Large scale continuous integration and delivery
Stahl, Daniel

IMPORTANT NOTE: You are advised to consult the publisher’s version (publisher’s PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date: 2017

Link to publication in University of Groningen/UMCG research database

Citation for published version (APA):

Copyright
Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

Take-down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): http://www.rug.nl/research/portal. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.
Chapter 2. Experienced Benefits of Continuous Integration in Industry Software Product Development

This chapter is published as:

Abstract

In this paper, we present a multi-case study of industrial experiences of continuous integration among software professionals working in large scale development projects. In literature, multiple benefits of continuous integration are suggested, but case studies validating these benefits are lacking. This study investigates the extent to which continuous integration effects – increased developer productivity, increased project predictability, improved communication and enabling agile testing – suggested in literature are experienced in industry development projects. The study involves four independent products at different levels of continuous integration maturity within Ericsson AB. In each of these products developers, testers, project managers and line managers have been interviewed. Their experiences of continuous integration are quantitatively assessed and discussed in comparison to the continuous integration benefits proposed in related work.

2.1 Introduction

Continuous integration was popularized in the late '90s as part of eXtreme Programming [Beck 2000]. It is a software development practice where changes are integrated early and often. There is a wide spectrum of proposed beneficial effects of this practice in related work, such as enabling agile testing, communicating development status within the team or increasing developer productivity, while others suggest an increase in project predictability. In other words, there isn't one homogenous understanding of what the exact consequences of introducing continuous integration in software development are, and case studies confirming claimed effects are lacking. Additionally, there may be contextual differences between development projects that impact the extent to which potential effects of continuous integration manifest, which are not yet fully understood.

This paper formulates, based on findings in related work, a series of hypotheses of what the benefits of continuous integration are. Then, through a quantitative case study of industry development projects, the validity of each hypothesis is examined. The contribution of this paper is first that it uses an industrial multi-case study of large-scale software development to validate two hypotheses related to continuous integration: it improves communication both within and between teams and it improves project predictability. Second, it questions another hypothesis: continuous integration supports agile testing; the empirical data of this study does not allow us to fully validate this. Third, it is validated that continuous integration increases developer productivity. In this case, however, only one of the two reasons for this increase suggested in related work is supported: while
the effect of continuous integration facilitating parallel development is validated, the claim that it provides a significant reduction of compilation and testing overhead prior to checking in changes is not.

The remainder of this paper is organized as follows. In the next section, the research method used for the study is described. In Section 2.3 the hypotheses based on claims made by related work are formulated. The collection of the case study data is discussed in Section 2.4. In Section 2.5 the data and the hypotheses are examined, and the paper is then concluded in Section 2.6.

### 2.2 Research Method

The research was conducted by reviewing existing articles on continuous integration. Based on these articles hypotheses as to the effects of adopting continuous integration were formulated. Data was then gathered through interviews with industry software professionals, and the data was investigated in order to validate or disprove the hypotheses.

#### 2.2.1 Study of Related Work

The first steps of systematic literature review were used to find articles that make explicit claims as to benefits of continuous integration. This resulted in a set of seven articles (see Section 2.3), some of which were in agreement, but several of which proposed continuous integration effects not mentioned in the other articles. This made it clear that there are differing expectations on continuous integration as a practice, and indeed differing experiences of it. To capture and investigate the benefits proposed in the selected articles, hypotheses describing the effects of continuous integration in software development projects were then formulated (see Section 2.3).

#### 2.2.2 Interviews

The most appropriate method to examine the validity of the formulated hypotheses, we concluded, was by investigating the extent to which software professionals recognize the proposed benefits in their work. In order to compile a representative data set we decided to interview professionals both in different development projects and working in different disciplines. Therefore four products where chosen (see Section 2.4.1), and in each of those products interviews with developers, testers, project managers and line managers were conducted. The interviews were semi-structured, with questions prepared to cover effects proposed in related work, but encouraging interviewees to discuss and elaborate on their answers. In some cases additional questions were added to the interview guide as a result of such discussions: in cases where the interviewee didn't necessarily disagree with an effect, but suggested other causes than those stated in the hypothesis we felt that it was important to capture this by including it in subsequent interviews.

For each effect the interviewees were asked whether they had experienced it from the continuous integration implementation currently in place in their product development. The interviewees were also asked to quantify their answers by assigning them scores representing the extent to which they perceived continuous integration contributing to those effects. These scores range from 0 (not at all), to 7 (very much).

