Challenges When Adopting Continuous Integration: A Case Study

Adam Debbiche[1], Mikael Dienér[1], and Richard Berntsson Svensson[2]

[1] Department of Computer Science and Engineering, Chalmers, Gothenburg, Sweden
{adam.debbiche,mikael.diener}@gmail.com

[2] Department of Computer Science and Engineering, Chalmers, University of Gothenburg, Gothenburg, Sweden
richard@cse.gu.se

Abstract. The complexity of software development has increased over the last few years. Customers today demand higher quality and more stable software with shorter delivery time. Software companies strive to improve their processes in order to meet theses challenges. Agile practices have been widely praised for the focus they put on customer collaboration and shorter feedback loops. Companies that have well established agile practices have been trying to improve their processes further by adopting continuous integration - the concept where teams integrate their code several times a day. However, adopting continuous integration is not a trivial task. This paper presents a case study in which we, based on interviews at a major Swedish telecommunication services and equipment provider, assess the challenges of continuous integration. The study found 23 adoption challenges that organisations may face when adopting the continuous integration process.

Keywords: continuous integration, software, challenges.

1 Introduction

Software organizations today face a market with ever changing requirements and pressure to release more often. The use of agile practices has increased because of the emphasis they put on customer collaboration and embracing change [14]. Still, companies have been looking into shortening the feedback loop further and releasing more often to the customer by embracing continuous integration [14].

Continuous integration (CI) relates to the frequency at which changes are checked in [19]. Each check in results in a new working build of the software provided that all tests are passed. Continuous integration emphasizes multiple code check-ins on a daily basis as opposed to nightly builds. It is the next step in the evolution of an agile R&D company that has established agile practices in place [14]. It has been suggested that CI increases the frequency of software releases and shortens the feedback cycle [6]. Additionally, Miller [13] state that

CI reduces the time developers spend on checking in new code while maintaining the same level of product quality.

In order to support more frequent integration, requirements need to be small enough in order for developers to test them separately then integrate multiple times a day. This is not always a trivial task in the context of CI [6]. Breaking down large user stories into small enough stories with the right level of detail and visible business and customer value has been identified as a challenge [6].

The purpose of this study is to gain in-depth understanding of challenges with organisational adoption of CI. Understanding the challenges faced by an organisation that have adopted CI will help increase both academic and practitioner understanding of the adoption of CI. This paper achieves its aim through a case study on a single organisation to identify challenges faced by that organisation when adopting the CI process. Data was collected through in-depth interviews with 13 practitioners.

The remainder of this paper is organized as follows. In Section 2, related work is presented, while the research methodology is described in Section 3. Section 4 presents the results, while Section 5 discusses and relates the findings to previous studies, and Section 6 gives a summary of the main conclusions.

2 Related Work

Continuous Integration has its roots in the eXtreme Programming (XP) agile method [11]. The goal for each developer is to commit new code several times a day then build and test the software. A successful test run results in a new build that the team can deploy. A CI system often involves using a CI server that automates the building process [5]. When a developer has added a feature or fixed a bug then the code is tested locally and built before being pushed to a CI server. At this stage, the server merges the new code with the latest version on the server. A new version of the software is built provided that the merge is successful. Otherwise, the developer is directly notified of the conflicts.

Fowler [5] presents a set of benefits related to the implementation of CI. First, the risk of integration errors decreases since integration is no longer a daunting and unpredictable task that is done at the end of each sprint. Second, bugs are discovered and fixed earlier due to the frequency of builds and tests. Bugs are also easier to catch since each small change to the code is checked in and tested separately before being integrated into the mainline [5]. Despite the advantages of adopting CI, several organisations have faced challenges when migrating to CI, e.g. challenges related to moving from agile practices to CI [14]. In [14], CI challenges were identified in relation to problems with handling component dependencies during development and integration. Moreover, automatic testing in the context of CI was also a barrier since it involved testing code running on embedded hardware [14].

