (e.g. Facebook [21,40], Atlassian [41], IBM [4], Adobe and Tesla [10], and Microsoft [26] that have succeeded in implementing agile practices to an extent that software is continuously deployed to their customers [42]). At the same time, a number of recent surveys [16,47] have revealed that agile software development practitioners have shown an increasing interest towards understanding the agile software development principle of "Deliver[ing] working software frequently" [7].

To be able to move towards, and implement a Continuous Deployment (CD) process of software, several steps need to be taken [33]. While continuous deployment of software may create new business opportunities, continuous deployment also presents new challenges. A number of practitioners' organisational blogs [4,22,41,48] have expressed the challenges they have faced when adopting and using CD. These challenges include the ease of deploying software bugs, establishing an organisation-wide culture, the need for adopting 'lean' principles, and cross-team collaboration. Moreover, Humble and Farley [27] states that an "intuitive

* Corresponding author.
E-mail addresses: gerryclaps@gmail.com (G.G. Claps), richard@cse.gu.se (R. Berntsson Svensson), aybuke.aurum@gmail.com (A. Aurum).

http://dx.doi.org/10.1016/j.infsof.2014.07.009
0950-5849/© 2014 Elsevier B.V. All rights reserved.

objection to continuous deployment is that it is too risky", and that it is a known fact that more releases lead to lower risk in any release. Hence, continuous deployment reduces the risks for each software release.

This article uses explorative case study research and presents the results of an empirical study that includes data collected through in-depth interviews with ten practitioners from one multinational company in Australia. The study focuses on identifying challenges when adopting the CD process as well as strategies to mitigate these challenges. The study incorporates two main perspectives to challenges when adopting to the CD process, technical and social adoption challenges [46]. Since CD is a software development process that automates the deployment of software [37], there will inherently be technical issues associated with adopting the CD process. Thus, technical challenges were investigated. Both Ries [37] and Fitz [22] have stated that culture is a key success factor towards successfully adopting CD. Furthermore, Arnold [4] states similar issues specifically detailing that more than just developers must work together when using CD, but the entire team used to produce the software product must work together. Thus, this study also examines the social challenges organisations may face when adopting the CD process.

The remainder of this paper is organised as follows: The next section presents the background and related work. Then, we present the research methodology and the limitations of our study, followed by a description of the case company. We then present and discuss the findings. The last section of the paper presents our conclusions and directions for future research.

2. Literature review

While there has not been a large focus on the deployment of software in past literature, the concept of immediately deploying of software is not new [15]. However, only recently has CD been used to deploy software as an extension of the Continuous Integration (CI) process in software development teams [33]. The underlying principles of CD are found in agile software development and lean software development, which is illustrated in Fig. 1.

The term CD was coined by Fitz [22] in 2009. CD is an extension of CI (see Fig. 1), which aims to immediately deploy code to a production server for customers to use [2,37].

2.1. Deployment in agile software development

The current trend for software development methodologies being used in industry is agile software development [18]. Using Dybå and Dingsøyr [18] assessment of the literature on agile software development, it was clear that one agile software development principle proposed by Beck et al. [7] was not thoroughly discussed, which was:

> "Deliver[ing] working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale."
>
> [Beck et al. [7]]

A recent study by De Cesare et al. [16] found that from an organisation's perspective the second most important agile software development principle was to "Achieve customer satisfaction through early and continuous delivery of valuable software". Another recent study by Williams [47] found similar results in that most practitioners agreed that the agile software development principle that resonated with them the most was "to satisfy the customer through early and continuous delivery of valuable software".

While there is an increased demand to understand how to deliver working software frequently from agile software developers [16,47] there has been little research on how to utilise agile methodologies and processes to achieve this. The process of CD aims to fill this gap by allowing software developers to frequently deploy and thus deliver working software faster.

2.2. Lean software development

The Lean Software Development (LSD) is defined as "the application of the principles of the Toyota Product Development System to software development" [34].

There are seven principles proposed by Poppendieck and Poppendieck [34] that form the basis of LSD: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole. Practitioners, such as [31,3,11], have adopted and advocate the use of LSD. However, none of their studies empirically investigate the effectiveness of the principles proposed by Poppendieck and Poppendieck [34]. They merely state that LSD has been beneficial for their software development teams. In spite of this lack of evidence, CD appears to strongly adhere to four of the seven principles of LSD: eliminate waste, amplify learning, deliver as fast as possible, and empower the team.

Lean software development has led to an emerging set of principles commonly applied to existing software development teams and agile methods to help to "deliver [software] as fast as possible" [34].

2.3. Continuous integration

Continuous Integration (CI) is a software development process "where developers frequently integrate their work in a centralized repository. As the work is integrated, an automated build is

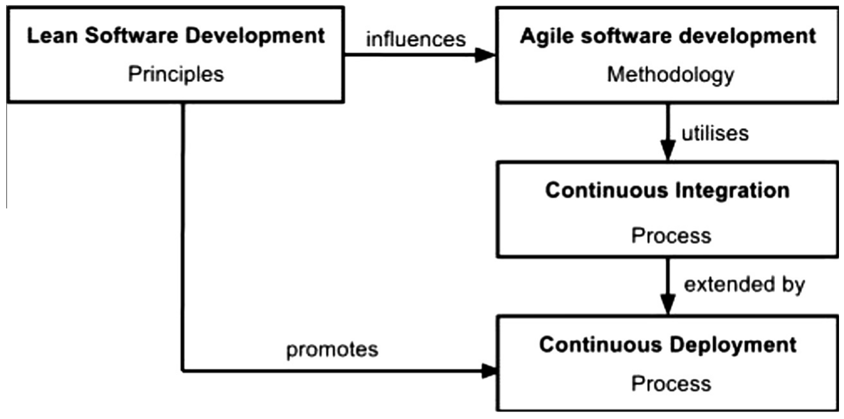

Fig. 1. Conceptual map of the CD process.

executed to discover integration errors as quickly as possible" [17]. Without CI each developer would be working on different versions of the same software product and would not have an integrated software product.

CI originates from the agile software development methodology of Extreme Programming; however, CI has evolved to work with any agile methodology [6]. CI has become such an integral part of developing software that it has been characterised as being a best practice in agile software development [24,44]. CI is fundamental to the success of Continuous Deployment (CD) [33], as CD leverages the CI process and extends CI to include the immediate deployment of software [37]. CI promotes this aim by reducing the barrier to frequent deployment [24]. Practically speaking, CI achieves this by ensuring that software is always in a ready to be deployed state. This is maintained by the CI process which integrates, builds and tests the software [45].

2.4. Continuous deployment

What differentiates CD from traditional software deployment is the frequency of deployment: CD deploys software to production more frequently than traditional software deployment. The key differences between traditional software deployment and CD are summarised in Table 1.