In total, 22 individuals spread across the products included in the study and representing the roles described above were interviewed.

22
2.2.3 Investigation of Data

The answers provided by the interviewees were collated, upon which average scores and standard deviation of scores were calculated for each continuous integration effect as well as for the entire data set.

The average scores assigned by the interviewees were used to determine whether their perceptions and experiences support the formulated hypotheses, and the standard deviations were used to determine to what extent there was consensus among them.

Some interviewees refrained from answering certain questions. This was due to one of two reasons: either the question was not understood, or they did not consider themselves to have the experience or be in a position where they could tell whether a particular continuous integration effect was manifest. Regardless of the reason, such missing answers were omitted from the data set and consequently not included in the calculation of score averages and standard deviations.

2.3 Hypotheses

To find articles based on which hypotheses concerning the effects of continuous integration could be formulated the first steps of systematic literature review were used. The IEEE Xplore database was searched for publications with the terms "continuous", "integration" and "software" either in their titles or abstracts, which returned 361 matches.

From reading the abstracts of these papers we determined that a large number of these 361 articles, however, did not deal with the software practice of continuous integration, but rather with subjects such as signal processing and artificial intelligence. From those dealing with the software practice of continuous integration, a smaller set of 33 articles were found to potentially make concrete and tangible claims as to benefits of continuous integration. These articles were reviewed in full, and the hypotheses below were formulated to represent explicitly proposed benefits of continuous integration effects found in seven of them [Boehm 2005, Downs 2010, Goodman 2008, Lacoste 2009, Liu 2009, Miller 2008, Stolberg 2009].

- **Hypothesis 1:** Continuous integration supports the agile testing practices of automated customer acceptance tests and writing unit tests in conjunction with production code. In [Stolberg 2009], continuous integration is considered essential in supporting agile testing. Agile testing is in this context considered to involve practices such as customer defined acceptance tests, automating said tests and running them in the regression test suite at least daily, as well as developing unit tests for all new code during a sprint (an iterative sprint based process was used) and then running those unit tests with every build.

- **Hypothesis 2:** Continuous integration contributes to improved communication both within and between teams. The problems of effectively communicating within a development team are investigated in [Downs 2010], which states that the continuous integration constituted a significant part in how the team communicated and consequently had a significant effect on the work flow. However, while [Downs 2010] investigated a single team, we consider it to be equally important to cover large scale development projects where more than one team is involved. Thus the hypothesis is phrased in such a way as to capture both intra-team and inter-team communication.
• **Hypothesis 3:** Continuous integration contributes to increased developer productivity as an effect of facilitating parallel development in the same source context and reduced compiling and testing locally before checking in. It is reported from one project switching to continuous integration [Lacoste 2009] that it allowed them to easily maintain more than one line of development, thereby increasing the flow of changes. Additionally, [Mille 2008] calculates the net gains of adopting continuous integration by measuring time saved by the developers not compiling and testing locally before checking in versus the time spent on the continuous integration framework itself. This implies that the primary benefit of continuous integration is as a time saver for developers. The third hypothesis was phrased to cover both of these proposed benefits.

• **Hypothesis 4:** Continuous integration improves project predictability as an effect of finding problems earlier. [Goodman 2008] claims that with continuous integration, the company studied in the article is now able to release frequently and predictably. [Liu 2009] finds that continuous integration testing is an effective method of discovering bugs continuously, and [Boehm 2005] supports this view, claiming that continuous integration is helpful in finding problems earlier rather than later.

### 2.4 Data Collection

This section describes the data collection process.

#### 2.4.1 Studied Products

This section describes the four products included in the case study. All of these are Ericsson products, due to the ease with which we were able to obtain access to them, but they are all developed by independent organizations within the company. From the many dozens of development projects in Ericsson these four were selected to ensure a good distribution and representativeness of software development projects at large. We wanted to capture projects with very short experience of continuous integration, as well as those with relatively longer experience. We also wanted to represent products where the continuous integration was explicitly broken into stages, with later stages focused on the integration of prebuilt binary components, similar to the approach described in [Roberts 2004], as well as products where this is not the case. In other words, we wanted to include one product representing each quadrant of the diagram shown in Figure 3.