The practice and implementation of CI varies in industry [17]. This is due to the fact that the concept is interpreted differently. Sthal and Bosch [17] developed a descriptive model that facilitates the documentation of CI practices and implementations. The model consists of an Integration Flow Anatomy which is essentially a directed acyclic graph (DAG) that shows the activities of a CI process. The graph consists of two types of nodes: input (e.g. source code) and activity (e.g. executing test cases) [17]. While the model has not been fully validated due to the small sample size, it has nevertheless been tested on a real project and the authors were able to identify areas of future improvement for the project and team [17].

Holck and Jorgensen [7] studied the use of a decentralized CI process in the open source community (FreeBSD and Mozilla) where developers are often distributed. The authors found that developers working on open source projects are often free to pick any tasks or bugs they want to work on. Deciding when to integrate changes is also delegated to the developer. This is an advantage when compared to the more plan-driven work assignment process of traditional projects [7]. Breaking down tasks into smaller pieces is also prevalent in open source. However, breaking down large tasks is not trivial. For instance, adding support for Symmetric Multi-Processing (SMP) to the FreeBSD kernel resulted in multiple build errors and severely disrupted the work of other developers [7].

3 Research Methodology

For this study we aim to understand and explain the challenges faced by an organisation when adopting CI. The research question that provided the focus for the empirical investigation is:

– **RQ:** What are the challenges of implementing a continuous integration process in practice?

Since the purpose of this study is to gain an in-depth understanding of the challenges an organisation may face when adopting CI, it is important to study software development teams in practice. The investigation presented in this paper was carried out using a qualitative research approach because it allows the researcher to understand the studied phenomenon and its context in more depth [20]. Due to the potential richness and diversity of data that could be collected, semi- structured interviews [16] would best meet the objectives of this study. Semi-structured interviews help to ensure common information on predetermined areas is collected, but allow the interviewer to probe deeper where required.

This study was conducted as a single case study with an interpretive perspective [9]. That is, to better understand social and organizational contexts where each individual's interpretation is of importance [9]. In addition, this study follows the explanatory-descriptive purpose as classified by Robson [15]. The explanatory-descriptive purpose does not only focus on describing a situation and how things work but also on finding causal relationships between problems and situations.

3.1 Case Company

This research was conducted at one case company located in Sweden. The case company is a world leading provider of telecommunication equipment and services. It offers a wide range of products such as base stations, radio access networks, microwave networks as well as products for television and video. The case company has more than 100.000 employees and offers its services to customers in 180 countries. The products developed by the case company consist of both hardware and software modules.

A number of cross functional teams have adopted CI as part of their development process, albeit with a varying degree of maturity. The teams working on the product use multiple branches for development and integration with different quality assurance policies. Once a team begins working on a new feature or a bug fix, changes are first pushed to their work branch (WB). When a feature is finished then the changes are pushed to the team's Latest Local Version (LLV) where the new functionality is tested. If the LLV tests are successful then the new changes are delivered to the Pre-Test Build (PTB) branch where full regression tests are run. Finally, the new code is integrated with the Latest Stable Version (LSV) branch where the system is tested as a whole.

3.2 Data Collection

Semi-structured interviews [16] were used as the method of collecting data in this study. The research instrument was designed with respect to the different areas of interest and inspiration from [2]. In order to identify relevant subjects for this study, the selection process was carried out with help from a "gatekeeper" at the case company. That is, the researchers did not influence the selection of subjects, nor did the researchers have any personal relationship with the subjects. A total of 13 interviews were performed for this study, eight subjects from Sweden and five subjects from China (see Table 1). Prior to conducting interviews, a pilot interview was conducted to improve the interview instrument (the pilot interview is not part of the collected data). In order to facilitate and improve the data analysis process, for all interviews (which varied in length from 25 to 65 minutes) 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 CI process, as well as challenges that they faced in CI. 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 analysis in this study was done using thematic analysis [1]. It follows a six steps method presented by Braun and Clarke [1]. The first two authors took part in all the steps as described below while the results from the analysis was validated and discussed with the third author:

Table 1. Study subjects

Team	**Number of months using CI**	**Subjects**
T1	18-24	2 developers
T2	2	1 developer
T3	12	1 developer
T4	0	2 developers
T5	2-3	1 developer
T6	12	1 developer
T7	8	1 CI driver
No team	N/A	1 Line Manager, 1 Agile Coach 1 Configuration Manager, 1 Product Owner

Step 1 - Familiarize yourself with the data: The goal of this step is for the researchers to get acquainted with the data collected. For this study it meant translating interview recordings into extensive notes.