By conducting interviews with a single organisation, Olsson et al. [33] found three main barriers when transitioning from CI to CD: configuring the software at the customer's site; the "internal verification loop" being too long; and a lack of transparency in deployments.

When exploring practitioners' organisational blogs a number of challenges related to the adoption of the CD process were discovered. Wilson [48] described that, even though it seems risky to deploy software frequently to customers, the deployed changes are only ever little.

Practitioners' organisational blogs identify not just the technical challenges encountered with the CD process, but also several social challenges. The company IMVU has successfully implemented CD for a number of years with an average day seeing 50 deployments to production by its developers [37]. The key factor that IMVU attribute to their success with CD is making software deployment a core part of their culture [22]. An article from IBM [4] extends this by stating that one of the largest challenges of CD is not only getting the entire software development team to work together, but also other teams involved in the software product, such as quality assurance personnel. Since software developers can now deploy code, these other team members need to be tightly integrated into the process. Arnold [4] also found the governance of CD to be a challenge; however, Arnold [4] did not provide a clear means of organisations overcoming this challenge other than through their own custom tools. Other challenges identified in the practitioners' blogs include: ease of deploying software bugs; the need for adopting 'lean' principles; and cross-team collaboration. Several of all the identified challenges in practitioners' organisational blogs have yet to be addressed in the academic literature.

2.5. DevOps

DevOps is a set of practices that advocate the collaboration between software developers and IT operations where the aim is to shorten the feedback loop while aligning the goals of both the development and IT operations departments [28]. According to Huttermann [28], DevOps emphasises the empowerment of people over processes, the need of automation in the development of new software, establishing quality measurements and creating a culture of sharing among people.

The widespread adoption of agile methodologies has improved the performance of software developer teams [18]. DevOps widens the spectrum of agile methodologies by not only bringing together programmers, testers and quality insurance engineers, but also the IT operations staff by fostering communication, collaboration and aligning objectives [28]. According to Bass et al. [5], the DevOps community advocates communication between the operations staff and the development staff as a means of ensuring that the developers understand the issues associated with operations. Roche [39] points out that, the influence of DevOps puts efficiency and process into perspectives. One of the main benefits of DevOps is the ability to quantify aspects from the development, i.e. the development process can be described with figures [39]. This in turn has lead to improvement of the product development due to a sharper focus on metrics.

3. Methodology

For this study we aim to understand and to explain the challenges faced by organisations adopting the CD process. Vavpotic and Bajec [46] look at both technical and social challenges from a software development process perspective. According to Vavpotic and Bajec [46], if a software development methodology (SDM) process is socially suitable but does not have the technical expertise to be used, it will be inefficient. On the other hand, if a SDM process is technically suitable, but is not socially suitable, it will be suitable to use but it will not be widely adopted. If the SDM process is both

Table 1
A comparison between traditional software deployment and the CD process.

	Traditional deployment	Continuous deployment
Frequency of deployment	Every 1–6 months [1]. Once when using waterfall or unified process, months when using evolutionary development, and weeks when using agile development [21]	Daily. Companies such as IMVU report deploying software up to 50 times a day using CD [37], and Facebook deploys new code every day using CD [21]
Relative risk of deploying	Higher. Since deployment is an infrequent process, and because the software being deployed contains many changes, there is a higher chance of the deployed software containing bugs [27]	Since changes are small but frequent, it is thought that "Little changes [will only] create little problems" [48]. This is because small, incremental changes reduce the risk (in size) posed by any one change [9]
Customer/ developer feedback loop	Long – dependent on frequency of deployments [1]	Very short, customers constantly receive updates [29,44]
Undeveloped features	Only 69% of required features in software products are developed, leaving approximately 31% of required software features undeveloped (The Standish [25])	CD helps in decreasing development of unnecessary features due to shorter feedback loop. Features are developed constantly so that meaningful feedback can be gained from customers on what features they would like developed [37]. This in turn helps to prevent the development of any wasted software [34]

socially and technically suitable, then the SDM process is useful for the team, and conversely if the SDM process is not socially or technically suitable, it is useless for a team to use. Therefore, it is important to understand both the technical and social challenges when adopting CD. The two research questions that provided the focus for the empirical investigation are:

- *RQ1:* What technical challenges may an organisation face when adopting the continuous deployment process?
- *RQ2:* What social challenges may an organisation face when adopting the continuous deployment process?

Since the purpose of this research is to gain an in-depth understanding of the challenges an organisation may face when adopting the CD process, it is important to study software development teams in practice. A qualitative explorative case study approach was chosen because it allows the researcher to understand the studied phenomenon and its context in more depth [50]. According to Benbasat et al. [8], case studies are appropriate when the objective of the research is the further understanding of a particular phenomenon that has not been investigated fully, as is the case in this study.

When conducting the case study, we followed the five steps process proposed by Yin [50]: (i) *case study design:* objectives are defined and the case study is planned, (ii) *preparation for data collection:* procedures and protocols for data collection are defined, (iii) *collecting evidence:* execution of data collection on the studied case, (iv) analysis of collected data and (v) reporting. Once we obtained permission from the case company for data collection, interviews were organised and conducted with several subjects, and the resulting data was analysed and reported as explained in the following sub-sections.

3.1. Case company

This research was conducted at one case company, Atlassian Software Systems (Atlassian), located in Sydney, Australia. Atlassian was founded in 2002 and currently employs more than 400 employees, and has more than 21,000 customers. Atlassian is an international organisation with offices located in Sydney, San Francisco, Gdansk, Kuala Lumpur and Amsterdam. Generally, a product team in Atlassian develops a single product, but there are some teams that develop multiple products. Atlassian has 13 software products, in four different product categories, namely: project and issue tracking, collaboration and content sharing, distributed version control software solutions, and code quality, that come in two different styles; either hosted by the customer themselves, or hosted by Atlassian. Customer-hosted software is a traditional style of software i.e. the customer pays a licensing fee and downloads the software from Atlassian's website to host on the customer's own servers. In 2011, Atlassian introduced their own-hosted software for their customers called OnDemand. OnDemand is a fully integrated suite of software products, and uses a 'Software as a Service' model where the software is hosted in the "cloud".

Atlassian currently adopts the CD process as a part of its agile software development methodology for 10 of their software products. Atlassian use several different software development methodologies in their product development. Each product team in Atlassian adopts its own software development methodology, which is a combination of agile software development practices, which are predominantly Scrum based but also includes Extreme Programming and Kanban techniques. Consistent with its predominantly agile software development approach, Atlassian deploys a new version of its software to customers using a 98-day release cycle; however, when using CD to deploy software, software is released daily.

