UNIVERSITY OF GOTHENBURG

# Metrics to measure the impact of continuous integration:

An empirical case study

*Bachelor of Science Thesis [in the Programme Software engineering and management]*

NAHID VAFAIE
MIKAEL ARVISDSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, augusti 2013

# Metrics to measure the impact of continuous integration:
## An empirical case study

Nahid Vafaie
Mikael Arvidsson



Examiner: Ana Magazinius

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

# Metrics to measure the impact of continuous integration:

An empirical case study

**Mikael Arvidsson**
Dept. Software Engineering & Management
Gothenburg University
Gothenburg, Sweden
mikael@apper.nu

**Nahid Vafaie**
Dept. Software Engineering & Management
Gothenburg University
Gothenburg, Sweden
n.vafaie@gmail.com

## Abstract

*The purpose of this paper is to investigate the impact of the continuous integration in large global organizations and find suitable metrics to measure these impacts. Continuous integration is a practice within agile movement in which members of a team integrate their work frequently, at least daily. Metrics are needed to measure the impacts of the continuous integration and compare them with other development process. This measurement allows managers within the organizations to assess the progress and impact of process changes and make data driven decisions. The research has been conducted as an empirical case study in a large software organization using a qualitative research approach and the result has been analyzed using thematic analysis. A total of 12 interviews were conducted with highly qualified software professionals about the impact of continuous integration in their organization. Several experienced positive and negative impacts are identified and ways to measure these impacts are investigated. The findings in this paper can be used by organizations to justify their continuous integration process or to predict and counter problems already during the introduction stage of that process.*

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, May 2013

# 1. Introduction

Continuous integration is an agile development practice that have been around since the 90's (Fowler and Foemmel, 2006, Bosch, 2013), but has recently increased in popularity with the agile movement. In the term continuous integration, integration refers to assembly of software parts and continuous refers to the absence of time-constraints (Holck and Jørgensen, 2007). Continuous integration originated in the eXtreme Programming development method (Beck, 2000). The Unified Process method also labels certain activities as continuous integration (Holck and Jørgensen, 2007). However, the principles of continuous integration can be applied to any iterative programming model in the agile development methodology (ibid). In the continuous integration practice, members of a team integrate their work frequently, usually at least daily, leading to multiple integrations per day. Each integration is verified by an automated build to detect integration errors as quickly as possible (Fowler and Foemmel, 2006).

To successfully measure the impact of the continuous integration process and compare it with the previous development process, suitable metrics are needed. The usage of quantitative measurements can be important and are often used within all sciences in order to validate the research and draw conclusions from the findings. This is true within computer science, and researchers have put major effort into finding suitable metrics for evaluating different aspects of software development. Futrell, Shafer and Safer (2000) describe a *metric* as a quantifiable measurement of a software product, process or project that can be directly observed, calculated or predicted. According to Basili et al. (1988) *measurement* is a process by which numbers are assigned to attributes of entities in the real world in a way that attribute them according to defined rules. Metrics are standards that define measurable attributes of entities, their units and their scopes. Software measurement allows managers to make timely, data driven decisions, to assess the progress and impact of process changes. It helps project managers by giving them a baseline to understand in what stage of the process they currently are, and to have a precise plan for the future of the process based on the goals and the results they gain by this measurement.

The contribution of this paper is to gain knowledge about the impact of continuous integration in large software organizations and then to explore suitable metrics in order to measure the impacts of continuous integration. The scope of this paper does not include any psychological or social impact aspects of continuous integration.

**Research questions:**
**RQ1:** What is the impact of implementing continuous integration in an organization?
**RQ2:** What metrics are most suitable to measure the impact of continuous integration in an organization?

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, May 2013

# 2. Related research

In this section we review the concept of positive and negative impacts of continuous integration as well as metrics to measure these impacts in an organization, in the related literature.

## 2.1 Positive impact of continuous integration

Continuous integration implies *communication*. Everyone in the organization will be able to get *the latest executable version of the software* and run it for the purpose of testing, demonstration or to observe the recent changes (Duvall et al., 2007, Fowler and Foemmel, 2006, Paul, 2007, Ståhl and Bosch, 2013). Ståhl and Bosch (2013) also state that continuous integration has a positive effect on communication, not only within the team but in larger projects also between teams. Continuous integration provides just-in-time information on the recent build status and quality metrics which will help everyone in the team to make effective decisions (Duvall et al., 2007).

*Immediate feedback* is one of the most positive impacts of continuous integration (Duvall et al., 2007, Paul, 2007, Rogers, 2004). This is usually shown when the developers make changes to their code so that a test case or a build fails. When this happens, the developer receives feedback about the failures so that these failures can be addressed directly (ibid). Both Fowler and Foemmel (2006) and Duvall et al. (2007) agree that continuous integration does not guarantee bug-free software, but it provides the possibility to continuously test and inspect it. Thus it makes it easier to find and remove bugs before they become widespread within the organization.

Fowler and Foemmel (2006) as well as Holck and Jørgensen (2007) believe that a greater benefit of continuous integration is *reduced integration risks*, due to the errors being found early in the process. According to Fowler and Foemmel (2006) the problem with deferred integration is that it is very hard to predict how long it will take to finish, and worse, it is very hard to see where you are in the process. This situation creates a blind spot in one of the most essential parts of the process. Continuous integration has the ability to completely solve this problem through integrating several times daily. In the continuous integration process, there are no long integration spawns so it can eliminate the blind spot (ibid). Continuous integration is also an alternative to ‘big bang’ integration where all modules are combined once. Big bang integration usually results in large number of errors which will be hard to address and correct (Holck et al., 2007).

Continuous integration can enable an organization to *release deployable software at any point in time* (Duvall et al., 2007, Fowler and Foemmel, 2006, Paul, 2007). With the help of continuous integration there will be small changes to the source code, and these changes will be integrated with the rest of the code base on a regular basis. If a problem occurs, the team members will be notified and then their solution will be applied to the software immediately. Fowler and Foemmel (2006) also state that continuous integration removes one of the biggest barriers to frequent deployment. Frequent deployment allows customers to get new features more rapidly, to give more rapid feedback on those features and become more collaborative in the development cycle (ibid).