Step 2 - Generating initial codes: This phase involves creating the initial codes from the data set [1]. Open coding as described by Robson [15] was used throughout the analysis phase. For this study, the answers in the extensive notes obtained from the interviews were coded into categories. It is important to note that none of the categories were obtained prior to coding as this is highly discouraged when using open coding [15]. The extensive notes were imported into NVivo and then initial codes were generated.

Step 3 - Searching for themes: This step entails classifying the codes initially generated into broader themes. For this study, the initial codes were put into a main category based on the research question. The main category was in turn divided into sub-categories containing sub-themes.

Step 4 - Reviewing themes: In this phase, the themes identified in the preceding step are refined. In this study, the focus was on determining whether the identified categories were appropriate and reflective of the actual data. This was decided based on recurring patterns found in the data and whether they are supported by multiple subjects. New codes were created as well as renamed or removed.

Step 5 - Defining and naming themes: This step requires the scope of the main category to be clearly defined. For this study, all the codes in each category were verified in order to make sure that they were consistent with the overall theme of the category. As part of this step, the final name of each theme was decided before producing the results.

Step 6 - Producing the report: Results of this study are reported in section 4.

3.4 Validity Threats

In this section, threats to validity in relation to the research design and data collection are discussed. We consider the four perspectives of validity and threats as presented in [16].

Construct validity reflects to what extent the research methodology captures studied concepts and what is investigated according to the research questions. The interview protocol was designed based on work done by Claps [2] and tested during one pilot interview. Minor revisions were made based on the feedback from the pilot interview. To reduce bias during the selection of subjects, a "gatekeeper" at the case company was used. That is, the researchers did not influence the selection of subjects. In order to obtain a true image of the subjects' opinion, anonymity was guaranteed, both within the company and externally.

Internal validity refers to the risk of interference in causal relations within the research. Since this study is of empirical nature, incorrect data is a validity threat. In case of the interviews, the recordings and the written extensive notes assured the correct data. In addition, the researchers had the chance to validate the questions and answers with the subjects lessening the chances of misunderstandings.

External validity refers to what extent the findings of the research can be generalised and of interest to other cases. This study included interviews with different roles and teams to benefit the external validity of this research, on the other hand, only performing interviews at one company affects the generalisability of the results negatively. However, the purpose of qualitative studies puts more focus on describing and understanding a contemporary phenomenon and less on the ability to generalize the findings beyond the boundaries of the studied settings. Nevertheless, results from this study may benefit the investigation of phenomena within similar contexts.

Reliability threats relate to what extent the findings of the study are dependent on the researchers that executed the research. This study has been performed by two researchers which increases the reliability and reduces the risk of single researcher bias. In addition, all research design artifacts, findings and each step of the research have been reviewed by the third author.

4 Results

This section answers the RQ, challenges of adopting CI in an organisation. The identified challenges are shown in Figure 1. The most frequent mentioned challenge is Tools & Infrastructure which was mentioned by 13 practitioners, followed by Domain applicability (mentioned by 11 practitioners), and Mindset and Understanding (both were mentioned by 10 practitioners). The challenges in Figure 1 are discussed in the following sections.

4.1 Mindset

The results of this study found that challenges related to the developer mindset play an important role when transitioning to CI.

Scepticism. One of the mindset challenge identified at the case company is being convinced about the benefits that come from adopting CI. Interviewees (7 out of 13) report that they are positive about the concept of CI but need