The release schedule for Atlassian's 'customer hosted' style of their products are fairly similar, and use a three tiered approach for software releases, which include a major release family, a major release and maintenance releases. These releases can take from weeks to months, with maintenance releases taken significantly less time to develop and release than a major release family or major release. Atlassian's OnDemand products are usually deployed much faster than its customer-hosted products due to the use of a CD process. Due to the frequent releases, there is an absence of versioning beyond the major release of each product. Rather than documenting new features, bug fixes and improvements for each specific version, a weekly update is provided to customers via the company website.

Fig. 2 shows a simplified process of Atlassian's dated CD process. This process lacks the true attributes of CD since there is still a 'deployment button' that is part of the process, which more accurately resembles the continuous delivery process.

3.2. Data collection

Semi-structured interviews were identified as the best data collection methodology for meeting the objectives of this study i.e. to understand the challenges faced by the case company when adopting the CD process for some of their products. The research instrument [14] was formulated with the guidance of a sensitising device, which for this study was a survey devised by Vavpotic and Bajec [46] that focuses on the technical and social challenges faced by organisations when adopting a software development methodology process. Appropriate questions in the Vavpotic and Bajec [46] survey were identified and modified to help create the majority of the interview questions in this study.

The sampling strategy used was a combination of convenience sampling (i.e. practitioners who are available for interviewing) and criterion sampling (i.e. practitioners who are knowledgeable

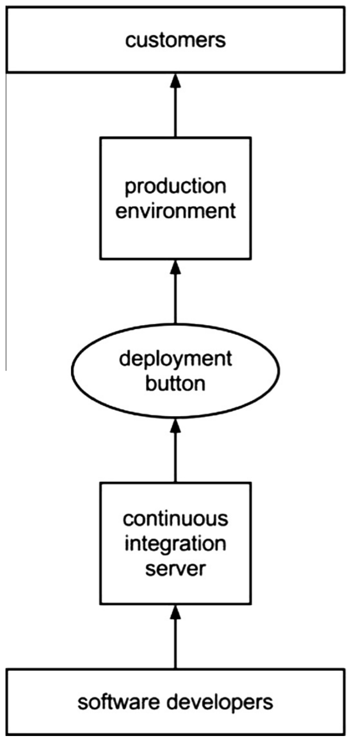

Fig. 2. Atlassian's continuous deployment processes, adapted from Schumacher [41].

on CD approach) [13,32]. The researchers contacted a ‘gate-keeper’ at the company who identified subjects (based on the major roles from Atlassian’s development teams, i.e. developers, technical lead and product managers) that he/she thought were the most suitable and representative of the company to participate in this study. That is, the researchers did not influence the selection of subjects, nor did the researchers have any personal relationship with the subjects. Ten interviews were conducted at Atlassian (see Table 2). Subjects were in the roles of software developer, technical lead and product manager for the products for which the CD process was adopted. Prior to conducting interviews, a 4-phase pilot study was conducted as described by Ellis [19]. In order to facilitate and improve the data analysis process, for all interviews (which varied in length from 20 to 60 min) we took records in the form of audio recordings and then transcribed the interviews using NVivo.

The interviewees were asked to talk about their understanding of, and their views on, the CD process, as well as the technical and social challenges that they faced in CD of products (in their current roles). A number of additional demographic and open-ended questions were added to ensure subjects could disclose all knowledge relevant to the research.

3.3. Data analysis

Data was analysed in this study using the six phase thematic analysis process established by Braun and Clarke [12], which includes: familiarising yourself with your data, generating initial codes, searching for themes, reviewing themes, defining and naming themes, and producing a report.

In the first phase, familiarising yourself with your data, data was transcribed by the first author from the audio recordings obtained during the interviews. Each audio recording was fully transcribed, i.e. word-for-word what the interviewer and interviewee said during each interview. All the transcribed data from the interviews and relevant company documents that was publicly available was imported into a software tool, NVivo (www.qsrinternational.com), for analysing qualitative data.

The second phase, generating initial codes, followed the process of open coding. Open coding begins by systematically reading through the transcripts. Statements from the transcribed interviews were read and coded as challenges by the first author. The Straussian approach was used to analyse the data [49]. Codes were assigned until no further codes were discovered and thus saturation was achieved [20]. Codes originated from either prior literature or from the data. During data analysis, several strategies were adopted. For example, one strategy involved understanding the type of products that were subject to the CD process to achieve an understanding of how the each team member interpreted the CD process in their team. The collection and analysis of data was an iterative and interpretative process.

In the third phase, searching for themes, all the codes that were created in the second phase were organised into higher abstraction level codes, and then assigned to a category by the first author. While a category could have concepts, and concepts could have properties, the reverse was not allowed.

Reviewing themes (fourth phase), once we started comparing transcribed interviews with each other, and identified technical and social challenges, patterns started emerging from the data. After the themes were reviewed, a summary map was created. The interpretation of the results was verified by the third author.

In the fifth phase, defining and naming themes, thematic analysis allowed us to group the technical and social challenges faced, and the mitigation strategies adopted, by the organisation into five concepts: (i) need to be “Lean”, (ii) management driven approach, (iii) changing responsibilities, (iv) risk of adopting CD, and (v) transition from CI to CD. An example of the coding process is given in Fig. 3. In order to further ensure the quality of the coding and the interpretation of the results, the second and third authors checked the results.

In the sixth and final phase, producing the report, the details of these concepts are explained in the following section.

Based on the on-going analysis of the data, we identified weak spots in our understanding of the challenges in CD, and aimed to fill such weak spots with further interviews until we reached a point of theoretical saturation.

4. Results and discussion

In total, 20 (9 technical and 11 social) CD adoption-related challenges were identified. Of the 20 challenges illustrated in Fig. 4, 9 have been uniquely identified by this study; 9 have been previously identified by practitioners in their publications; while only 4 of the challenges have previously been identified in academic literature (Table 3). The most common identified (i.e. how many subjects mentioned the challenge) technical adoption challenges were challenges 1–4 (by 7 or more subjects), followed by challenges 5 and 6 (4–6 subjects) and challenges 7–9 (mentioned by 1–3 subjects) in Fig. 4. Among the social adoption challenges, the most common ones were challenges 10–13 (by 7 or more subjects), followed by challenges 14–16 (4–6 subjects), and challenges 17–20 (mentioned by 1–3 subjects).

Table 3 integrates the technical and social challenges faced, with mitigation strategies adopted by Atlassian when employing the CD process to some of their products. Development of Table 3 is one of the contributions of this research. Although the identified challenges are specific for Atlassian, the identified challenges in Table 3 may be helpful to practitioners that are considering adopting CD process, and for researchers for further investigations of the CD process. The challenges in Table 3 are discussed in the following sections.