According to Humble and Farley (2010) another positive impact of continuous integration may include a *reduction of stress* in all the people involved with the software release. They argue that a key to

reduced stress is to actively perform the automated deployment process as frequently as possible (ibid). But this has not been confirmed by other literature, in fact Fowler and Foemmel (2006) and Holck and Jørgensen (2007) state that the responsibilities of each individual will increase, so it can be interpreted that the stress level will not be decreased.

Continuous integration helps to *reduce repetitive manual processes* (Duvall et al., 2007). It affects all project activities including code compilation, database integration, function and system testing, deployment and feedback. The process will run the same way every time a commit occurs in the version control repository.

Continuous integration creates *motivation* (Duvall et al., 2007, Holck and Jørgensen, 2007). Continuous integration motivates all the members of the teams working with it to try to fix the faults as soon as possible in order to keep the build clean and unbroken. Everyone is motivated to find and try to fix the error regardless of who introduced it (Holck et al., 2007). By rebuilding and testing software in a clean environment using the same process and scripts on a continual basis, *the amount of assumptions will be reduced* (Duvall et al., 2007), and therefore *project predictability will be improved* (Ståhl and Bosch, 2013). It will also increase the *product confidence.* With every build, every member in the team will know that tests are run to verify the build, that project coding and design standards are met and that the result is a functionally testable product. Continuous integration can help the team members to be confident enough to make changes in the code, since they will be informed when something goes wrong (Duvall et al., 2007).
Team member also will be able to *track the general health and complexity of the product* over time (Fowler and Foemmel, 2006) and therefore find the *courage to innovate new improvements* (Ståhl and Bosch, 2013 and Duvall et al., 2007).

## 2.2 Negative impact of continuous integration

We could not find as many articles describing “disadvantages of continuous integration” or “negative impact of continuous integration”. However there are some problems that prevent organizations from implementing continuous integration as part of their software process. In this section these problems, as they are described in literature, are being discussed.

*Increased overhead in maintaining* the continuous integration system is usually considered as an obstacle against using continuous integration (Duvall et al., 2007, Rogers, 2004). Regardless of whether continuous integration is used or not, there is still a need to integrate, test, inspect and deploy the software. So it is better to manage a robust continuous integration process with automatic builds than to manage manual processes. However complicated multi platform projects are the ones that need continuous integration the most, and still these projects often have more resistance toward applying continuous integration because they think it is too much work (Duvall et al., 2007). This is contradicting Rogers (2004) that argue that continuous integration is about people more than about the tools that they use. It is easy to forget that continuous integration is a practice and that it is about what people do, not what tools they use to help them (ibid).

One of the barriers toward implementing continuous integration is that some people feel that in order to implement continuous integration they need to go through *many changes for their legacy project*. But in this case, incremental approach toward continuous integration is most effective (Duvall et al., 2007).

If developers do not perform a private build prior to commit their code to the version control repository, the result can be *too many failed builds* and is usually because a developer forgets to check in or have some failed tests. But in this case, when using continuous integration, rapid response is imperative due to the frequency of changes. (Duvall et al., 2007).

## 2.3 Metrics to measure the impacts of continuous integration

Generally, project metrics are measured to determine the current state of the health of a project (Tsourveloudis and Valavanis, 2002, Kassim and Zain, 2004). The identification and measurement of valid parameters that affect software development have been a lacking factor in many software projects (Pádua, 2010). The followings reasons can be used to explain why metrics are useful in software development (Misra and Omorodion, 2011):

- To make decisions in business
- To challenge team members
- Team members can be proud of the results
- To determine success
- It increases satisfaction
- It helps to change the behavior of team mates
- It increases decision making process

The important question is, what should be measured? Whatever is chosen to be measured, will have an enormous influence on the behavior of the team (Humble and Farley, 2010). For example if the area of measurement is the number of code lines that each developer writes, the impact might be that the developers place focus on writing many short lines of code instead of smart code. Measuring the number of defects fixed, has the effect on testers to log bugs that could be fixed by a quick discussion with a developer. So according to the lean philosophy it is important to have a global metric that can be used to determine if the delivery process as a whole has a problem (Humble and Farley, 2010).

With the development of the Agile software development techniques, there is a need for metrics that suit the measurement of the processes, products and projects involved in this new system. Metrics are basically selected based on the needs of the firm and the kind of software project. A team can design its own set of metrics based on the general classification of metrics available and use them to evaluate its activities and measure value depending on the kind of project and their needs. These are some core Agile metrics, their uses and measurements that are taken under them (Misra and Omorodion, 2011):

- Product Metrics deal with size metrics, architecture metrics, structure metrics, quality metrics and complexity metrics.
- Resource Metrics deal with personnel metrics (effort metrics, etc), software metrics and hardware metrics, and performance metrics.
- Process Metrics: Maturity metrics, management metrics and life cycle metrics are in this category.
- Project Metrics: Earned business value, cost, time, quality, risk and etc.
- Strategic metrics (net present value, earned business value, return on investment, etc).

- Engineer metrics (scope burn up, cost per iteration, etc).
- Test metrics (unit tests per user story, acceptance tests per user story, defect counts per user story. test times to run, time to fix tests, etc).
- Iteration metrics (velocity based on story points velocity based on the ideal engineering hours, backlog size, etc).
- Automation metrics (code coverage, number of builds per day, number of failed and succeeded builds, etc).
- Code metrics (lines of code size, code duplication, escaped bugs, etc).
- Project management metrics (schedule variance, performance variance, cycle time, etc).

