Institut für Softwaretechnologie

Abteilung Software Engineering

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Master Thesis Nr. 3201 - 003

Improving Software Quality with Continuous Integration in an Agile Development Process

Lingwei Li

Studiengang:	INFOTECH
Prüfer:	Prof. Dr. Stefan Wagner
Betreuer:	Yang Wang
begonnen am:	04.04.2016
beendet am:	16.09.2016
CR-Klassifikation:	D.2.9

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben.
Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet.
Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens.
Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht.
Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Unterschrift:

Stuttgart, 16.09.2016

Declaration

I hereby declare that the work presented in this thesis is entirely my own.
I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations.
Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before.
The electronic copy is consistent with all submitted copies.

Signature:

Stuttgart, 16.09.2016

Acknowledgements

The author of this thesis would like to thank the project PBK PT from Daimler for providing an inspiring forum for this research to take place. Special thanks go to my industry supervisor Toms Beske and Sven Bolz for giving me the opportunity to conduct this research in Daimler on their premises. And I am especially grateful to Toms Beske for his input, encouragement and support throughout the whole work.

I also would like to thank the supervisor Yang Wang for her academic and professional guidance on conducting the research and the invaluable feedback on the thesis. I wish to thank the examiner Prof. Dr. Stefan Wagner for evaluating this thesis.

Additionally, this study has obtained access to researching different agile projects in Daimler. Especially Daimler Fleetboard and Daimler Test Factory have provided valuable information and pointers for research. Warmest thanks to Markus Huehn and Kalyan Boggarapu from these two organisations for their great contribution on the case study, support in gathering case study data and share of their valuable practical experience!

Lingwei Li, Stuttgart 16.09.2016

Abstract

To face the growing complexity of software development and the increasing demand for the software quality, agile practices have been widely praised for their focus on fast feedback, shorter release cycle and frequent customer collaboration. As a typical agile practice to improve software quality and reduce the common risk for software development, Continuous Integration is of high demand to be adopted in the software development practice. Currently, CI (Continuous Integration) is not popularly used in the practice like other agile methods. The implementation of CI process is not well known in the practice. Some agile projects face difficulties to implement CI process in their software development. Vice versa, CI process cannot cover all the requirements from software development teams.

This thesis presents a case study on CI process in agile development process, in which the implementation of CI process in the practice and effects of CI process, especially on software quality has been researched with multiple units of analysis. The research results about CI process implementation can be referred as guidelines for the agile projects intending to implement CI. The occurred difficulties to implement CI process and the corresponding solutions from the researched agile projects can be learned from experience by the other practitioners. Moreover, the research results about effects of CI process on Software quality, Communication, Development process and Risk management can give a clear view to the practitioners, what benefits CI can bring to their agile projects. Finally, based on the research on implementation and effects of CI process, a set of optimizations of CI process to improve Software quality have been proposed and validated by the researcher. For both practice and academic research, the optimization proposes are invaluable references.

Keywords: Agile practice, Continuous Integration, CI, Software quality, Software development, Agile development process, Implementation, Effects, Case study

Contents

List of Figures

List of Tables

List of Abbreviations

PBK	Programm, Bedarf, Kapazität
CI	Continuous Integration
XP	Extreme Programming
RQ	Research Question
VCR	Version Control Repository
SW	Software
IBP	Integrated Business Planning
SolMan	Solution Manager
ChaRM	Change Request system
TMS	Transport Management System
QTP	Quick Test Professional
UFT	Unified Functional Testing
ALM	Application Lifecycle Management
TAO	Test Acceleration and Optimization

1 Introduction

1.1 State of the art of Continuous Integration

Nowadays, Software development is facing the challenge of rapidly changing requirements and demands on frequent product releases [1]. The Agile development process is getting more and more popular in the software development world, owing to its emphasis on individual and interactive development, the reaction to changing requirements, customer collaboration and continuous delivery. Apart from frequent product releases, companies have high demands on the quality of the software being developed.