4.1. The need to be ‘Lean’

The successful use of CD requires a “lean” mind-set. Ries [37] advocates the use of CD, in conjunction with lean manufacturing principles e.g. those seen in the Toyota Production System. Only

Table 2
Product development teams.

Team	Deployment frequency to internal customers	Deployment frequency to external customers	Uses CD?	Role			Total
				Software developer	Technical lead	Product manager	
A	Daily	Daily	Yes	1	1	0	2
B	Daily	Weekly	Yes	3	0	1	4
C	Daily	Fortnight	Yes	1	0	1	2
D	Ad-hoc	Quarterly	No	1	0	1	2
Total				6	1	3	10

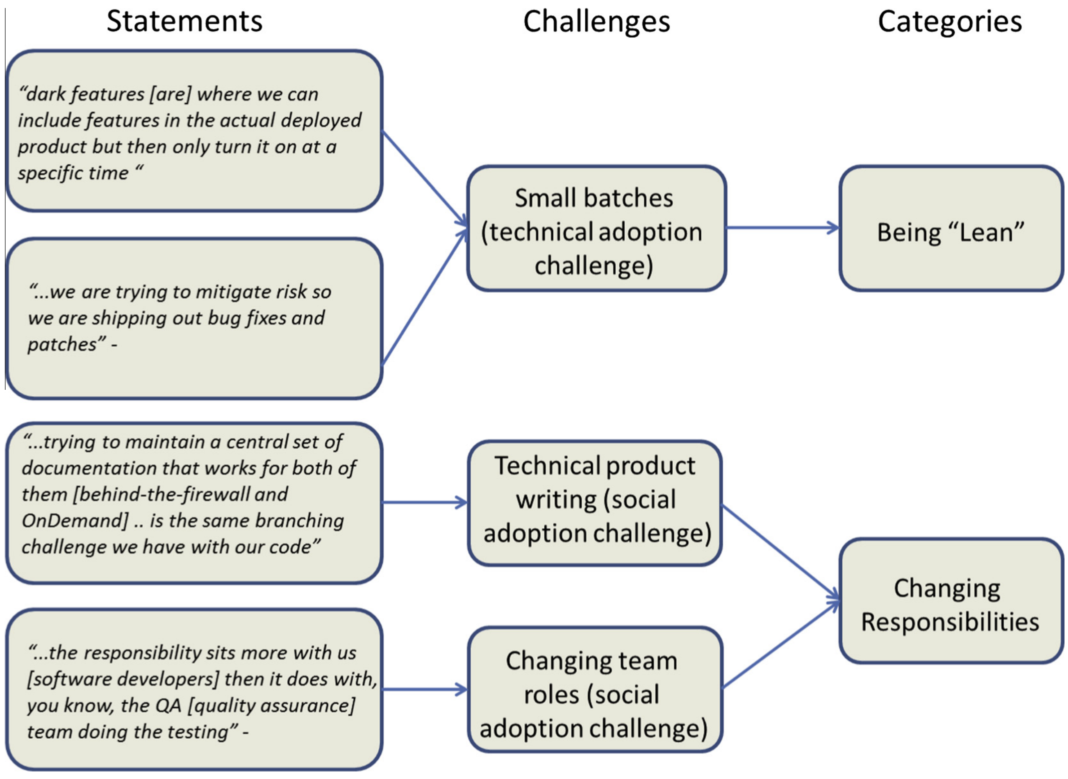
Statements
Challenges
Categories
"dark features [are] where we can include features in the actual deployed product but then only turn it on at a specific time "
"...we are trying to mitigate risk so we are shipping out bug fixes and patches" -
Small batches (technical adoption challenge)
Being "Lean"
"...trying to maintain a central set of documentation that works for both of them [behind-the-firewall and OnDemand] .. is the same branching challenge we have with our code"
"...the responsibility sits more with us [software developers] then it does with, you know, the QA [quality assurance] team doing the testing" -
Technical product writing (social adoption challenge)
Changing team roles (social adoption challenge)
Changing Responsibilities

Fig. 3. Example of coding process.

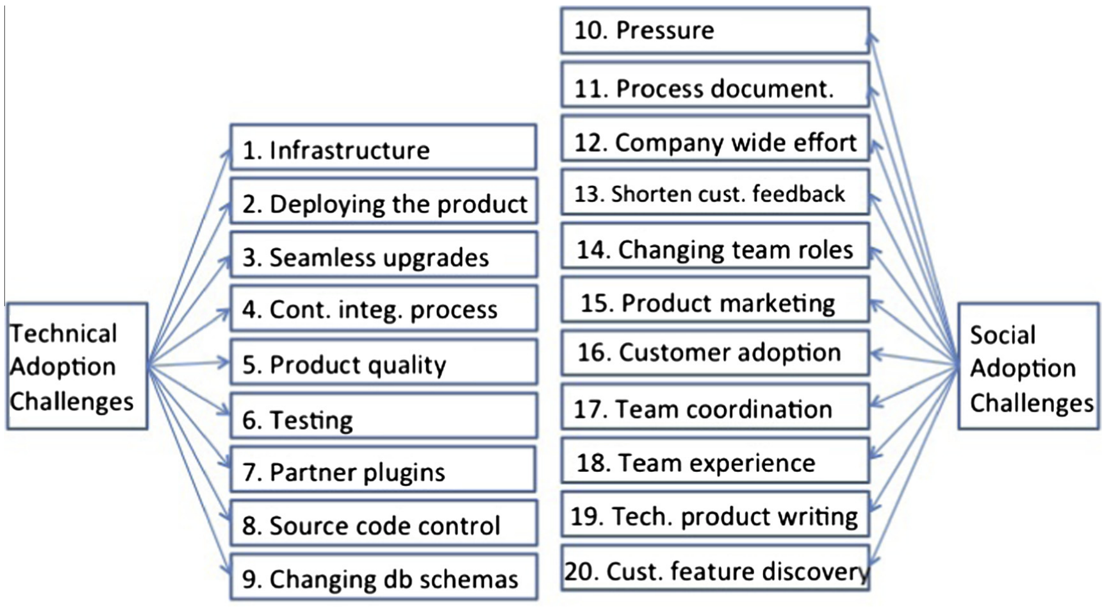

Fig. 4. Technical and social challenges.

recently have lean manufacturing principles been applied to software development to "Deliver [software] as fast as possible" [34]. These 'lean' principles, combined with the CI process, have helped bring CD into existence. The challenges identified in Table 3 highlight a number of key characteristics of CD that are grounded in 'lean' principles (challenges 2 and 13 in Table 3). When these lean characteristics are effectively harnessed, they can reduce waste and shorten the customer feedback loop in software development – a key goal in CD.