Each sub-metric also defines a range of metrics such as velocity, running tested feature, story points, function points, earned business value, return on investment, effort estimates, downtime and etc. A developer or tester can define more metrics based on all the above metrics, based on the needs of the team, development process or other factors. These metrics include velocity history, user stories, stories completed, bugs open today, sprints with stories incomplete, progress in a release, exact progress in an iteration, on-time-delivery and etc (Misra and Omorodion, 2011).

Almost all of the above metrics result into quantitative measures based on the piece of software or collected data from the process. These metrics are not always equally applicable throughout all development methodologies, especially when moving from waterfall towards agile development (Misra and Omorodion, 2011). Agile development techniques indicates that the technique used for measurement for the traditional development processes may not apply. There are other aspects such as motivation, commitment and satisfaction which are always hard to find quantifiable measures for them (ibid).

It was not easy to find metrics to measure the impact of continuous integration specifically, in related research reviewed so far, but since continuous integration is a practice within Agile software process, it can be assumed that some of the metrics within Agile software process area such as quality metrics, process metrics and project management metrics can be applied to continuous integration too. One of the metrics suggested by Williams, Krebs, Layman, Antón and Abrahamsson (2004) is *response to customer change* (the number of user stories added and removed based on customer priority and preference change). This metric is important because it shows the degree of flexibility or agility among the teams. *Internally-visible quality* which is based upon the defects identified prior to release to customer and *externally-visible quality* assessed by the customer are good metrics to measure the quality of the project and product (Williams et al., 2004). According to Kan (2003) and (Williams et al., 2004) *Customer satisfaction* is another metric that can show how much a software process is successful. *Pre release defect density* (number and lifetime of defects) and *defect removal efficiency* are also good indicators of the quality (Shen and Ju, 2007). Kan (2003) suggests that the *Mean time to failure*, *defect density,* and *customer problems* are suitable metrics to measure the quality of the product. To measure the software delivery process, the most important global metric is *cycle time* (Humble and Farley, 2010). Cycle time is the time starting with the time deciding that a feature needs to be implemented until this feature is released to the customer. This metric is hard to measure because it covers many parts of the software process, from analysis, through development, testing, to release, but it still tells more about the process than any other metric (ibid).

# 3. Methodology

## 3.1 Research site

The research has been conducted in a large organization currently located in Gothenburg, Sweden. The site that was the base of the case study for this research had started implementing continuous integration as a pilot in one area of their organization involving only few teams. So the process of continuous integration was at the beginning stage. The implementation was not in full scale but the plan was to involve more products and teams in the next coming sprint.

## 3.2 Research approach

The focus of the study was to investigate the impact of continuous integration. The study has therefore been conducted using a qualitative research approach, which can be used to answer the research questions stated.

## 3.3 Data collection

A total of 12 employees from the teams working with continuous integration, were interviewed with a semi-structured interview approach .This approach allows for improvisation, which was used during the interviews to ask follow-up questions. The roles of the participants were managers and developers. About 60 minutes was spent on each interview and all the interviews were recorded using an audio recording device. The interview was focused on the impact of continuous integration and the possibility to measure the impact.

The interview guide (see appendix A for the interview guide) was written to guide the interview and make sure that all the questions were covered if the interview got off track.

All the interviewees were selected using a convenience sample because of the limited resources that was available. In the final stage of the data collection, only 10 of the interviews were fully transcribed due to time constraints, but the findings of all the 12 interviews were used in the paper. The transcription of the recorded interviews was done to ensure that the data collected was correct. The alternative way was to transcribe directly during the interview, but direct transcription could lead to misinterpreted data.

## 3.4 Data analysis

The data gathered from the interviews has been analyzed using the thematic analysis approach. The following steps were taken to analyze the data:

1. Transcription step:
   In the transcription step we familiarized ourselves with the data that we got from the interviews. The data was transformed from verbal sound from the recordings, to text that was much easier to process further.

2. Finding the initial codes:
   In this step we decoded the data from the perspective of our research questions. The information was considered as “code” when it could be used to partly or fully answer one of our research questions.

3. Finding themes:
   At this step we grouped the codes into different groups based on the similarities. These themes/groups were then considered impact areas and were spitted into positive impact, negative impact and impact measurement categories.

4. Cross checking the themes:
   In this step we reviewed the themes that we had found from the analyzed codes to make sure that all the themes corresponded correctly to the data.

### 3.5 Data validation strategy

To provide more validity to the research we decided to take the findings back to the participants to see if they agreed with the results presented. We also tried to give a rich description of the process and findings so that the reader could get a better understanding and therefore gain more credibility. In the paper we tried to present all negative or contradictory information with the purpose of creating more credibility for the reader.

We tried to anticipate the potential ethical issues that could occur during the research. When interviewing the participants in the site, we asked for the participants' permission to record the interview and at the same time we assured them that the interview would be held confidential if they wished so. The identification and confidentiality of the organization and interviewees are not disclosed in the paper.

The weakness of this paper lies mostly on this fact that the interviewees were not familiar with the process of continuous integration, so the information they provided could be partly incorrect or insufficient.

## 4. Results

In this section we interpret the results of our interviews. This section is structured according to the same categories identified in related research.

### 4.1 Positive impacts of continuous integration

Data about the positive impact of implementing continuous integration were gathered from interviewees. They were asked about the impact of the continuous integration in their organization. *Table 1* displays a summary of all positive impacts that the interviewees have experienced with the introduction of continuous integration at their workplace.