![Figure 3: The four quadrants of software development projects represented in the case study.](image-url)
It shall be noted that all of the studied products were, at the time of the study, being actively developed by multiple teams.

### 2.4.1.1 Product A

Product A is a network node, developed by tens of cross-functional development teams. Each team delivers into the product mainline every few months, on average. In other words, new changes are integrated into the product mainline, with the product consequently being rebuilt and retested several times a week. At the time of the case study, the development organization of product A was in the planning stage of their continuous integration implementation and was chosen to represent products with short experience and lesser focus on binary components (see Figure 3).

### 2.4.1.2 Product B

Product B is actually a portfolio of products that interacts with a wide array of network nodes. Unlike the other products in the case study, which are considering or have adopted continuous integration at a mature stage in the product life cycle, product B was designed for large scale continuous integration from the outset, using a binary integration based approach similar to that described by [Roberts 2004]. At the time of the case study the portfolio was still in its first year of development. This product was chosen to represent products integrating binary components, but with short experience of continuous integration (see Figure 3). Measured in number of teams, product B was at the time of writing the smallest product in the case study, with less than ten cross-functional development teams.

### 2.4.1.3 Product C

The software delivered by product C is developed by tens of cross functional teams and integrated into a large number of hardware and software configurations, placing strict requirements on variability and extensive verification. Each team pushes into the product mainline approximately once a week, on average, using a time slot reservation system. In this regard, the continuous integration of product C can be described as a quicker variant of the process previously in place in product A. This illustrates the elasticity of how the term continuous integration is sometimes used in large scale software development.

Having a similar approach as product A to building the product, but longer experience, product C was chosen to represent the upper left quadrant of Figure 3.

### 2.4.1.4. Product D

Product D is a network node, developed by tens of cross-functional development teams. Several years prior to the case study the development organization switched from a previous integration approach, with late integration of large changes, to continuous integration. Their continuous integration setup compiles changed modules in the first stage and then, via binary integration, builds and verifies the full node.

For these reasons, product D was chosen to represent product development with a longer experience of continuous integration as well as a greater focus on binary components.
2.4.2 Interview Guide

The interview guide consisted of eleven questions, designed to address the four hypotheses (see Section 2.3).

2.4.2.1 Supporting Agile Testing

The first hypothesis states that continuous integration supports the agile testing practices of automated customer acceptance tests and writing unit tests in conjunction with production code. To examine this hypothesis, two questions were included in the interview guide:

1. To what extent have you experienced continuous integration supporting agile testing, in the sense of automated customer acceptance tests?
2. To what extent have you experienced continuous integration supporting agile testing, in the sense of writing unit tests in conjunction with new production code?

2.4.2.2 Improving Communication

The second hypothesis states that continuous integration contributes to improved communication both within and between teams. This hypothesis was addressed by the following questions:

3. To what extent have you experienced continuous integration contributing to improved intra-team communication?
4. To what extent have you experienced continuous integration contributing to improved inter-team communication?

2.4.2.3 Increasing Developer Productivity

The following questions were designed to address the hypothesis that continuous integration contributes to increased developer productivity as an effect of facilitating parallel development in the same source context and reduced compiling and testing locally before checking in:

5. To what extent have you experienced continuous integration improving developer productivity, as an effect of less local compiling and testing before checking in?
6. To what extent have you experienced continuous integration facilitating parallel development in the same source context?

In addition to these questions, it was suggested during the study that there may be other causes of potentially increased developer productivity, and so the following questions were also included in the interview guide:

7. To what extent have you experienced continuous integration improving developer productivity as an effect of easier re-basing and merging?
8. To what extent have you experienced continuous integration improving developer productivity as an effect of more effective troubleshooting?

2.4.2.4 Improving Project Predictability

The final hypothesis is that continuous integration improves project predictability as an effect of finding problems earlier. The following question was included to examine this:
9. To what extent have you experienced continuous integration improving project predictability, as an effect of finding problems early?

While related work is largely focused on describing unit tests and (functional) acceptance tests in relation to continuous integration, we were also curious about whether early non-functional system testing is improving predictability in the industry:

10. To what extent have you experienced continuous integration improving project predictability, as an effect of early non-functional system testing?

Furthermore, it was suggested during the study that predictability also increases because integration is performed outside of the project's critical path. The reasoning here is that small, early and incremental integrations can be done in parallel with development, while traditional "big bang" integrations towards the end of the project inevitably take place on the critical path. We considered it to be worth dedicating an extra question to this:

11. To what extent have you experienced continuous integration improving project predictability, as an effect of integration taking place outside of the critical path?