Continuous integration is one of the most effective agile practices, which can shorten the feedback loop through continuous integration build and a set of automated processes. In continuous integration (referred to as CI subsequently), the software being developed will be integrated with the existing parts continuously, actually after every committed change. In continuous integration, the developer first uploads the code to the version control repository, then builds and tests the code part in CI server. Finally, feedback will be given to the developer. It has been suggested that continuous integration increases the frequency of software releases and shortens the feedback cycle [3]. Additionally, Miller and Carter [4, 5] stated that continuous integration reduces the time developers spend on checking in new code while maintaining the same level of product quality.

In a word, with CI not only feedback will be accelerated, but also the software quality can be improved, and the risk can be reduced [2].

1.2 Problem Statement

CI is an agile practice, which facilitates agile projects to realize agility in software development process. But in the software development practice, CI is not so well known and popularly used as other agile methods, for example Scrum. A lot of practitioners hold the view, that CI is just an additional process for agile development. How to build up effective CI process is not well known to some agile projects.

What effects CI process will bring to the software development is not fully and profoundly researched in the previous researches. In contrast, Scrum is well known to fas-

ten the development process. Duvall et al [2] stated that, with CI the software quality can be improved and the risk can be reduced. But little research has been done about what effects CI process has on software quality and risk management. The effects of CI process are also not well known in the software development practice world. The practitioners, who have never implemented CI process in their software development, are usually holding skeptical attitude for CI.

For both development and stakeholder side, the software quality of the developed product is of great significance. The software development practice world has great demands on improving software quality. But the previous research on CI has done little studies in this field: what effects has CI process on software quality, and how to realize positive effects on software quality with CI process.

1.3 Motivation

This study has been performed during a student partnership at the Daimler AG in Esslingen, Germany at the IT department for logistical processes in the Powertrain Area of Mercedes-Benz Passenger cars. For a new agile project in the IT department, the cooperation partner would like to get more knowledge about agile development processes, CI process and the effects of CI process.

Therefore, through this case study on different agile projects, the researcher tries to find out how CI process is built up in the researched agile projects, what challenges occur during building up CI process in the practice, and what effects their CI processes have on the software development. The case study consists of both interviews and questionnaire.

The research results would be very helpful for the practitioners. Through the experience of building up CI process from other agile projects, the practitioners who never used CI process can obtain an overview of how to build CI process. Through the research on CI effects, the practitioners who hold skeptical attitude for CI can obtain a clear view, what effects CI process has on the software development. A step further, through the research on CI process to software quality, the practitioners who have high demand on software quality can get an idea regarding, what effects CI process has on software quality in details, and how to realize the positive effects on software quality with CI process further. Based on practical demands on CI process, CI process can be optimized to meet greater demands of applications and to support more agile projects.

The research results could also fill the vacancy in the research field about, how CI process is built in the practice, what effects has CI process on software quality and other

additional aspects. The optimization of CI process can also be very meaningful for the research on CI.

1.4 Outline of the Thesis

The Outline of this thesis is structured as follows:

- **Section 1** introduces state of the art of continuous integration, problem statement, motivation, and outline of this thesis.
- **Section 2** describes the background and related work to this thesis.
- **Section 3** describes the research approach.
- **Section 4** introduces the data analysis procedure, data analysis techniques and data analysis tool for the case study.
- **Section 5** presents, how case study is conducted in questionnaire and interview, what data has been collected, and how data is coded to be analyzed in the next section, how the hypothesis is generated and confirmed, and how validity process has been taken for this case study.
- **Section 6** presents the result of data analysis.
- **Section 7** discusses the result of data analysis and threats occurred during data analysis.
- **Section 8** ends with the conclusion and future work.

2 Background and Related Work

2.1 Background

According to the definition from Martin Fowler, Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily, which leads to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. In the practice, many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly [7]. Lacoste describes Continuous Integration as an agile practice, which has its roots in the agile method- Extreme Programming (XP) [6]. As a supplement to the background of CI, agile software development with its methods and practices will be introduced in this section.