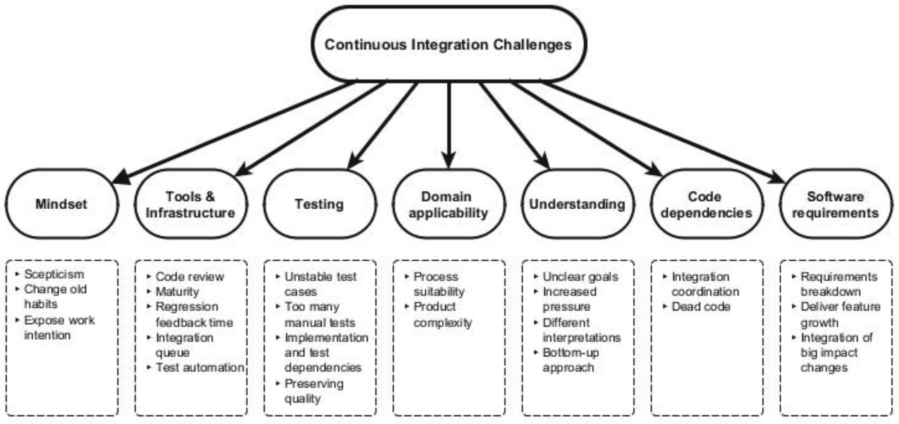

Fig. 1. Continuous integration challenges

to, by themselves, experience the benefits that comes with the change. This to be fully convinced and furthermore promote the adoption. Introducing such a big process change to about 30 teams can be a challenge, as described by a CI driver: "*In the summer of last year, when we started to pilot CI we introduced this change to the team. However, not all of the people are buying the change, not all of them like this change.*"

Some developers that were interviewed are not fully persuaded about the benefits that CI might bring to the product described by agile coach: "*From the very beginning, the team they question about the value for that, how much benefit we can get from the CI because actually, we have already the streamline and we have already monthly delivery and they just want to know what's the benefit, if we want to improve delivery frequency to a weekly delivery. what's the benefit for that.*" Similarly, a CI Driver involved in coaching and helping teams with transitioning to CI highlights the importance of questioning a change.

Change Old Habits. Developers often get used to working in a certain way. The introduction of CI challenged those habit. Consequently, some developers found it hard to give up their old habits. A line manager mentioned that people that have been working at the company for a long time are harder to change compared to junior people because they got used to working in a certain way.

Exposing Work Intention. Continuous integration emphasises early and frequent integrations. As a result, developers are compelled to expose their work earlier. Some interviewees (4 out of 13) found this to be challenge because they were used to big bang integrations at the end of a sprint. This gave them enough time to polish their code before integrating it. With the adoption of CI and the increase in integration frequencies, developers are worried about integrating low quality code that could be questioned by experts and managers, as an agile coach put it, "*some teams they are not familiar or used to frequent delivery, because* *they feel safe if they can deliver once a month because they can make everything ready, if they have some changes, he can correct it on his own branch, don't have to deliver to the main branch and then everybody can see your faults right.*" Developer confidence plays a role in this issue. Teams that are more experienced working with CI seem to be more comfortable about exposing their work earlier.

4.2 Tools and Infrastructure

The tools and infrastructure supporting the CI process are developed and maintained in-house at the case company. These include tools for reviewing code, visualising regression test results, and running automated test suites, checking in code and such.

Code Review: Tools for reviewing integrated code has been reported to lack the necessary features for supporting an efficient CI process. For instance, visualisation of the "bigger picture" while performing a code review has been requested. However, current code review tools only support visualisation of each integrated change separately. This limits the ability to see the impact of smaller changes and how they related to the "bigger picture".

Maturity: The maturity of the tools and their surrounding infrastructure that developers use when integrating code has been reported as a challenge. These tools are often seen as not ready for an efficient CI process. Currently, it takes a long time for developers to integrate their work, especially to the PTB and LSV branches. This in turn prevents them from reaching the desired integration frequency. The build framework and the delivery tools are lacking in terms of maturity and new ways for developers to integrate their codes need to be developed, as one configuration manager puts it, "*after years of developing a product, process and tools tend to be tightly intertwined. When new processes come along then you need to adopt the tools for it.*" This highlights the challenge that the introduction of CI has placed on the tools used at the moment. They need to be better adopted to facilitate CI.

Regression Feedback Time. One common opinion at the case company is that the feedback loops from the automated regression tests are too long. Regression feedback times are reported to take anywhere from four hours to two days. This highlights the problem of getting feedback from regression tests up to two days after integrating code. By then, it could already be out of date. According to one developer, in order to fully derive the efficiency benefits that could come with the change of process, fast feedback loops are important when adopting CI. Nevertheless, long feedback loops, in contrast to "big bang integrations" are still preferable as mentioned by one software developer: "*I think that doing it every day actually is better than doing it in a big bang as we used to do.*"

Integration Queue. The process of managing the integration queue has been difficult due to the nature of the product. There are hundreds of developers working on common and different parts at the same time from different locations. This means that new code is constantly added to the integration queue.