2.4.3 Interviewees

To ensure a sufficient distribution of interviewees in each of the projects developing the studied products, managers of these products were asked to suggest interviewees representing developers, testers, project managers and line managers. One to two representatives of each role in each product were interviewed.

2.5 Hypotheses and Data Examination

This section examines the data gathered in the study and discusses it in relation to the formulated hypotheses.

2.5.1 Examination of Data

A total of eleven questions pertaining to the formulated hypotheses are included in the data set. Across all eleven questions, the average standard deviation of scores assigned by interviewees (on a scale of 0 to 7, see Section 2.2.2) to their experiences of continuous integration effects was 2.30. We believe that this high standard deviation reveals a large amount of disagreement. Even though the exact causes of this disagreement are currently not understood, we nevertheless find this to be an interesting result in itself: it supports the view that perceptions and experiences of continuous integration differ (see Section 2.2.1).

In order to determine whether a hypothesis was validated by the empirical data, the average score for the questions pertaining to that hypothesis was used: an average of 3.5 or above is considered a validation. For hypotheses with multiple stipulated causes, all causes must be validated for the hypothesis itself to be fully validated. Also note that some questions (e.g. questions 10 and 11) are not designed to validate any hypotheses, but merely to provide additional data. Furthermore, we do not consider a score below 3.5 to necessarily rule out the validity of an effect, but instead may be an indication of a correlation that is not yet fully understood.
2.5.2 Examination of Hypotheses

This section discusses each hypothesis in turn and presents the results of the questions pertaining to those hypotheses. The scores given by the interviewees in response to these questions are depicted in Figures 2 through 5. In each figure, the average score is displayed as a horizontal bar. Also, the distribution of responses is represented as a candlestick chart [Morris 1992], with the vertical bar representing the minimum and maximum scores, while the lower and upper boundaries of the box represent the first and third quartiles respectively.

2.5.2.1 Supporting Agile Testing

The first hypothesis is that continuous integration supports the agile testing practices of automated customer acceptance tests and writing unit tests in conjunction with production code, which is addressed by questions 1 and 2 (see Section 2.4.2.1). The scores are displayed in Figure 4.

Question 1 (support of agile testing in the sense of automated customer acceptance tests) received an average score of 2.82, with a standard deviation 1.90.

Question 2 (support of agile testing in the sense of unit tests written in conjunction with production code) received an average score of 3.77, with a standard deviation of 2.33.

The support for the hypothesis from this result is ambiguous: while the interviewees perceive an effect on supporting unit tests written in conjunction with production code, their experience of the customer acceptance test effect is significantly weaker. Some of the interviewees suggested that this is because they don't have any direct interaction with end customers – rather their "customers" tend to be another department or internal testers – and that they didn't make use of this particular testing process.

Even though scores of question 2 are significantly higher, it was argued several times during the study that this should not be considered so much an effect of continuous integration, as a prerequisite.
In conclusion, we find that our study can not fully validate this hypothesis, although it is partly supported. It does appear clear, however, that there is a correlation between agile testing practices and continuous integration (and a point could arguably be made that continuous integration itself is one such practice), but the exact nature of this correlation is not fully understood. In particular, it remains unclear what is cause and what is effect, and what the contextual prerequisites of successful interaction between continuous integration and customer acceptance tests are. It is also unclear to what extent the particular circumstances of the studied products affect the ability of these benefits to manifest: as hinted at by the interviewees themselves, it is possible that in a different context the support for automated customer acceptance tests would be more pronounced.

### 2.5.2.2 Improving Communication

The second hypothesis states that continuous integration contributes to improved communication both within and between teams, which is addressed by questions 3 and 4 (see Section 4.2.2). The scores are displayed in Figure 5.

![Average scores assigned by interviewees in response to questions 3 and 4 (see Section 2.4.2.2). The lower and upper bounds of the boxes represent the first and third quartiles, respectively.](image)

**Figure 5:** Average scores assigned by interviewees in response to questions 3 and 4 (see Section 2.4.2.2). The lower and upper bounds of the boxes represent the first and third quartiles, respectively.

Question 3 (improving intra-team communication) received an average score of 3.77, with a standard deviation of 2.83.