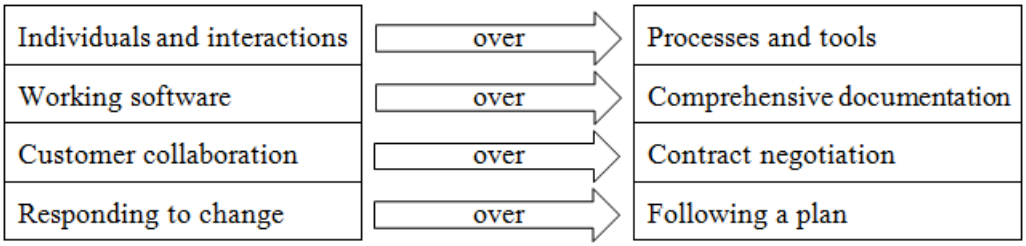
Duvall et al introduced the components of a CI process in detail and the effects of CI process at a superficial level. By performing automated and continuous database integration, testing, inspection, deployment, and feedback, the CI system can reduce common risks on the project, thus leading to better confidence and improved communication. Even better, this continuously repeated process can reduce risks throughout the whole software development lifecycle [2]. The components and construction of CI process will be introduced in detail in this section. Afterward, the effects of CI process will be summarized from the literature review and described in related work.

2.2 Agile Software Development

Agile software development contains a set of methods and practices based on values and principles described in the manifesto for agile software development, which is known as the agile manifesto. In agile software development, requirements and solutions are developed through collaboration between self-organizing and cross-functional teams, which utilize the appropriate practices for their context. Agile software development promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change [21].

Among its promotions, early delivery and continuous improvement can be realized through continuous integration.

According to the agile manifesto, in comparison with traditional software development, agile software development has some specific values [22], which have been described in the Figure 1:



(Source: Own illustration)

Figure 1: Comparison between agile and traditional software development

Manifesto authors have generated twelve principles in the Manifesto for Agile Software Development [23], which have been listed in Table 1.

Table 1: Twelve principles behind the Agile Manifesto

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.	7. Working software is the primary measure of progress.
2. Changing requirements are welcoming, even late in development. Agile processes rules change for the customer's competitive advantage.	8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	9. Continuous attention to technical excellence and good design enhances agility.
4. Business people and developers must work together daily throughout the project.	10. Simplicity--the art of maximizing the amount of work not done--is essential.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	11. The best architectures, requirements, and designs emerge from self-organizing teams.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	12. The team reflects at regular intervals on how to become more effective, then tunes and adjusts its behavior accordingly.

2.2.1 Agile Methods

Agile methods support software development in a whole life cycle, which are also known as software development frameworks. A set of agile software development methods are popularly used in the practice[15]. For example, Agile modeling, Agile Unified Process (AUP), Dynamic systems development method (DSDM), Extreme programming (XP), Feature-driven development (FDD), Kanban, Scrum. These agile methods are focused on different aspects of the software development life cycle, which have been demonstrated in Table 2.

Table 2: Agile methods and their aspects of software development life cycle

Agile method	**Aspects of software development life cycle**
XP, Agile modeling	Practices
Scrum, Kanban	Managing the flow of work
DSDM, AUP	whole development life cycle
FDD	Requirements specification and development

XP (Extreme programming) and Scrum are the most popular agile methods. Extreme programming is an agile software development methodology, which emphasis on improving software quality and providing reactions to changing requirements from customers. XP advocates frequent "releases" in short development cycles, which improves development productivity and provides frequent checkpoints during the whole software development lifecycle. So that, on one side, the customers can check the released product and give feedback. On the other side, the development team can adopt new customer requirements. Extreme programming includes some specific elements: programming in pairs, doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements, and frequent communication with the customer and among programmers. One typical practice of Extreme programming is pair programming [24].