*Table 1. Positive impacts of continuous integration found in the interviews*

| Theme | Positive impact |
|---|---|
| Feedback | - Immediate feedback<br>- Real-time information helps to make better decisions<br>- Visibility at any time to everyone<br>- Improved visualization of the code quality<br>- Gaining the ability to measure the quality continuously<br>- Faster feedback from the customer |
| Performance | - Quick fixation of faults<br>- Faster integration<br>- Fast fault finding<br>- Increased team performance<br>- Increased integration capability with the main flow<br>- Less time to market<br>- Customer satisfaction |
| Responsibility | - Increased responsibility awareness toward quality and fixing the faults<br>- Less dependency on other teams<br>- Increased author pride<br>- Easier to pinpoint the fault owner |
| Code quality | - Always have the latest version of the code (rebasing)<br>- Less customer fault reports<br>- The majority of small faults are found early<br>- Automatic execution of tests<br>- Tests are performed where its important to test<br>- Small merges decrease severity of the conflicts<br>- Always have a stable build<br>- Number of faults decreased |
| Work environment | - Less stress<br>- More communication between coworkers<br>- Easier to plan work<br>- Sprint plans<br>- Change direction fast |

Feedback was a category of advantages mentioned by interviewees. Almost all of the interviewees stated that *immediate feedback* was the most important advantage of continuous integration. According to one of the managers, faults could be major and could affect a lot of other test cases that they were developing. So it was very helpful that they could get quick feedback in order to solve the problems fast. According to the interviewees *Real time information* was another positive impact that continuous integration brought into the organization. Some of them believed that by implementation of continuous integration they could get necessary information by looking at the dashboard and the portal to pick the suitable code or component test. *Visibility at any time to everyone* was an issue that some of the interviewees believed as one of the positive impacts of continuous integration in the organization.This meant that everyone could be able to see everything that was on with the software at all times. This impact was highly related to the *Improved visualization of the code quality* and to *gain the ability to measure the quality continuously*, which also were mentioned as positive impacts by interviewees. *Faster feedback from the customer* was another positive impact that some of the interviewees believed that they would gain by implementing continuous integration.

Performance was another category of advantages suggested by interviewees. Many of them believed that *fixation of the faults* was quicker after implementation of continuous integration. *faster integration, fast fault finding* as well as *increased integration capability with the main flow* were other impacts in this category mentioned by interviewees. They stated that at the same time continuous integration *has increased team performance.* They believed that after implementing the continuous integration in all levels of products and teams in the organization they would gain *customer satisfaction.*

Another category of positive impacts of continuous integration according to the interviewees was responsibility. A few of the interviewees stated that by implementing continuous integration *responsibility of awareness toward quality and fixing the faults* has increased. One of the developers believed that after implementation of continuous integration *dependency on other teams has decreased*. Another developer stated that it was now *easier to pinpoint the fault owner* and at the same it would *increase the author pride* when they knew that they have produced less faults.

Code quality was another category of the positive impact of continuous integration. The interviewees showed a general belief that now they always would have the *latest version of the code. less customer fault reports* has also been mentioned by the interviewees as positive impact of continuous integration. One of the managers explained the different stages of the process of creating a feature in this way that first customer requested a feature, then after the feature was ready they would test it in their lab and make report and sent back and finally developers would fixed the errors and release it to customer. Implementation of continuous integration would affect customers, because of the faster feedback that they could get from the customers and consequently would lead to faster releases. Other interviewees stated that the continuous integration process was only active within a fixed process and that even though continuous integration would make the department faster, it would not affect the customer because of the fixed release dates. Many of the interviewees also believed that *Small merges decrease severity of the conflicts* and that continuous integration would help them contain *a stable build* at all times. They also believed that it would lead to a *decrease in the Number of faults.* According to one of the managers feedback loop would quickly show if something was not working and this would increase the quality awareness. According to one of the developers by implementing continuous integration everyone was more responsible towards their code and would try to fix the problems as soon as possible. Some of the interviewees stated that their continuous integration included *automatic execution of tests* and *tests are performed where its important to test*.
These are both seen as positive impacts that can help the organization increase productivity through automation. As a result of the automatization they achieved and *the majority of small faults were found early* in the process. Continuous integration process has also helped them to *pinpoint the fault owner easier* and has produced *less dependency on other teams.*

Work environment has also been one of the areas affected positively by implementation of continuous integration according to interviewees. Some of them believed that the *sprint plans* were based on the information they received as a result of implementation of continuous integration, and that it helped them to *plan work easier* and *change direction faster*. *Less stress* was another impact that a few of interviewees believed as an advantage of continuous integration. They felt that since errors were introduced quickly, they could fix them as soon as these errors were introduced. So they felt less stress and it helped the teams to have *more communication with each others.*

## 4.2 Negative impacts of continuous integration

*Table 2. Negative impacts of continuous integration found in the interviews*

| Theme | Negative impact |
|---|---|
| Work environment | - Increase of stress level<br>- Harder to isolate yourself to work on something<br>- Complexity of work has been increased |
| Code quality | - Not always sure about the build quality<br>- Faults have the possibility to propagate fast<br>- Currently unstable baseline |
| Responsibility | - Too many people can enter the baseline |
| Development process | - No clear guidelines for working with continuous integration |

*Table 2* shows a summary of the negative impacts that the interviewees have experienced with their implementation of continuous integration. Many of the interviewees believed that continuous integration was flawless and did not have any negative impacts on the organization. Some of the negative impact that the interviewees talked about during the interviews were not a direct product of continuous integration itself, but a consequence of the way continuous integration was implemented at the organization and because of their early stage in the continuous integration process.

Work environment was one category of negative impacts according to some of the interviewees. In this category, the *stress level* was mentioned as a negative impact of continuous integration by a few of interviewees, which showed how people interpreted stress differently. A developer stated that he would feel more stressed now because after introducing some faults they had to fix it immediately. A manager explained that introducing fault for some developers meant a failure. But mindset should shift and they should know that introducing a fault was a learning and there would not be a punishment for that as long as they would feel the responsibility to fix the errors. One interviewee felt that it was *harder to isolate himself* with the purpose of working on something after the introduction of continuous integration. It was considered as a negative impact of continuous integration for him and he explained that according to some calculations that some managers did, it turned out that those teams who had been sitting in their cave for a couple of months had produced features faster. So he felt that now it was hard for him to do the same thing he used to do before.