Question 4 (improving inter-team communication) received an average score of 3.85, with a standard deviation of 2.44.

The data gathered in the case study validates the hypothesis, both on account of intra-team and inter-team communication. It should be noted, however, that the average scores for these questions are not exceedingly high, while the standard deviation is above average. But this does not tell the full story: looking at the distribution in Figure 5 it becomes evident that experiences, particularly in response to question 3, are very polarized. Indeed, elaborating on these questions, interviewees gave diverging accounts of how product build and quality status was communicated in their respective products. Therefore we believe that further investigation into how differences in continuous integration implementations affect these potential benefits is warranted.
2.5.2.3 Increasing Developer Productivity

The third hypothesis is that continuous integration contributes to increased developer productivity as an effect of facilitating parallel development in the same source context and less compiling and testing locally before checking in, which is addressed by questions 5 and 6. In addition, questions 7 and 8 address the same effect, but suggest different causes (see Section 4.2.3). The scores are displayed in Figure 6.

![Box plot showing average scores for questions 5 to 8 regarding increased developer productivity](image)

Figure 6: Average scores assigned by interviewees in response to questions 5, 6, 7 and 8 (see Section 2.4.2.3). The lower and upper bounds of the boxes represent the first and third quartiles, respectively.

Question 5 (increased productivity as an effect of less local compiling and testing before checking in) received an average score of 2.08, with a standard deviation of 2.79.

Question 6 (increased productivity as an effect of facilitating parallel development) received an average score of 4.91, with a standard deviation of 2.02.

Question 7 (increased productivity as an effect of easier re-basing and merging) received an average score of 3.46, with a standard deviation of 2.95.

Question 8 (increased productivity as an effect of more effective troubleshooting) received an average score of 2.92, with a standard deviation of 2.18.

There are striking differences in how interviewees rate the contribution to developer productivity, depending on which cause the question addresses. Increased productivity as an effect of easier re-basing and more effective troubleshooting receive moderate support, while the time saving aspect (question 5) received very low scores. All of these effects also had medium to high standard deviation, indicating differing opinions. Figure 4 clearly shows that questions 5 and 7, especially, received polarized responses which warrant further investigation.
In contrast, in the experience of the interviewed software professionals in this study, there is a significant effect in facilitating parallel development in the same source context (question 6). Here the standard deviation is also relatively low, with first and third quartiles close together, indicating a higher degree of consensus. It is worth noting, however, that even so there are individuals who do not perceive this effect at all.

In conclusion, we find that the case study partly supports the hypothesis, while also indicating that there may be other (albeit lesser) causes for increased productivity, such as more effective troubleshooting and easier rebasing and merging.

It should be noted that in addition to the low scores assigned to the time saving effect, as suggested by [Miller 2008], one interviewee went as far as to claim that in fact the exact opposite is true: as a consequence of adopting continuous integration, developers become more careful in verifying their changes before checking them in, as it is considered of utmost importance to not break the build needlessly. Indeed, this point of view also has support in literature [Humble 2010]. However, this does not necessarily rule out the validity of the effect itself: there may be differences in context that determine to what extent certain benefits can manifest. For instance, all the development projects of this study are significantly larger than that studied in [Miller 2008], containing many times the number of development teams.

It is conceivable that such environmental factors can influence whether it's beneficial for developers to skip local compiling and testing before checking in. The high standard deviation can be interpreted as supporting this assumption – there is simply significant disagreement among the interviewees as to whether this is a benefit of continuous integration or not – but currently it is not understood what those factors are or what their influence is.

### 2.5.2.4 Improving Project Predictability

The fourth hypothesis is that continuous integration improves project predictability as an effect of finding problems earlier, which is addressed by question 9. Additionally, questions 10 and 11 are also about improved project predictability, but suggest different reasons for it (see Section 2.4.2.4). The scores are displayed in Figure 7.

Question 9 (improving predictability as an effect of finding problems early) received an average score of 4.77, with a standard deviation of 1.53.

Question 10 (improving predictability as an effect of early non-functional system testing) received an average score of 3.42, with a standard deviation of 1.98.

Question 11 (improving predictability as an effect of performing integration outside of the critical path) received an average score of 3.67, with a standard deviation of 2.36.