Scrum is an agile project management framework. Scrum places great emphasis on how to organize an agile project. Specific roles are needed to build a scrum team: product owner, scrum master, and development team. Specific processes are needed to be executed: designing user story backlog, running the sprint, sprint repository and so on. Three key points in scrum are transparence, inspection, and adoption. In scrum, without acceptance criteria and displaying the test results, the quality of the software cannot be

assured. Scrum organization can be adjusted according to changing requirements from agile projects.

2.2.2 Agile Practices

Agile development is supported by a bundle of concrete practices, covering areas like requirements, design, modeling, coding, testing, project management, process, and quality [15]. Some notable agile practices with the relative function areas are displayed in Table 3.

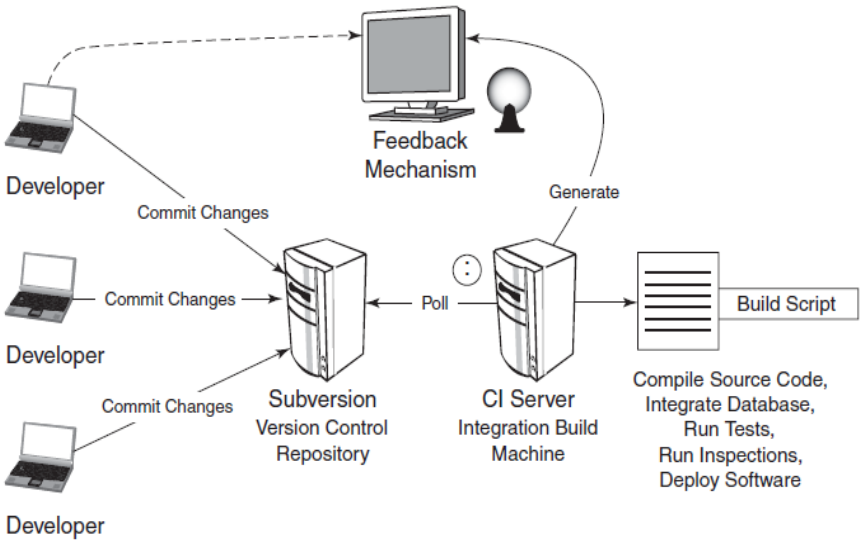
Table 3: Agile practices and their function areas

Agile practice	**Function area**
1.Acceptance test-driven development (ATDD)	requirements, design
2. Agile modeling	modeling
3. Continuous integration (CI)	coding, testing
4.Iterative and incremental development (IID)	coding, testing
5. Pair programming	coding, testing
6.Scrum events(sprint planning, daily scrum, sprint review and retrospective)	project management
7. Test-driven development (TDD)	process, quality
8. Agile testing	testing

2.3 Continuous Integration

2.3.1 Components of a CI process

Figure 2 shows the components of a CI process, which will be explained in detail in the following section.



(Source: Duvall et al, 2007)

Figure 2: Components of a CI process

Component 1: Developer runs private build

After finishing all of the related modifications, a developer will run a private build to integrate changes from the rest of the team. The changes are committed to the version control repository. A private build occurs at any time, but no integration build will occur unless changes applied to the version control repository [2].

Component 2: Version Control Repository

A version control repository is necessary to perform CI, of which the purpose is to manage changes to source code and other software assets by controlling access repository. In this way, all source code is available from one primary location and it allows the developer to go back in time and get different versions of source code or other files [2].

Component 3: CI Server

A CI server runs an integration build, which acts as a process for putting source code together and may consist of compilation, testing, inspection, and deployment. It is typically configured to check for changes in a version control repository regularly. The CI server will retrieve the source files and run a build script(s). Usually, a convenient dashboard is also provided by CI servers for publishing build results [2].

Component 4: Feedback Mechanism

A fast feedback on if there was a problem with the latest build is critical, which is helpful for fixing problems early and quickly and thereby reducing the common risk of development. One of the key purposes of CI is to give feedback from an integration build.