Consequently, two issues emerge. First, keeping track of all the deliveries while preserving quality becomes difficult. Second, the chance of blocking the integration queue increases due to the surge in integrations which can lead to the branch being blocked for several days. These two challenges manifest themselves primarily when integrating to the Pre-Test Branch.

Test Automation. The support of test automation is lacking in the current infrastructure. This makes maintaining the quality of the product difficult according to one developer. There are a lot of manual steps involved in integrating new code. Some system tests are run manually. The ability to automate tests is highly sought after by developers when checking in new code.

4.3 Testing

Challenges associated with testing at the case company are the lack of automated tests along with a stable test framework according to developers.

Unstable Test Cases. Test cases are sometimes unstable (i.e. likely to break or not reflecting the functionality to be tested) and may fail regardless of the code. This makes the evaluation of the results difficult. The varying test coverage between different branch levels, for instance the LLV and the PTB, further complicates the evaluation of the integrated code. Since the PTB covers more tests that are stricter, teams may not be able to guarantee that integrated code will not break the PTB branch, based solely on LLV test results.

Too Many Manual Tests. Automated tests are considered a prerequisite for CI according to an interviewee. The current amount of manual tests is an obstacle to the efforts of adopting CI. Many developers state that there are a lot of tests that need to be run manually before the code can be integrated. This resulted in gaps in the current testing frameworks. The problem is more prevalent on the platform level of the product. This means that teams working closer to the hardware suffer from more manual tests compared to the teams working on the application level.

Implementation and Test Dependencies. A problem related to writing test cases is syncing them with the code they are supposed to test. Often, code is implemented before its test cases are written (or vice versa). This makes it complex to coordinate the integration of new code with its corresponding test cases. Developers do not always know what to do with new code that has no corresponding tests yet or test cases without implemented code.

Preserving Quality. Maintaining the right level of quality while adopting CI has been a concern. According to a CI driver, there has been too much focus on how to introduce CI but not so much on how to retain the quality of the product to be delivered. Additionally, the large amount of people working on the product means that guaranteeing the quality of the increased integrations becomes challenging. Finding an appropriate integration frequency without threatening quality is currently an issue. This is closely related to another identified challenge while adopting CI, more specifically the increased pressure regarding higher integration frequencies. Pushing the integration frequency goal too eagerly could jeopardise the quality of each individual integration.

4.4 Domain Applicability

The challenges related to the suitability of CI at the case company and the related product are presented in this sub-section.

Process Suitability. While taking a step towards more frequent integrations using CI, the case company has been experiencing problems using the same desired frequency throughout all parts of the product. Additionally, some subjects (6 out of 13) are of the opinion that CI is not feature, domain and task independent. Meaning, the possible integration frequency differs depending on what kind of work being carried out. Several software developers highlight the problem of using CI for all parts of the product and features. Another identified challenge related to the suitability of CI is the ability to break down software requirements (described in Section 4.7)

Product Complexity. Product complexity is something that the developers struggle with when transitioning to CI. As a result, many think that CI is difficult to adopt in regards to the product and that it cannot be followed by the book. Compared to smaller products, where all code is merged to a single branch, the development makes use of many branches which adds to the complexity.

Some believe that the complexity is due to how the product is designed: some changes (sometimes minor) require the node to be rebooted which is not appreciated by the customer. The use of quick workarounds rather than fixing the main problem also contributes to the complexity of the product. While the size of the organisation, product and people involved in it has been recognized as a challenge by most interviewees (7 out of 13). Some think that the complexity of the product and the difficulty of transitioning to CI is more related to the confidence of the teams and the tools at their disposal.

4.5 Understanding

The results indicate that teams and management interpret the concepts and objectives of CI differently. These challenges are described below.