Some of the interviewees felt that continuous integration has added more complexity to their work. One of the developers mentioned the issue of *complexity*, stating that since some teams might use some parts of the code from other teams and change it, then it would create complexity. This situation would create a lof of communication to ask how they should proceed and needed more synchronizing. Another developer also stated that continuous integration has affected the complexity in a negative way, although he believed that it should affect it in a positive way. He believed that there should be a way to make this process more stable.

Code quality was another category that covered a few of negative impacts of continuous integration. Some of the interviewees mentioned that they could not be sure about the quality of the build right now because of the unstable environment. One interviewee explained that faults had the *possibility to propagate fast*. Some of the interviewees were not sure about the *build quality* all the time and they felt that *baseline was currently unstable*. One of the interviewees explained his concern when it came to the quality of the build as some kind of drawback. He explained that at the time of testing, they had to the testing many times because they did not have the proper quality on the baseline.

Responsibility was another category according to some of the interviewees that was affected by continuous integration. One of the developers was concerned that *too many people could enter the baseline at the same time*, which might cause conflicts. Another developer mentioned that whenever he wanted to commit his code he had to wait for a good baseline, but then at the same time other people would come in and it would cause very big problem.

Development process was another area that was affected by continuous integration in a negative way. The fact that the organization has not got *clear guidelines for working with continuous integration* was another issue that some interviewees considered as a negative impact. Their main concern was that there was not clear guideline for the time that teams wanted to deliver but nobody knew how it should be done.

## 4.3 Metrics to measure the impacts of applying continuous integration

The findings from the part of the interviews that focused on the measuring the impact of continuous integration showed that most of the interviewees believed that the impact should be measured by amount of *time spent*, amount of *faults found*, amount of *tests done,* or *feedback from the customer*. The metrics that were mentioned by the interviewees are summarized in *table 3*.

*Table 3. Metrics to measure the impact of continuous integration found in the interviews*

| Theme | Metric type |
|---|---|
| Time | - Time from code freeze to delivery<br>- Testing time (only for the testing)<br>- Time spent on integration<br>- Hardening time<br>- Opportunity to cash time<br>- Fault report turnaround time (the time from a fault is found until its available to everyone)<br>- Time spent on build problems<br>- Sprint time<br>- Quality estimation time<br>- The team velocity |
| Faults | - Fault rate (amount of defects found when testing)<br>- Fault severity (type of fault) |
| Test | - Test pass rate<br>- Test status<br>- Testing time |
| Customer feedback | - Customer fault report<br>- Amount of requirements fulfilled |

Time related metrics was the first category that covered all the metrics related to time introduced by the interviewees. This category included *time from code freeze to delivery*, *testing time*, *time spent on integration* and *hardening time*. The *time from code freeze to delivery* was described by one of the managers as the time between the code freeze in the team until the time of the delivery. According to him, by implementation of the continuous integration this time has become very short. According to another manager, the length of *hardening time* was defined as the time starting with the introduction of the latest code changes until the time of release. According to this manager continuous integration has reduced the hardening time because testing procedure was running all the time in the background and would not take weeks afterwards. *Opportunity to cash time* was another metric in this category and according to one of the managers it meant the time that would take from when an idea of a feature was created until this feature would be delivered to the customer. According to this manager, the idea behind the continuous integration was to slice up customer requests and ideas and get these ideas into the market in as small portions as possible in order to get money faster. *Fault report turnaround time*, *correction time* and *lifetime of the defect* suggested by some of the interviewees, had different names but all indicated the time from when a fault was created until the solution was available for everyone. According to one of the managers lifetime of the defect was identified from when a problem was addressed until that problem was fixed on the main line. So the time between the problem was discovered until it was solved was very important. *Feedback time* mentioned by some other interviewees, was almost the same metric as *correction time*, *lifetime of the defect* and *fault report turnaround time*. ”*Time spent on build problems* was another metric suggested by some of the interviewees. Another time related metric was *sprint time*, which one of the developers described as a

good measurement in Scrum teams. As he explained there would always be dependencies between teams. In previous process, dependencies would create time frames that became a blocking but with the continuous integration developers could take a part of the code from other teams very fast and use that part without blocking and continue working on it. *Quality estimation time* was yet another metric suggested by some of the interviewees. One of the managers explained that this metric could measure the time it would take for everyone in the organization to get a common understanding of the product health. The *team velocity* metrics was suggested by some of the interviewees. One developer explained that velocity in the teams was based on the fact that the deliveries and rebase should be simpler when the teams verify more often.

The second category of metrics suggested by the interviewees was "fault related metrics". *Fault rate* was one of the metric mentioned by some interviewees. *Fault rate* was described as the amount of faults that were detected during a certain measurement period by one of the interviewees. The *type of fault* should also be measured according to some interviewees. According to them if it was a severe fault it could crash the whole software, so these were the most important to measure and keep statistics on. Another interviewee suggested that it was good to know both the number of the faults and where in the process it occurs.

The third category of the metrics summarization table covered *test pass rate, test status* and *testing time.* One of the developers mentioned that before applying continuous integration they used to measure the progress by number of test pass and he believed that they could still use it to measure the impact of continuous integration. Another developer suggested *test status* as a good metric, but then mentioned that it was really hard to measure it.

Customer satisfaction was another category introduced as a metric by the interviewees. This category included: *customer fault report* and *amount of requirements fulfilled.* Some of the interviewees believed that *customer faults report* would be less, with applying continuous integration and that it was a good measurement to compare two processes with each other. One of the developers stated that by testing more, the number of defects would increase which would be fixed quickly, but the number of defects reported by customer should decrease. The *amount of requirements fulfilled* was suggested as a metric to compare the impact of continuous integration with previous process by some of the interviewees. It was basically a measurement to see how many features the developers could fit into the final product.

# 5. Discussion

In this section, we discuss similarities and differences between our findings and previous research in relation to our research questions what are the impacts of implementing continuous integration? and what metrics are most suitable to measure the impact of continuous integration in an organization?