2.3.2 Processes in CI Sever

In CI Server, a set of processes may be running. Some important processes which are related to this study are introduced in the following:

Process 1: Build Script

A build script is used to compile, test, inspect, and deploy software, which can be a single script, or set of scripts. And it can be used without implementing a CI system. Ant, NAnt, make, MSBuild, and Rake are examples of build tools that can automate the software build cycle, but they don't provide CI by themselves [2].

Process 2: Source Code Compilation

Continuous source code compilation (involving creating executable code from the human-readable source) is one of the most basic and common features of a CI system, even become synonymous with CI. However, CI is much more than source code compilation [2].

Process 3: Database Integration

In the practice, the source code integration and database integration are considered as two completely separate processes, which are always performed by different groups. By using a CI system, the integration of a database can be ensured by the version control repository [2].

Process 4: Testing

Without automated tests, it will be difficult for developers to have confidence in software changes. CI provides automated, continuous testing. Most developers from projects with CI system use unit testing tools, such as JUnit, NUnit, or other xUnit frameworks. Furthermore, to speed up builds, different categories of tests can be run from a CI system, which may include unit, component, system, load/performance, security test and so on [2].

Process 5: Inspection

Automated code inspections is to enhance the quality of the software by enforcing rules. For instance, a project might have a rule that no class may be longer than 300 lines of non-commented code. CI system can be used to run these rules automatically against a code base [2].

Process 6: Deployment

Continuous deployment enables to deliver working, deployable software at any point in time. Another key purpose of a CI system is to generate the bundled software artifacts with the latest code changes and make it available to a testing environment. In Continu-

ous deployment, the source files from the version control repository must be checked out, a build must be performed, all tests and inspections must be successfully executed, the release must be labeled, and the deployment files must be staged [2].

2.3.3 Typical CI scenario and Features of CI

CI scenario

The steps in a CI scenario [2] will typically go something like this:

- First, a developer commits code to the version control repository.
- Meanwhile, the changes occurred in the version control repository will be detected by the CI sever, as the CI server on the integration build machine keeps on polling the repository for changes.
- Then the CI server retrieves the latest copy of the code from the repository and then executes a build script.
- Finally, the CI server generates feedback by e-mailing build results to specified project members.
- Afterward, the CI server continues to poll for changes in the version control repository.

Features of CI

Based on the above descriptions, there are only four features required for CI.

- A connection to a version control repository.
- A build script, which integrates the software.
- Some sort of feedback mechanism, such as e-mail.
- A process for integrating the source code changes, including manual or CI server.

The key to an effective CI system is that, an automated build is running with every change applied to the version control system. Other features can also be added to a specific CI system [2].

2.4 Researched effects of CI process

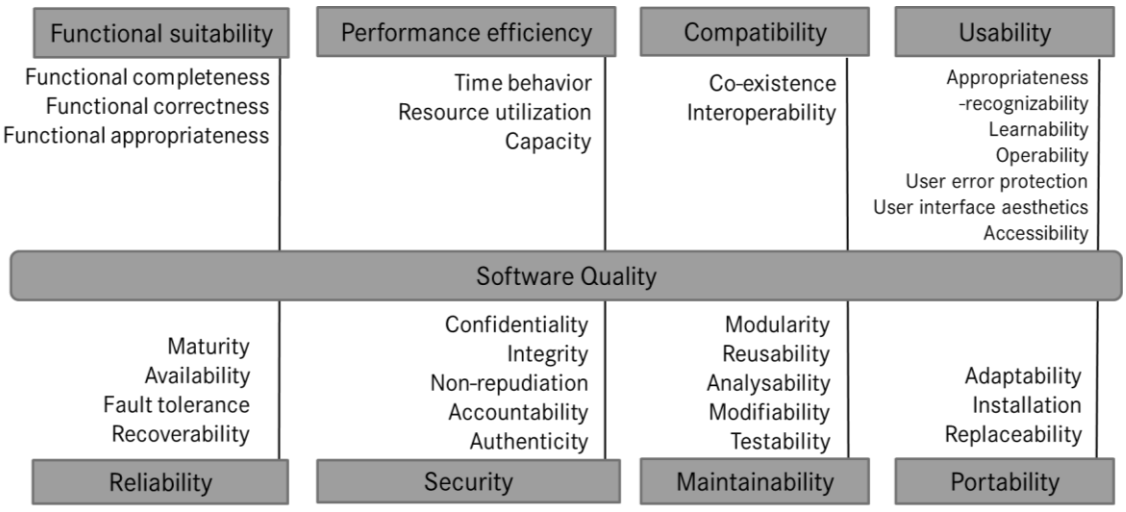
Based on the literature review about CI process from Duvall et al, it can be concluded that, by performing CI process, the common risks on the agile project can be reduced, thus leading to better confidence and communication between stakeholders. Also, the repeatable process in CI can help reduce risks throughout the whole development lifecycle [2].