The gathered data clearly supports the hypothesis that continuous integration improves project predictability as an effect of finding problems early. Not only did this effect receive the second highest average score in the data set: it also received the lowest standard deviation and was the only effect that every individual participating in the study perceived to some extent. This indicates a strong consensus among the interviewees.
In addition to this, the data suggests that predictability may also be improved by early non-functional system testing and integrating outside of the project's critical path, although to a lesser degree. It deserves to be pointed out, that even though the average scores of questions 10 and 11 were not among the highest in the data set, they do not show the same polarization in responses as questions pertaining to communication and productivity (see Sections 2.5.2.2 and 2.5.2.3).

![Graph of Experienced Continuous Integration Effects](image)

**Figure 7:** Average scores assigned by interviewees in response to questions 9, 10 and 11 (see Section 2.4.2.4). The lower and upper bounds of the boxes represent the first and third quartiles, respectively.

### 2.5.3 Validation of Hypotheses

Based on the data presented above, we find that the first hypothesis (continuous integration supports the agile testing practices of automated customer acceptance tests and writing unit tests in conjunction with production code) can not be validated by this case study, although interesting questions remain to be answered.

Furthermore, we find that the second hypothesis (continuous integration contributes to improved communication both within and between teams) is validated, with the caveat that experiences are very disparate.

In addition, we find that the fourth hypothesis (continuous integration improves project predictability as an effect of finding problems earlier) is validated.

Finally, the third hypothesis (continuous integration contributes to increased developer productivity as an effect of facilitating parallel development in the same source context and less compiling and testing locally before checking in) is partly validated: we found support for the first of the stipulated causes, but not the latter.
2.6 Conclusion

It is our conclusion that there exists not one, but several benefits to continuous integration. We also find that each of our hypotheses (see Section 2.3) is, at least partly, supported by the collected data, even in cases where they can not be unambiguously validated.

It is shown that there is a relationship between continuous integration and the agile testing methods of automated customer acceptance tests and writing unit tests in conjunction with new production code. There are, however, unanswered questions raised as to whether continuous integration supports the unit test practice, or if it's the other way around, or if they support each other. It is also difficult to isolate the effect of continuous integration on the practice of automated customer acceptance testing from contextual factors (such as organizational structure, culture and customer availability) in our data.

Furthermore the data shows, with the caveat that interviewees report disparate experiences, that continuous integration is generally perceived as having a positive effect on communication – not just within the team, as suggested by [Downs 2010], but in larger projects also between teams.

There is also strong support for the hypothesis that continuous integration improves developer productivity, but only for one of the reasons originally stipulated: while there is a strong consensus in the study that it facilitates parallel development, there is very weak support for the time saving aspect proposed by [Miller 2008]. It shall be noted, however, that in this case there is an exceptional lack of consensus: a number of interviewees do perceive this effect very strongly. This leaves us with an unanswered question: why do the experiences from this proposed effect differ? Are there factors that determine whether it's a benefit or not to save time by not compiling and testing before checking in?

In addition to the causes for improved productivity mentioned in the original hypothesis, we also found support for productivity increases due to more effective troubleshooting and easier re-basing and merging, although to a lesser extent than the parallel development effect.

We also find that our case study strongly supports the hypothesis that continuous integration improves project predictability by finding problems earlier. In addition to this, it is also shown that predictability is further improved because non-functional system testing can be performed early and integration can be performed outside of the project's critical path.

Additionally, we find no reason to believe that the effects discussed in this article constitute an exhaustive list of continuous integration benefits. It should also be noted that potentially negative effects of continuous integration have not been discussed, which we believe to be an important topic of further research.

Finally, it is our conclusion that, for most effects, the standard deviation of responses is very high, indicating that experiences of continuous integration differ. Viewed in light of the wide spectrum of continuous integration benefits described in literature, however, it is not altogether surprising. We do, however, believe that this is an important area that deserves further investigation: what are the causes of these differences in experienced continuous integration effects? Are they due to individual perception, differences in culture and process between the development projects, or inherent differences in the products being developed? Or could it be that the concept of continuous integration has been interpreted and implemented in different ways? At the time of writing this is not clear. It is conceivable, however, that a better understanding of potential differences in continuous integration implementations and their effects could help software development projects to shape their continuous integration in such a way as to optimize for the benefits they seek to achieve.
Acknowledgments

We would like to extend our sincere thanks to all the interviewees who shared their valuable time and experiences with us. We also want to thank Ericsson AB for allowing us to study their product development.