## 5.1 Positive impact of continuous integration

The advantages of continuous integration mentioned by interviewees were the real tangible advantages that they had gained so far. They believed that after fully implementation of continuous integration they would gain more benefits. Due to the discipline and the role that these interviewees had in the organization, sometimes their responses to the questions were different from each other. The reason for this could be the degree of their involvement in the continuous integration process, or the problems and

difficulties that they were facing not because of the continuous integration itself, but because of the natural problems when an organization goes through a transition from one process to another. These problems could be technical, environmental, organizational or human resistance.

Most of the interviewees agreed that *immediate feedback* was one of the good impacts of continuous integration. According to one of the interviewees there were benefits of getting quick feedback, such as everyone would feel confident that their changes in the code would not cause any problem. Duvall et al. (2007) and Fowler and Foemmel (2006) also suggest that continuous integration would provide rapid feedback.

A few of the interviewees believed that *communication* between team members and even within teams got better after implementation of continuous integration. This idea is supported by Duvall et al. (2007) and Fowler and Foemmel (2006) and Ståhl and Bosch (2013). *Just-in.time information* and *visibility at any time to everyone* were two advantages that most of the interviewees stated. They suggested that when they wanted to start their weekly integration, they would get necessary information by looking at the dashboard and the portal. Even some of them believed that continuous integration has provided them information to *plan their work easier, plan their sprints* and *change direction faster*. This is totally in line with the study of Duvall et al. (2007) who suggests that continuous integration would provide *just--in-time information* to help the team members in the team to make effective decision.

*Fast fault finding, quick fixation of faults,* and f*aster integration* were stated by the interviewees as the positive impacts of continuous integration which are in line with the idea of Fowler and Foemmel (2006) as well as Holck and Jørgensen (2007) who believed that one of the greatest benefit of continuous integration was *reduced integration risks* due to *the errors found early in the process*. Many interviewees believed that they could find *errors early in the process* after they implemented continuous integration and they would see it as a great benefit that continuous integration could bring to the organization.

Another advantage that most interviewees suggested was *increased responsibility awareness toward quality and fixing errors* . In the literature there was not an advantage of continuous integration specifically labeled as such but some of the literature derived it indirectly from the advantage of getting immediate feedback and having continuously a health build. Fowler stated that “With a good build, I can then think about committing my changes into the repository. The twist, of course, is that other people may, and usually have made changes to the mainline before I get chance to commit. So first I update my working copy with their changes and rebuild. If their changes clash with my changes, it will manifest as a failure either in the compilation or in the tests. In this case it's my responsibility to fix this and repeat until I can build a working copy that is properly synchronized with the mainline. ” (Fowler and Foemmel, 2006, P.3). Holck, et al. (2007) described this advantage as *motivation.* They believed that continuous integration would create necessary motivation in all the members of the teams working with it and to try to fix the faults as soon as possible in order to keep the build clean and unbroken. Everyone would try to fix the error regardless of whether the introducer of the fault was himself or another person.

A few of interviewees also believed that *latest executable version of the software* was one of the positive impacts of continuous integration. This is also supported by Duvall et al. (2007) who stated that everyone in the organization could run this version for the purpose of testing, demonstration or to observe the recent changes.

Some of the interviewees also described *reduction of stress* as a positive advantage of continuous integration. They felt less stressed because they could get quick feedback after they would introduce faults into the system, so they had better chance to fix them immediately. According to Humble and Farley (2010), one of the obvious benefits of continuous integration was *reduction of stress* in people that are associated with a release.

*Less customer fault report* was mentioned as a positive impact of continuous integration by a few of the interviewees. Fowler and Foemmel (2006) stated that continuous integration would remove one of the biggest gaps between customer and development by frequent deployment. *Less customer fault report* was the result of frequent deployment. But it was a little surprising when the interviewees were asked about the *customer satisfaction* and the effect of continuous integration on the customer. Many of them believed that continuous integration was an internal process which would not affect the customer. On the other hand in the literature, *customer satisfaction* was one of the most important advantages of continuous integration. Frequent deployment would allow customers to get new features more rapidly which would let them to provide more rapid feedback on those features and become more collaborative in the development cycle. So why despite this fact that continuous integration should ultimately have the effect on the customers requirement and needs, some of the interviewees believed that it would not affect the customer? One of the reasons could be that the implementation of continuous integration was only in one part of the organization which was not dealing with delivering any feature to the customer at the time of interview. The second reason could be that, they still had fixed time-boxed deliveries to the customer, so even though continuous integration was optimizing the time, currently it would only affect the internal speed. One of the interviewees believed that continuous integration did not affect the customer because it was an internal efficiency. They stated that continuous integration currently was just a procedure to make the process faster, but it would affect the customer when continuous integration would be implemented in all levels of products. *Less time to market* was another advantage that was brought up by some of the interviewees. The result of this advantage would directly affect the customer which would lead consequently to *customer satisfaction*.

## 5.2 Negative impact of continuous integration

Many interviewees believed that there were not any negative impact about continuous integration and if there would exist any problem right now, it was not created by the continuous integration itself, it could be a technical problem or it existed because continuous integration had not been implemented yet in all parts of the products and teams. By reviewing the previous research about the impact of continuous integration, a few research discussed about some problems that could happen when an organization started to implement continuous integration. However these problems were assumed as negative impacts of continuous integration by some people. These research tried to explain that people should not consider these problems as the impact of continuous integration but as obstacles and problems in the way of implementing continuous integration.

*The level of stress* after breaking the code by committing a new code, was different among the interviewees. A few of the interviewees believed that after applying continuous integration the stress level was less, but some other believed that now they had more stress, and the rest believed that it was the same as before. But why some people felt more stressful after implementation of continuous integration even though the organization they were working was a forgiving organization so there would be no punishment for the errors they made. The common belief in the organization would allow developers to have errors in their code as long as they would try to fix it and learn lessons from that. This question was asked from the leaders and managers among the interviewees in the organization that why continuous integration was giving the feeling of more stress to some of the developers. One of the managers that it was normal that they got more stressed now when they broke the software and they did not have this kind of stress before. In his opinion this stress was a good sign. It meant that they could take the responsibility for their action. Another manager also believed that now developers would feel more stressful because they believed that it would hit them. It would make them worry that if they would introduce a fault, they would be punished for that. According to the interviewees conducted with managers and initiative of continuous integration, they have started trying to give the developers a notion that introducing a fault and fixing it, was a learning which was a good thing. They should see it as an opportunity.