To research the effects of CI process, the features mentioned above are used as the foundation. According to the understanding and analysis from the researcher, following effects of CI process will be researched, whether they have been improved through CI process. These effects are Software quality, Communication, Development process, Risk management and their sub characteristics.

2.4.1 Effects of CI process on Software quality

The sub characteristics of Software quality have been chosen according to the product quality models from ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE), in which the product quality properties are categorized into eight characteristics: functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability and portability [11].

In this study, the main aspect of CI process is software quality. The sub aspects of software quality are related to these eight characteristics in the quality model, namely Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability and Portability. The sub characteristics of these eight characteristics will also be studied to analyze software quality. The relationship between software quality and its sub aspects is presented with the Figure 3.



(Source: Own illustration)

Figure 3: Software quality and its sub aspects

These 8 sub aspects of software quality will be shortly introduced in this section. More explanation about the sub aspects and the introduction about the sub characteristic from

each sub aspect can be found in ISO/IEC 25010:2011 Systems and software engineering.

Functional suitability means, to which degree the functions provided by a product or system meet stated and implied, when the product or system is used under specified conditions [11]. Functional suitability has 3 sub characteristics: Functional completeness, Functional correctness and Functional appropriateness.

Performance efficiency means how many resources have been used under stated conditions [11]. It has 3 sub characteristics: time behavior, resource utilization and capacity.

Compatibility means, to which degree, a product can perform its required functions efficiently while sharing a common environment and resources with other products and how well the products can exchange and use information [11]. It has 2 sub characteristics: Co-existence and Interoperability.

Usability means, to which degree, a product or system can be used by specified users to achieve specified goals effectively, efficiently and satisfactorily [11]. It has 6 sub characteristics: Appropriateness recognizability, Learnability, Operability, User error protection, User interface aesthetics and Accessibility.

Reliability means, to which degree, a system, product or component performs specified functions under specified conditions for a specified period of time [11]. It has 4 sub characteristics: Maturity, Availability, Fault tolerance and Recoverability

Security means, to which degree, a product or system protects information and data, thus a person or other products or systems have the degree of data access according to their types and levels of authorization [11]. It has 5 sub characteristics: Confidentiality, Integrity, non-Repudiation, Accountability and Authenticity.

Maintainability means to which degree of effectiveness and efficiency, a product or system can be modified [11]. It has 5 sub characteristics: Modularity, Reusability, Analyzability, Modifiability and Testability.

Portability means to which degree of effectiveness and efficiency, a system, product or component can be transferred from one hardware, software or other operational or usage environment to another [11]. It has 3 sub characteristics: Adaptability, Installability and Replaceability.

2.4.2 Additional effects of CI process

Effect of CI process on Software quality is the key point of this study. During the research phase, additional effects of CI process on Communication, Development process, and Risk management have been detected. In order to fully and profoundly research the

effects of CI process during agile development process, these additional effects will be researched together with software quality in this study.