For organisations to become 'lean', Ries [37] suggests working in small batches. By breaking down large tasks into small tasks, software (and more importantly, value) can be delivered to customers faster. By continuously deploying small batches of software to customers, organisations can reduce the feedback loop between the organisation and their customers [44].

While the concept of working in small batches is not new, there is no literature to date that has specified the size of the features that can be deployed using CD. The findings of our study have

Table 3
Summary of the challenges faced when adopting CD.

No.	Technical and social challenges	Mitigation strategies	Academic references	Practitioner references
1	*Infrastructure* – requires proper hardware and software to handle the CD process and its related problems e.g. cross-product dependencies	Provide more hardware resources to a product's CI servers to allow the software product to be continuously integrated as often as necessary, thus allowing the software product to be deployed at any time	Olsson et al. [33]	None
2	*Deploying the product* – adopting the practice of small batches	Develop large features (dark features) in small batches that only appear visible to the customer when the entire feature is finally developed	None	Ries [37]
3	*Seamless upgrades* – complications in implementation of seamless upgrades due to resource limitations, zero downtime deployment and customer data preservation	Use two parallel running systems (the old and the new system) to upgrade the user to the next version of software	None	None
4	*Continuous integration process* – challenges with the CI process include merge conflicts, having Atlassian's software in an ever-ready state of deployment, and determining the deployability status of software builds	Employ a strategy of stopping the entire team (i.e. as many people as necessary) from doing what they are doing to fix issues to make sure the software is in an ever-ready state of being deployable	None	Agile Alliance [2], Ries [36]
5	*Product quality* – the quality of the software product may decrease since bugs may slip through and be deployed to customers since deployments occur more frequently	This challenge is outweighed by the benefits of CD. It is mitigated by fixing bugs quickly, and by roll back software to a previous version if any code changes leave the software in a less than fully functioning state	Lacoste [29]	Wilson [48], Schumacher [41]
6	*Testing* – having production quality tests and maintaining the process of 'code review'	Adopt diligent testing, i.e. ensure testing is thorough since deploying software is dependent on tests passing on the continuous integration server	Stolberg [43]	Schumacher [41]
7	*Partner plugins* – only a small fraction of plugins are available due to code integration issues involved with a continuously changing software product	Offer customers a smaller fraction of plugins that are known to be compatible with the Atlassian product	None	None
8	*Source code control* – having one single branch of code to maintain one working version of the software product in a CD environment	Develop each feature for a software product on a separate branch and, when completed, merge that feature into the main branch of code	None	Ries [36]
9	*Changing database schemas* – minor changes in code create unplanned changes in database schema	Ensure there is rigorous testing around the database of a software product to help prevent any deployments affecting the database. Roll backs can also help reset issues	None	None
10	*Pressure* – software developers may feel an increased amount of pressure to have code ready to be deployed immediately	Improve communication among developers and managers	Li et al. [30]	None
11	*Process documentation* – a lack of understanding of the CD process by novice developers due to inconsistent documentation and a lack of industry standards	Adopt 'social rules' which must be adhered to when deploying software	None	None
12	*Company-wide effort* – a lack of motivation for adopting CD	Ensure that top-management implement a strategy to push the need to implement the CD process	None	Ries [37], Arnold [4]
13	*Shorten customer feedback* – being able to analyse data when justifying the deployment of a new feature	Deploy minimal versions of features quickly using experiments, then use the results to update or remove depending on the customers' interactions with the feature	None	Ries [37]
14	*Changing team roles* – team members have to adapt to different tasks in new roles due to working in a CD environment	The team, as a whole, must work closer together to allow the frequent deployment of software to customers without negatively affecting customers	None	Arnold [4]
15	*Product marketing* – marketing CD products requires marketing a versionless product, which requires alternative marketing strategies	Blog a product's changes	None	None
16	*Customer adoption* – Not all customers are pleased to receive updates of features	Apply CD for non-business critical software development	None	Arnold [4]
17	*Team coordination* – requires extra effort to coordinate multiple teams	Engage in an increased amount of planning in terms of planning around deployments with multiple systems that are interdependent	None	None
18	*Team experience* – having an experienced team is critical in the successful adoption of CD	Integrate the automated deployment of software using CD into the existing CI workflow of developers to ensure there is no, or a low learning curve	None	None
19	*Technical product writing* – maintaining multiple documentation for each product which may have multiple offerings (e.g. a customer-hosted version and a versionless Atlassian-hosted online version), where features may be released in one offering, but not in the other	Use Atlassian's own product, Confluence (which is a wiki), to document software products	None	None
20	*Customer feature discovery* – customers may not notice the newly added features	Show the new features in blogs that can be viewed by customers	None	None

shown that the ability to break up large features into small batches, and to ensure that incomplete features do not get sent to customers, are challenges that Atlassian is actively working to overcome (more information about incomplete batches cannot be revealed due to confidentiality reasons). Atlassian develops all their features of their OnDemand products using CD. To break down large features into small batches, Atlassian has adopted the strategy of using 'dark features' when deploying large features. A software developer from Team B described dark features as being, "…*where we can include features in the actual deployed product but then only turn it on at a specific time*". In essence, dark features are features that are too large to develop in a small period of time, and that are instead developed in small batches that only appear visible to the customer when the entire feature has been developed. As each small batch of a large feature is deployed to customers, a switch is turned on to enable the functionality of the feature to come into operation. This means that during the development and deployment of a dark feature, customers do not realise a feature is being developed, nor can they interact with the feature. For Atlassian, it has been essential to understand what features are "switched off" and "switched on", since a feature may otherwise be incorrectly rolled out or not rolled out to customers.

Another challenge faced by Atlassian related to small batches was incomplete changes. If a small batch of code is developed and deployed to customers, but is not fully complete, customers will potentially see the incomplete change and may interact with the software product in undesirable ways. For Atlassian, using small batches to deploy low risk features to customers was the starting point and, as their confidence grew, larger features were deployed using dark features. One product manager explained, "*we are trying to mitigate risk so we are shipping out bug fixes and patches*".

To effectively utilise the shorter feedback loop that results from implementing these lean software development and DevOps techniques, Atlassian has implemented a strategy to monitor customer behaviour through a data analytics platform. By understanding how their customers use their software products, Atlassian can change what software they develop in response to customers' actions, thus eliminating waste and minimising inventory. In addition, Atlassian conducts experiments by developing features and deploying them to certain subsets of customers to test whether these new features are likely to be used by other customers. This is similar to the idea of DevOps, moving beyond the point where quality assurance engineers' insights are coupled with questionable data [39]. That is, the ability to quantify aspects from the development. This is used as a risk mitigation strategy to ensure that software goes out to all customers only if Atlassian believe that all their customers for a particular product should use the new software. This is especially useful if the feature does not end up being well received.