Another issue that some of the interviewees assumed as a disadvantage of continuous integration at the moment was: *not being sure about the quality of the build.* However this idea was not supported by the literature. The quality of the build would be guaranteed by the amount and quality of the tests run on the integrated code. As Kaplewicz et al. (2005) suggested migration of large amounts of internal development projects into a continuous integration environment required tight planning and coordination to successfully migrate.

Some of the disadvantages mentioned by interviewees such as *lack of stability, lack of testing, need to be easier to stop integration, the process is not spread wide enough , no clear guideline for working with continuous integration* were the results of incomplete implementation of continuous integration. But as Duvall et al. (2007) suggested implementing continuous integration needed many changes but it should be through an incremental approach which would make it most effective.

## 5.3 Metrics to measure the impact of continuous integration

Most of the interviewees had clear idea about the impacts of the continuous integration, but when it came to metrics and how these impacts should be measured, most of the time they were not sure about the suitable metrics, at least it seemed that they had not thought about that before. The reason could be that continuous integration was very new in this part of the organization and it had not even been completed to show the impacts on the outcome. However when they thought more precise about these metrics, they could come with very good ideas and therefore suggested efficient metrics.Also when we reviewed previous research about metrics to measure the impact of continuous integration, we were not very successful to find enough articles in this regard. It could be due to our weakness in finding related article or due to this fact that less research had been done in this field. But most papers that we reviewed argued about metrics in software development in general. We assumed that since continuous integration

was one of the practices within Agile software development, it would be good enough to use one of the Agile metrics to measure the impact of continuous integration. So we focused on literature arguing about Agile metrics.

One of the interviewees who was a manager suggested *opportunity to cash time* as a good metrics. According to his definition, *opportunity to cash time* was the time started when an idea of a feature was introduced until the time this feature would be released to customer. Cycle *time* according to Humble and Farley (2010) was the time starting when it was decided that a feature needed to be implemented until this feature was released to customer. These two metrics are different name, but the functionality of both are exactly the same.

*Fault report turnaround time* or *correction time* was a metric introduced by some of the interviewees. This metric according to a manager meant the time from when a fault was created until the time it was available to everyone in the organization and was solved. On the other hand *Pre-release defect density* was a measurement introduced by Shen and Ju (2007) for measuring the quality of a product or project in a software development process. It could be also described as the number and lifetime of defects. *Internally-visible quality* which was introduced by Williams et al. (2004) was the defect identified prior to release to customer. These metrics despite of having different names had exactly the same functionality.

*Sprint time* and *team velocity* were metrics brought up by several interviewees. These two metrics were suggested by Misra and Omorodion (2011) as a suitable metrics. They stated that a developer or tester could define a metric based on the current metrics and the needs of the team, development process or other factors.

*Test pass rate, test status* and *testing time* which according to some of the interviewees were good metrics to measure the process of testing and were applicable to compare two processes such as continuous integration with waterfall process belonged to the Test Agile metrics. According to Oza and Korkala (2012) Test metrics consisted of *test times to run, time to fix the tests* resulted into quantitative measures based on the piece of software from a process. They also mentioned that these metrics were not always applicable through all development processes, especially when there was a transition from waterfall toward agile development.

*Customer fault report* and *Amount of requirements fulfilled* were two customer related metrics which were suggested by some of the interviewees as good metrics to measure the impact of continuous integration. As Williams et al. (2004) suggested *externally-visible quality assessed by the customer* was a good metric to measure the quality of the project and product. A few of interviewees also recommended that *customer satisfaction* could be another good metric that would show how much a software process was successful. Kan (2003) also stated that *customer problems* and *customer satisfaction* would cover the definition of software quality.

*Hardening time* according to one of the interviewees was the time when the latest code changes were introduced until the time the feature was released. Shen and Ju (2007) suggested *defect removal*

*efficiency* (a measure to detect defects before delivery) as a good indicator of quality which is totally in line with *hardening time.*

*Quality estimation time* was suggested by a few interviewees. One of them explained that this metric would show how much time, it would take for people involved in a software development process to get a common understanding of the product health in order to be able to release the product. We have not found similar metric in reviewed articles and papers to support this idea.

# 7. Conclusion

Continuous integration, an agile development practice has recently been implemented in many organizations. In continuous integration process, members of the team integrate their work every day. An automated build verifies each integration to detect errors as quickly as possible. This practice will naturally have impacts, positive or negative on the organization, product, and team members. This study attempts to identify the impacts of continuous integration and find suitable metrics to measure these impacts. The research has been conducted as an empirical case study in a large software organization using a qualitative research approach and then analyzed using thematic analysis. Interviews were conducted with 12 of the employees at the organization who have been piloting the continuous integration process in their daily work.

The findings from the conducted interviews were categorized into three areas; positive impact of continuous integration, negative impact of continuous integration and metrics to measure and compare the impact of continuous integration with the previous software development process.

Based on the findings from the interview and to answer **RQ1**, we concluded several impacts on *time*, *feedback*, *performance*, *responsibility*, *development process*, *code quality* and the *work environment*. Some of these impacts were seen as both positive and negative by the interviewees. The findings from the study that are related to **RQ2** shows that the impact can be measured by *time*, *faults*, *tests* and the *customer feedback received*.