Unclear Goals. Lack of setting up clear goals for the teams migrating towards CI is currently an impediment for the teams. Pilot teams are used to explore the possibilities of working with CI. These pilot teams later help other teams migrating. How the teams adopt CI has been up to them. While this freedom is generally welcome, some interviewees (5 out of 13) still believe that the overall goals are unclear. One line manager thinks that there are often some differences in how teams work with CI which can lead to coordination problems and that a more general way ought to be developed. Some developers indicated that they want the organisation to provide clearer instructions on how to proceed with the adoption of CI. Another CI coach believes that more feedback from the pilot teams is needed before any clear goals can be established while emphasising that the aim should be to maximize integration frequencies while enabling teams to set their own pace and goals.

Increased Pressure. The initiative to adopt CI has resulted in increased pressure on the teams according to some interviewees (4 out of 13). Despite the positive support and attitude towards the concept of CI, teams feel that management would like it to happen faster than currently possible which leads to increased pressure. Some developers feel that they lack the confidence and experience to reach desired integration frequencies. There seems to be a general consensus among developers that transitioning to CI carries risks, a period of chaos and increased pressure. Hence, the frequency of integrations and how to proceed should be done in steps in order to minimize the risk of increased pressure.

Bottom-Up Approach. A pilot team was initially established to pilot the CI concept. Currently, the pilot team members act as CI drivers. They provide assistance and training through meetings, workshops and discussions to the other teams. It seems that both management and the teams are mostly happy with the work done by the pilot teams. Most think that they are committed to helping other teams transition to CI. However, the bottom up approach seems to have led some to believe that management could be more involved in the overall process of transitioning to CI.

4.6 Code Dependencies

A consequence of more frequent integrations when adopting CI is that work needs to be divided into smaller pieces. This could mean that work that would otherwise benefit from being developed in one single integration, now might need to be split up into several integrations. Additionally, by dividing work into several integrations, development might be carried out by several developers instead of a single one. This stresses the importance of considering code dependencies and how this affects the integration process.

Integration Coordination. The results from this study indicate that the task of coordinating integration dependencies has been more difficult since the adoption of CI. Consequently, four different issues were reported by developers:

- Component interfaces need to be more clearly defined.
- It is harder to locate the source of errors during integration, because code is delivered from different teams.
- More failures have been experienced during integration.
- Need to wait until other components/parts are done before integrating work.

It has been suggested that a solution to some of the issues presented above could be the use of "dead code", which is described further in the next paragraph.

Dead Code. Integrating partial code for a feature, user story or delivery is currently an issue for the case company. This due to the testability of such integrations. Tests will fail until all parts are in place. A suggestion presented by multiple developers could be to create a test framework that allow integration of so called "dead code". Meaning, code that is activated and tested when all parts are in place. However, making code support this type of activation/deactivation might be more costly, according to one software developer.

4.7 Software Requirements

Software requirements were identified as a challenge when adopting CI. It has increased the frequency at which teams integrate their code. Consequently, requirements that previously could be integrated on sprint basis now need to be broken down to allow more frequent integrations.

Requirements Breakdown. Interviewees report that since the adoption of CI, breaking down requirements to enable more frequent integrations, has been challenging. These challenges are related to finding a balance between size, testability and assuring quality on the integration line. In addition, one developer reports on the lack of experience, constant re-prioritisation and new decisions on requirements, which makes the task of finding a balance even more challenging.

Deliver Feature Growth One of the issues reported by developers with breaking down requirements when using CI is delivering feature growth. It is difficult to know whether small changes that do not directly add value to a feature are worth integrating. Some developers feel this is inevitable. For instance, sometimes you need to re-factor code or make minor changes. These changes do not necessarily contribute growth to the feature itself but are still needed. This means that teams need to be prepared for integrations that do not automatically add feature growth to the customer per say.

5 Discussion

The findings indicate that adopting the mindset aligned with CI is a challenge. Interviewees were skeptical about the benefits that they could gain from adopting CI. Similarly, Claps [2] found that teams adopting continuous deployment need to adopt the mindset needed for it. This means that there needs to be a shift towards a single and united organisational culture that adopts the principles of CI. The case company is transitioning to CI in order to be able to integrate more often and deliver better quality software according to the interviewees. This implies that adopting CI can be seen as introducing a change to an existing software process. Also, bringing change to an existing process in an organisation with the aim of improving software quality is considered a software process improvement (SPI) endeavor [4]. As such, CI is a SPI initiative undertaken by the studied company in a bid to further increase the efficiency of the software development process. Mathiassen et al. [12] argue that improving a process (such as implementing CI) involves changing people. They argue that people do not change simply because processes change. As such resistance to change should be expected. This could explain why some developers express some degree of scepticism towards the adoption of CI. Another reason could be the lack of motivation to change, due to the complexity of adopting a new process, especially if the change is perceived as being too complex [3].