4.2. Management-driven adoption

Both practitioners' literature [4,37] and the results of this study indicate that to successfully adopt the CD process in an organisation, a company wide effort must be made (challenge 12 in Table 3). Olsson et al. [33] identified that adopting the CD process "requires involvement of different organisational units in order to fully succeed". This holds true for Atlassian, which has required an increased amount of collaboration to take place between teams for products that adopt CD (challenge 17 in Table 3). This is the same challenge IBM reported in one of its organisational blog posts when adopting the CD process [4], and it is also emphasised in DevOps. At Atlassian, although the team experience (challenge 18 in Table 3) played an important role in successfully managing CD process, cross-team collaboration could not be achieved without top-management implementing a strategy to push the need to implement the CD process. This made the implementation of the CD process a goal for specific product teams to achieve. A technical lead from Team A described a challenge in adopting the CD process as "*it is something that I would find very hard to do just from a, you know, intrinsic motivation like from within the product team, because it touches so many things*". Another challenge of team coordination when using CD was: who should fix, what, and by when? At Atlassian, teams seem to see the benefits of ensuring their software is always deployable. As described by a software developer from Team A, "*there is a much bigger incentive for the team to keep the build green because it does give us the bonus of having it continuously deployed*".

4.3. Changing responsibilities

A key finding in this study is the changes in roles and responsibilities necessitated across multiple team members that are involved in the CD process (challenges 10, 14, 15 and 19 in Table 3). Changes in roles and responsibilities were found to impact software developers, product managers, product marketers, technical product writers and the support team, as shown in Table 4.

After adopting the CD process, software developers also become responsible for deployments, which meant that the quality assurance team did not overlook quality of the deployed code. This change in responsibility has placed the burden for code quality on the software developers, which in turn has resulted in increased pressure on these developers. Atlassian mitigates this issue by improving communications between developers and managers. The strategy of improving communication between developers and managers is part of the idea of DevOps, to widen the spectrum of agile methodologies by bringing developers, quality assurance engineers, and managers [28].

This result highlights the need for a proper management program for adopting the CD process in a company to ensure that the CD process is not destructive for teams that adopt it.

4.4. The risks of adopting CD

The findings exposed several risks associated to the adoption of CD (challenges 5–9, 11, 16, and 20 in Table 3). Frequently, continuous deployment may not be suitable for all types of software. According to Retting [35] enterprise software is not well suited to this method of development.

For Atlassian, CD is not considered risky because they work in small batches, meaning that bugs are associated with these batches also small. Moreover, since the software is frequently deployed, bugs are found faster, albeit frequently by customers. Although there is a risk in relying on automated testing in the CD process, it is not be possible to deploy software frequently without automated testing. As a consequence, software developers must ensure the tests they generate for their code are of production quality, otherwise, the software deployed to customers may contain bugs. However, it is expected that bugs may still slip through to customers. Atlassian mitigates this problem by rolling back software to a previous version (most versions can be rolled back to after CD was

Table 4
Responsibilities for different roles before and after adopting the CD process.

Role	Before	After
Software developer	Develop software and create tests for that software	Now also responsible for deployments
Product manager	Manage the software product, including its roadmap of features	Experiment features with customers using customer usage information to reduce unnecessary feature development and thus waste
Technical product writer	Write a single set of user documents for a software product	Write two sets of user documentation for a single product to cater for the OnDemand and customer-hosted offerings
Product marketer	Market a software product for a major version release of that product	Market a versionless software product that is constantly changing
Support team	Support a relatively set amount of bugs that are tied to a major software release	Manage an increased frequency of bugs due to the lack of major software releases

implemented) if any code changes leave the software in a less than fully functioning state. Nonetheless, since it is easier to deploy bugs when using CD to deploy software, larger companies are seemingly hesitant to purchase such software. For example, in some cases, large companies have internal policies which state that they cannot use software that is not thoroughly tested.

Another challenge Atlassian has faced when adopting CD is customer feature discovery (challenge 20 in Table 3). CD enables software products to be constantly updated, but it does not assist in introducing these updates to customers. One product manager from Team B found it especially important, "*you know we deploy continuously and like it just goes over their [a customer's] head, so feature discovery is actually, I think, our biggest challenge right now for customers*". Atlassian has found it difficult to show customers when they have deployed new features. Atlassian has found when using CD that not all of their customers welcome this method of development. A software developer from Team B commented that "*…it's affecting them negatively a little bit, but it's only a small subset of customers, because just in general it runs fine*". This has inhibited Atlassian from being able to demonstrate to customers the added value of the ability to introduce new features into their OnDemand software products.

The Atlassian developers also pointed out that there was little or no process documentation associated with the CD process in their teams. However, in most cases they also stated that they did not feel that there were any changes to their original workflows when their teams adopted CD.

The developers described the implementation of CD as both complex and time consuming. Team D found it especially hard to adopt CD because of constantly changing database schemas (see Table 3). Challenges were faced not only with the code itself, but also with the management of the code. An Atlassian technical lead described that a flexible source code repository is needed when adopting the CD process so that frequent deployments can be easily managed. Changing database schemas issues go beyond internal development and extend to partners who provide plugins for Atlassian.

There have been issues with the compatibility of partner plugins which provide additional functionality to some of Atlassian's products. By constantly changing versions and the code of their CD based software, some partner plugins no longer work with Atlassian's latest software updates, and it has become infeasible for partners to keep updating their plugins to work with Atlassian's constant updates.

4.5. The transition from continuous integration to continuous deployment

The findings of this study revealed a number of challenges that Atlassian faced when transitioning from CI to CD (challenges 1–4, 7–9, 11, 17 in Table 3).

A unique challenge for Atlassian has been managing the online suite of products (i.e. the OnDemand products) that are deeply integrated with one another (challenge 1 in Table 3). Customers can pick and choose which products they use from the suite of products. To handle cross-product dependencies, Atlassian has changed its product upgrading process by removing the product that is being upgraded from OnDemand, and then re-integrating the product once it has been upgraded. Due to the complexity of cross-product dependencies, several interviewees believed this was the main challenge for the company when adopting CD. A software developer from Team A explained, "*we had to overcome a lot of obstacles because the OnDemand product is kind of a set of tightly coupled applications*". Moreover, Atlassian has found the scaling of its CI tool with the CD process to be a challenge. The CI process runs multiple times a day when using CD. This has proved to be taxing on the hardware resources that run the CI process. Similar to the adoption of the CI process, the CD process has come with a large upfront set-up time, e.g. managing seamless upgrades (i.e. the ability to deploy software to customers without preventing them from using the software in the meantime). Seamless upgrades are exceptionally complicated to implement, and at scale can be extremely intensive on hardware resources.