The result of this study shows that even though the interviewees did not have any previous experience with continuous integration and continuous integration process was in its early stage in their organization, they thought that this process had already positive impacts on their organization, their daily work and the final product. Most of the positive impacts stated by interviewees were supported by the related research papers. How ever there were a few negative impacts suggested by some of the interviewees that were not supported by related articles, such as i*ncreased stress level* and *not being sure about the quality of the build.* A few more disadvantages of continuous integration mentioned by some of the interviewees such as *lack of stability, lack of testing, need to be easier to stop integration, the process is not spread wide eno*ugh and *no clear guideline for working with continuous integration* were not also supported by previous literature and they could be the result of incomplete implementation of continuous integration in their organization.

Findings related to suitable metrics to measure the impact of continuous integration shows that interviewees suggested applicable metrics that are suitable for measuring the process of continuous integration and compare it to other integration processes such as waterfall process. These metrics are *cycle time, opportunity to cash time , fault report turnaround time and hardening time.*

## 8. Future Work

Since this study was conducted at a stage where the organization had just recently introduced continuous integration to their development process, it would be of benefit to do a follow-up study at a later stage and compare the experienced benefits with the results of this study. We would also like to see the study replicated in other organizations to see if they experience different impacts of continuous integration and have different opinions of how to measure these impacts.

## 9. Acknowledgement

We would like to give a special thanks to the organization that allocated resources and made it possible for this study to be conducted in their environment. A special thank you also goes to our supervisor Agneta Nilsson for her effort and help with this paper and the related research.

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, May 2013

# References:


BASILI, V., ROMBACH, H., PARK, R., GOETHERT, W. & FLORAC, W. 1988. SENG 421: Software Metrics. *IEEE Trans. Software Engineering,* 14**,** 758-773.

BECK, K. & ANDRES, C. 2004. *Extreme programming explained: embrace change*, Addison-Wesley Professional.

DUVALL, P. M., MATYAS, S. & GLOVER, A. 2007. *Continuous integration: improving software quality and reducing risk*, Addison-Wesley Professional.

FOWLER, M. & FOEMMEL, M. 2006. Continuous integration. *Thought-Works) http://www.thoughtworks. com/Continuous Integration. pdf*.

FUTRELL, R. T., SHAFER, L. I. & SHAFER, D. F. 2001. *Quality software project management*, Prentice Hall PTR.

HOLCK, J. & JØRGENSEN, N. 2007. Continuous integration and quality assurance: a case study of two open source projects. *Australasian Journal of Information Systems,* 11.

HUMBLE, J. & FARLEY, D. 2010. *Continuous delivery: reliable software releases through build, test, and deployment automation*, Addison-Wesley Professional.

KAPLEWICZ, J., BAQIR,Y., NORMANDEAU, J., BEGUN, A., MARTIN, E., BLUM, A., & ANWER,T.,. 2005. Continuous Integration: Why and How to Build a Continuous Integration Environment for the .NET Platform. http://www.espusa.com/whitepapers/continuous_integration_v1.0.pdf. © Enterprise Solution Providers, Inc. 2005

KASSIM, N. M. & ZAIN, M. 2004. Assessing the Measurement of Organizational Agility. *Journal of American Academy of Business, Cambridge,* 4**,** 174-177.

MISRA, S. & OMORODION, M. 2011. Survey on agile metrics and their inter-relationship with other traditional development metrics. *SIGSOFT Softw. Eng. Notes,* 36**,** 1-3.

OZA, N., & KORKALA, M. 2012. Lessons Learned In Implementing Agile Software Development

PÁDUA, W. 2010. Measuring complexity, effectiveness and efficiency in software course projects. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*. Cape Town, South Africa: ACM.

PAUL, D. 2007. *Continuous Integration*, Pearson Education India.

ROGERS, R. O. 2004. Scaling continuous integration. *Extreme Programming and Agile Processes in Software Engineering.* Springer.

SHEN, B. & JU, D. 2007. On the measurement of agility in software process. *Software Process Dynamics and Agility.* Springer.

STÅHL, D. & BOSCH, J. Experienced Benefits of Continuous Integration in Industry Software Product Development: a Case Study. The 12th IASTED International Conference on Software Engineering, 2013.

TSOURVELOUDIS, N. C. & VALAVANIS, K. P. 2002. On the Measurement of Enterprise Agility. *Journal of Intelligent & Robotic Systems,* 33**,** 329-342.

WILLIAMS, L., KREBS, W., LAYMAN, L., ANTÓN, A. I. & ABRAHAMSSON, P. 2004. Toward a framework for evaluating extreme programming.

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, May 2013

# Appendices

Appendix A
Interview questions:

**Introduction questions:**

- Is it ok for you if we record this interview for later reference?
- Please introduce yourself, what are you doing and for how long, do you have a degree?
- Please describe your normal workday, what is part of your job?
- Do you have any previous experience with CI?

**General questions:**

- Can you explain a little about the process of CI?
- What is your opinion about the current implementation of CI?
    - Is there any room for improvement in the current implementation of CI?
- In what ways have CI affected your daily work (*both negative and positive ways*)?
    - In what areas?
    - Has it affected the quality of your work?
    - Name some good things and bad things about CI

- Does CI have an impact on the maintenance of previous products?
- Is there a difference between CI and the previous process when you break the code of the software?
    - What is the difference?
    - Does it affect the stress level?
- How close is the current process to pure CI?
- How much time was spent on integration before the CI implementation, and how much time do you spend now?
- How has the performance of the teams been affected by the CI?
- How has the sprint backlog been affected by implementing CI?
- Does CI provide you just-in-time information to help you for more effective decisions?
    - If yes, how does it?
- Does CI help you to get faster feedback from customers?
- Does CI help you to get immediate feedback on your work?
- Does CI affect the complexity of your work?

**Metrics questions:**

- Upon what criterion should this CI be measured in your opinion?
- In your opinion, what metrics are suitable to measure CI and compare it with the previous processes (both qualitative and quantitative)?
    - ***Otherwise:*** Which person in your opinion is most qualified to answer this question?

- Do you use any specific metrics to measure the internal team progress (eg. burndown charts)?
- How have you measured progress in previous process?

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Gothenburg, Sweden, May 2013