Results show that all of the 13 interviewees mentioned challenges related to tools and infrastructure such as code review, regression feedback time when adopting to CI. The maturity of the tools and infrastructure was found to be a major issue. This has been mentioned in earlier literature. For instance, Olsson et al. [14] identified tools with support for automated tests as a barrier when adopting CI in two companies. Similarly, they argue that developing a fully automated infrastructure remains the key focus when adopting CI [14]. At the case company, the tools and the infrastructure are not entirely ready to accommodate CI. Many developers feel that the current tools available are holding them back. This means that teams are still not able to fully exploit the benefits of CI. That said, a lot of progress is being made such as developing new tools to better support automated tests and the reduction in the amount of branches which is a challenge in itself [14]. The maturity issue is most likely due to the fact that CI is a new concept that is still being adopted. Also, the ongoing responsibility shift means that tools and infrastructure need more time to adapt.

Another issue is the tests themselves. The need for automated tests when transitioning to CI has been recognized in previous research [5], [8]. While organisations realise the importance of automated tests when moving towards CI, they still struggle with it [14]. Improving the testing tools and the infrastructure is not enough according to the interviewees. Olsson et al [14] have also identified automated tests as a key barrier when transitioning from agile to CI. In order to mitigate this problem, two companies facing this issue, have made it their priority to greatly increase the number of automated tests [14] since this is important for CI [5]. Doing this at the case company might not be a trivial task due to the nature of the requirements they are facing. Some are too large and cannot be tested separately. Therefore, writing automated tests might not currently be an option. For instance, one issue mentioned by some of the developers is automating the tests for some features that are part of the Linux kernel used in the product. This issue is similar to the problem encountered by the developers of the FreeBSD project when they were implementing the Symmetric Multi-Processing (SMP) module in the Linux kernel using CI [7]. The ongoing struggle with manual tests could be due to to the complexity of the product itself, which has been developed for well over a decade. Hence, updating the old and large test cases could be a tedious and time consuming task.

Results from this study show that there is an ambiguity regarding the goals and the organisational vision for the implementation of CI. More than half (7 of 13) of the interviewed subjects had a hard time answering questions due to their lack of knowledge of what was expected from them personally. According to Kotter [10], successful cases of process change all share a common denominator, which is divided into 8 steps, liable for their success of change implementation. Kotter [10] highlights the importance of spending the necessary amount of time on each step, and failing to do so will lead to undesired results. For instance, lacking and under-communicating a vision might lead to confusion and incompatible results, which will take the organisation in the wrong direction. This might be a reason to the unclear goals and expectations regarding CI at the case company. Another reason could be the use of a bottom-up approach where directions and guidance come from experience gained in pilot teams. This might have led to the confusion regarding a vision, since the most expected communication channel for organisational visions ought to be management. According to research done by Stahl and Bosch [17] there is no general consensus regarding the practice of CI. Besides the fact that this makes the comparison between different CI practitioners difficult, it could be a reason for the ambiguity at the studied company as well.

Findings from this study show that the suitability of CI is questioned by the majority of subjects interviewed (11 of 13 interviewees). Interviewees mentioned the complexity of the product and the industry in which the case company operates as not being ideal for CI. Research done by Olsson et al [14] describes barriers identified in hardware oriented companies moving towards CI. These companies are used to hardware oriented processes and are struggling with adopting a software oriented process such as CITherefore a shift in culture is needed. Additionally, research done by Turk et al. [18] on the suitability of agile development methods shows that assumptions (e.g. face-to-face communication, quality assurance, changing requirements) made by such methods are not appropriate for all organisations, products and projects. The authors [18] highlight important limitations of said methods, where two is of particular interest for this research, namely limited support for large complex software and large teams. Agile development methods are considered a prerequisite for CI [14], therefore findings by Turk et al. [18], might apply to the applicability of CI. Therefore, the suitability issues of CI raised by interviewees are most likely due to the complexity of the product and t he number of people working on it.