The CD process requires software to be in an ever-ready deployable state [27] (challenge 4 in Table 3). To achieve the state of being always ready to deploy, teams must run extremely efficiently by ensuring that any blockages that the software faces are instantly addressed. Ries [37] refers to this behaviour as "Kaizen", a term coined by the Toyota Production System that describes the continual improvement of processes. However, notwithstanding Kaizen behaviour, software that is considered in a state of being deployable should not necessarily be deployed, as it does not mean that the software is free of bugs or errors. It simply means the software has passed a sufficient amount of automated tests that are run on the CI server in the CD process. Atlassian ensures that its OnDemand software is always deployable by immediately stopping the entire team from performing their current responsibilities and redirecting them to work on any issue preventing the software from being deployed.

5. Limitations

For this study, as for any study, there are limitations worth discussing. The threats to description and interpretation validity and the steps taken to mitigate them are reported herein, and the generalizability of the results is discussed. The limitations are described based on guidelines for flexible designs provided by Robson [38].

5.1. Description validity

The main threat to providing a valid description of what has been seen or heard lies in the inaccuracy or incompleteness of the data. This has been mitigated by audio recording all interviews and, later on, transcribing them. Another threat to description validity is the risk of participants not freely expressing their views during the interviews. To mitigate the risk of participants not freely sharing their opinions each participant was guaranteed company internal and external anonymity.

5.2. Interpretation validity

The main threat to providing a valid interpretation is that of imposing a framework or meaning on what is happening, rather than this emerging from what is learnt during the involvement with the setting. This does not preclude starting with a set of predefined categories, but these categories must be subjected to checking of their appropriateness, with possible modification. It requires that the researchers demonstrate how the interpretation of the end results was researched. In this study, the threat of interpretation was managed by the researchers discussing the interviews and how the different researchers interpreted the interviewees' answers. This was accomplished by having multiple researchers interpreted the interviewees' answers, the coding of the statements, and the final results of the data. Furthermore, the transcripts were analysed using the qualitative analysis tool NVivo (www.qsrinternational.com). In the tool, the codes made it possible to trace the route by which we have come to a certain interpretation or conclusion.

5.3. Generalizability

Internal generalizability is concerned with conclusions drawn within the setting studied. That is, the interviewees or observed situations should not be biased by the researcher. This was managed during sampling as the 'gate-keeper' selected interviewees from different teams. Threat to selection bias is always present when subjects are not fully randomly sampled. However, given that four teams with different experience of continuous deployment are included, and interviewees were selected based on their roles by a 'gate-keeper' at the company, this threat has limited effect. In terms of external generalizability, the results of this study are limited to the case company. However, qualitative studies rarely attempt to generalise beyond the actual setting since they are more concerned with characterising, explaining and understanding the phenomena under study. The nature of qualitative designs also makes them impossible to replicate since identical circumstances cannot be recreated. However, the development of a theory can help in understanding other cases and situations. The fact that more than one participant acknowledged several of the discovered results and challenges increase the possibility of transferring the results to other situations. To avoid the interaction of selection and treatment, interviewees were selected by a 'gatekeeper' at the company; hence the researchers did not select the subjects themselves.

6. Conclusion

This research presents the results of an empirical study that examines the adoption of the CD process at a case company. Understanding the intricacies of this complex phenomenon is critical for framing research directions that aim to improve CD software development practices. Through the consolidation of academic and practitioner literature, this study has solidified the current understanding of the challenges faced by organisations adopting the CD process, and has reported a number of additional technical and social challenges e.g. source code control, changing database schemas, changing team roles, team coordination, and customer feature discovery.

While a number of individual technical and social adoption challenges were uncovered by the case study in this research, most challenges were not faced in isolation. The severity of these challenges was reduced by a number of mitigation strategies employed by the case study organisation. For example, to deliver software in small batches, the case company adopted the strategy of using 'dark features'. When a dark feature has been fully deployed, it is activated for customers at a specific time. However, it was discovered in this research that some of the mitigation strategies had led to further challenges themselves. For example, while small batches are used to deploy software multiple times a day using CD, it was difficult to demonstrate to customers the added value of the ability to introduce new features into the software product. Moreover, the use of two parallel running systems leads to other challenges or dependencies between challenges.

The findings of this study have highlighted several risks of adopting the CD process. From a managerial point of view, the findings indicate that organisations need to be well prepared to handle technical and social adoption challenges with their existing expertise, processes and tools before adopting the CD process. Understanding the social and technical issues faced by organisations that have adopted this relatively new software development process will increase both academic and practitioner understanding of the adoption of agile software development processes in organisations generally.

For practitioners, knowing how to address the challenges an organisation may face when adopting the CD process provides a level of awareness that they previously may not have had. The challenges found in this study relating to the adoption of the CD process by organisations may be used as a checklist for practitioners to determine whether their teams, projects or products are suited for transitioning to the CD process. Although the checklist in Table 3 is specific for Atlassian, and cannot be generalised for all organisations, it still can be used as a starting point and can be customised for an organisation's own use. By using the list of challenges found in this study as a checklist, practitioners can examine each challenge in relation to their current situation, and make an informed judgment on whether they should adopt the CD process.

This study has examined the technical and social challenges arising from the adoption of the CD process from a software company perspective. Future work should include an exploration of the challenges faced at the business level, as well as the challenges that customers face in this context. Furthermore, the findings of study can be extended by observing an increased sample of practitioners and companies using CD.

References

[1] P. Agarwal, Continuous SCRUM: agile management of SAAS products, in: Proceedings of the Fourth India Software Engineering Conference, 2011, pp. 51–60.

[2] Agile Alliance, Continuous Deployment. <http://guide.agilealliance.org/guide/cd.html>, 2012 (accessed 25.07.12).

[3] A.L. Alwardt, N. Mikeska, R.J. Pandorf, P.R. Tarpley, A lean approach to designing for software testability, in: Proceedings of the AUTOTESTCON, 2009, pp. 178–183.

[4] S. Arnold, Make Continuous Deployment Practical and Cost-effective with Rational ALM Tools. <http://www.ibm.com/developerworks/rational/library/continuous-deployment-rational-alm/index.html>, 2012 (accessed 24.04.12).

[5] L. Bass, R. Jeffery, H. Wada, I. Weber, L. Zhu, Eliciting operations requirements for applications, in: Proceedings of the First International Workshop on Release Engineering, 2013, pp. 5–8.

[6] K. Beck, C. Andres, Extreme Programming Explained: Embrace Change, Addison-Wesley Professional, 2004.

[7] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, S. Mellor, K. Schwaber, J. Sutherland, D. Thomas, J. Grenning, R.C. Martin, Agile Manifesto. <http://agilemanifesto.org/> (accessed 31.03.12).

[8] I. Benbasat, D.K. Goldstein, M. Mead, The case research strategy in studies of information systems, MIS Quart. (1987) 369–386.

[9] J. Birchler, IMVU's Approach to Integrating Quality Assurance with Continuous Deployment. <http://engineering.imvu.com/2010/04/09/imvus-approach-to-integrating-quality-assurance-with-continuous-deployment/>, 2010 (accessed 21.04.12).

[10] S. Blank, Tesla and Adobe: Why Continuous Deployment may Mean Continuous Customer Disappointment. <http://www.forbes.com/sites/steveblank/2014/01/03/tesla-and-adobe-why-continuous-deployment-may-mean-continuous-customer-disappointment/>, 2014 (accessed 10.04.14).

[11] L. Bocock, A. Martin, There's something about lean: a case study, in: Proceedings of the Agile Conference, 2011, pp. 10–19.

[12] V. Braun, V. Clarke, Using thematic analysis in psychology, Qual. Res. Psychol. 3 (2006) 77–101.

[13] R.B. Burns, Introduction to Research Methods, Pearson Education, 2000.

[14] G. CLaps, Continuous Deployment: An Examination of Technical and Social Adoption Challenges, Honours Thesis, University of New South Wales, Australia, 2012.

[15] T. Coupaye, J. Estublier, Foundations of enterprise software development, in: Proceedings of the Fourth European Software Maintenance and Reengineering (CSMR), IEEE Computer Society, Zurich, 2000, pp. 65–73.

[16] S. De Cesare, M. Lycett, R.D. Macredie, C. Patel, R. Paul, Examining perceptions of agility in software development practice, Commun. ACM 53 (2010) 126–130.

[17] G. De Souza Pereira Moreira, R.P. Mellado, D. Montini, L. Vieira Dias, A. Marques Da Cunha, Software product measurement and analysis in a continuous integration environment, in: Proceedings of the Seventh International Conference on Information Technology: New Generations, 2010, pp. 1177–1182.

[18] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: a systematic review, Inf. Softw. Technol. 50 (2008) 833–859.

[19] L. Ellis, Research Methods in the Social Sciences, Brown & Benchmark Pub., 1994.

[20] D. Ezzy, Qualitative Analysis, Routledge, London, 2002.

[21] D.G. Feitelson, E. Frachtenberg, K.L. Beck, Development and deployment at Facebook, IEEE Internet Comput. 17 (2013) 8–17.
[22] T. Fitz, Continuous Deployment at IMVU: Doing the Impossible Fifty Times a Day. <http://timothyfitz.wordpress.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/>, 2009 (accessed 21.04.12).
[23] N. Fogelstrom, T. Gorschek, M. Svahnberg, P. Olsson, The impact of agile principles on market-driven software product development, J. Softw. Maint. Evol. – Res. Pract. 22 (2010) 53–80.
[24] M. Fowler, Continuous Integration. <http://martinfowler.com/articles/continuousIntegration.html>, 2006 (accessed 21.10.12).
[25] Group TS, Chaos Manifesto – Think Big, Act Smal, 2013.
[26] B. Harry, Announcing Continuous Deployment to Azure with Team Foundation Service. <http://blogs.msdn.com/b/bharry/archive/2012/06/07/announcing-continuous-deployment-to-azure-with-team-foundation-service.aspx>, 2012 (accessed 07.06.12).
[27] J. Humble, D. Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley Professional, 2010.
[28] M. Huttermann, DevOps for Developers, Apress, Berkeley, USA, 2012.
[29] F.J. Lacoste, Killing the gatekeeper: introducing a continuous integration system, in: Proceedings of the Agile Conference, 2009, pp. 387–392.
[30] J. Li, N.B. Moe, T. Dyba, Transition from a plan-driven process to scrum: a longitudinal case study on software quality, in: Proceedings of the Fourth International Symposium on Empirical Software Engineering and Measurement, 2010, pp. 1–10.
[31] P. Middleton, Lean software development: two case studies, Softw. Qual. J. 9 (2001) 241–252.
[32] M.D. Myers, M. Newman, The qualitative interview in IS research: examining the craft, Inf. Organ. 17 (2007) 2–26.
[33] H.H. Olsson, H. Alahyari, J. Bosch, Climbing the "Stairway to Heaven" – a multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software, in: Proceedings of the 38th EUROMICRO Conference on Software Engineering and Advanced Applications, 2012, pp. 392–399.
[34] M. Poppendieck, T. Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, 2003.
[35] C. Retting, The trouble with enterprise software, MIT Sloan Manage. Rev. 49 (2007).
[36] E. Ries, Continuous Deployment in 5 Easy Steps. <http://radar.oreilly.com/2009/03/continuous-deployment-5-eas.html>, 2009 (accessed 21.10.12).
[37] E. Ries, The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, USA, 2011.
[38] C. Robson, Real World Research, Blackwell, Oxford, 2002.
[39] J. Roche, Adopting DevOps practices in quality assurance, Commun. ACM 56 (2013) 38–43.
[40] C. Rossi, Ship Early and Ship Twice as Often. <https://www.facebook.com/notes/facebook-engineering/ship-early-and-ship-twice-as-often/10150985860363920>, 2012 (accessed 13.10.12).
[41] J. Schumacher, Continuous Deployment at Atlassian. <http://blogs.atlassian.com/2011/02/continuous_deployment_at_atlassian/>, 2011 (accessed 21.04.12).
[42] A. Shalloway, G. Beaver, J.R. Trott, Lean-Agile Software Development: Achieving Enterprise Agility, Addison-Wesley Professional, 2009.
[43] S. Stolberg, Enabling agile testing through continuous integration, in: Proceedings of the Agile Conference, 2009, pp. 369–374.
[44] T. Van Der Storm, The sisyphus continuous integration system, in: Proceedings of the Eleventh European Conference on Software Maintenance and Reengineering, 2007, pp. 335–336.
[45] T. Van Der Storm, Backtracking incremental continuous integration, in: Proceedings of the Twelfth European Conference on Software Maintenance and Reengineering, 2008, pp. 233–242.
[46] D. Vavpotic, M. Bajec, An approach for concurrent evaluation of technical and social aspects of software development methodologies, Inf. Softw. Technol. 51 (2009) 528–545.
[47] L. Williams, What agile teams think of agile principles, Commun. ACM 55 (2012) 71–76.
[48] F. Wilson, Continuous Deployment. <http://www.avc.com/a_vc/2011/02/continuous-deployment.html>, 2011 (accessed 21.04.12).
[49] C. Wohlin, A. Aurum, Towards a decision-making structure for selecting a research design in empirical software engineering, Empirical Softw. Eng. J. (2014), http://dx.doi.org/10.1007/s10664-014-9319-7.
[50] R.K. Yin, Case Study Research: Design and Methods, Sage, London, 2003.