6 Conclusion

This research presents the results of an empirical study that examines the adoption of the CI process at a case company. Understanding the intricacies of this complex phenomenon is critical for framing research directions that aim to improve CI practices. Through a case study, 13 semi-structured interviews were conducted and a set of challenges related to the adoption of CI were identified.

A number of adoption challenges were uncovered by the case study in this research, where the most dominant results include: 1) The mindset is an important factor in the success of implementing CI. Scepticism towards the introduction of a new process needs be considered in order to win over non believers. 2) Testing tools and the maturity of the infrastructure supporting the CI process is required in order to facilitate the daily tasks involved. Continuous integration advocates the use of automated tools to allow more frequent and efficient integrations. 3) Similar to Agile, assumptions made by the concept of CI may not apply to all organisations, products and projects, especially those of larger dimensions. Some of the identified challenges such as mindset, tools and infrastructure maturity and testing have been mentioned in previous literature when transitioning to continuous integration. However, this study also identified software requirements as a challenge when adopting CI.

For practitioners, knowing how to address the challenges an organisation may face when adopting CI provides a level of awareness that they previously may not have had. These challenges can be used as a checklist by companies that are about to adopt CI. The findings of this study can be extended by observing an increased sample of practitioners and companies using CI.

References

1. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative Research in Psychology 3(2), 77–101 (2006)
2. Claps, G.: Continuous Deployment: An Examination of the Technical and Social Adoption Challenges. diploma thesis, The University of New South Wales (2012)
3. Conboy, K., Coyle, S., Wang, X., Pikkarainen, M.: People over process: Key challenges in agile development. IEEE Software 28(4), 48–57 (2011)
4. Deependra, M.: Managing change for software process improvement initiatives: A practical experience-based approach. 4(4), 199–207 (1998)
5. Fowler, M.: Continuous integration @ONLINE (May 2006), `http://martinfowler.com/articles/continuousIntegration.html`
6. Goodman, D., Elbaz, M.: It's not the pants, it's the people in the pants, learnings from the gap agile transformation what worked, how we did it, and what still puzzles us. In: Agile Conference AGILE 2008, pp. 112–115 (August 2008)
7. Holck, J., Jørgensen, N.: Continuous integration and quality assurance: A case study of two open source projects. Australasian J. of Inf. Systems 11(1) (2003)
8. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, 1st edn. Addison-Wesley Professional (2010)
9. Klein, H., Myers, M.: A set of principles for conducting and evaluating interpretive field studies in information systems. MIS Quarterly 23(1), 67–93 (1999)
10. Kotter, J.P.: Leading change: Why transformation efforts fail. Harvard Business Review 85(1), 96 (2007)
11. Lacoste, F.: Killing the gatekeeper: Introducing a continuous integration system. In: Agile Conference, AGILE 2009, pp. 387–392 (2009)
12. Mathiassen, L., Ngwenyama, O., Aaen, I.: Managing change in software process improvement. IEEE Software 22(6), 84–91 (2005)
13. Miller, A.: A hundred days of continuous integration. In: Agile Conference, AGILE 2008, pp. 289–293 (August 2008)
14. Olsson, H., Alahyari, H., Bosch, J.: Climbing the "stairway to heaven"– A multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: 38th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 392–399 (September 2012)
15. Robson, C.: Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Regional Surveys of the World Series. Wiley (2002)

16. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2), 131–164 (2009)
17. Ståhl, D., Bosch, J.: Modeling continuous integration practice differences in industry software development. J. Syst. Softw. 87, 48–59 (2014)
18. Turk, D., France, R., Rumpe, B.: Assumptions underlying agile software-development processes. Journal of Database Management 16(4), 62–87 (2005)
19. Van Der Storm, T.: Backtracking incremental continuous integration. In: 12th European Conference on Software Maintenance and Reengineering, CSMR 2008, pp. 233–242 (2008)
20. Yin, R.K.: Case study research: Design and methods. Sage (1994)