As the study moves on, it has been investigated that, it is necessary to dived some of the additional effects into sub aspects. Based on the literature review and the analysis from the researcher, the sub aspects of the three additional effects of CI process have been defined. Communication is dived into Communication in development team and Communication with stakeholders. Development process consists of Time to run build/Test, Time for development and Cost for development. Risk management is supplemented with rate of refactoring, but not necessary.

3 Research Approach

This section introduces the purpose and approach of this study. First, the research purpose and the research questions are presented. Next, the methodology and the approach for collecting data are introduced.

3.1 Research Purpose

The cooperation partner of this study is the new agile project PBK PT from Daimler IT-department, who would like to get scientifically founded information about agile development processes, agile practice – CI process, effects of CI process, and especially effects of CI process on improving software quality.

Based on the intention of the cooperation partner and the goal of the academic research, the purpose of this study has been set. The purpose of this study is to identify how to build continuous integration in the agile development process, the challenges to build and the effects of continuous integration to the agile development process. Further, how to optimize CI process to improve software quality will also be researched and validated.

3.2 Research Questions

Currently, little research has been done about CI process in agile development practice. Duvall et al [2] introduced the construction of CI process. But how CI process has been built in the practice has been little researched in the previous research. In the software development practice, how to build a suitable and effective CI process is not well known to some agile projects.

To the author's best knowledge even less research has been done about effects of CI process. Duvall et al [2] stated that, with CI the software quality can be improved and the risk can be reduced. But little research has been done about effects of CI process on Software quality, Communication, Development process and Risk management.

The software development practice world has high demands on improving software quality. But what effects has CI process to software quality, how to realize the positive

effects on software quality with CI process is not well known to the practitioners. As a result, following research questions are formulated:

RQ1: How is CI built in the agile development processes?

RQ2: What are the challenges to build CI in agile development processes?

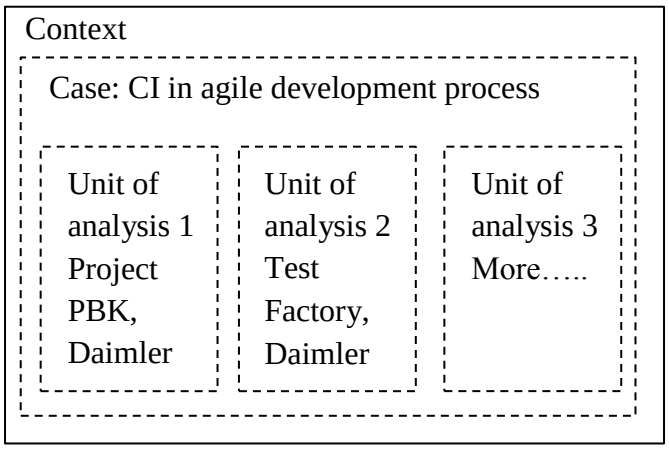
RQ3: What effects has CI to agile development processes?

RQ4: How to optimize CI process?

RQ5: To validate the optimization of CI process in the practice.

3.3 Research Methodology

This thesis reports a 6 month (April 2016 – September 2016) case study. In this case study, the researcher examined a number of distinct activities, for example, the evaluations for CI in several agile development processes. As each of the projects are distinct from the others. Therefore each of these activities could constitute a unit of analysis itself. As a result, an embedded case study design is more appropriate. The case for this case study is CI in agile development process. For this case, multiple units of analysis are selected. The following figure displays the case structure for the case study.



(Source: Own illustration)

Figure 4: Case structure for the case study

The multiple units of analysis for this case study consist of various projects and organizations. The agile development processes involved in the study was selected based on existing relationships with the researcher. Units of analysis in this case study are selected based on availability in Extreme/deviant and Maximum variation [9].

The case study has been performed in the IT department for logistical processes in the Powertrain Area of Mercedes Benz Passenger cars. The department is responsible for

the logistical and logistic planning processes for ordering, producing and delivering engines, transmission and axles from German and international Daimler powertrain plants e.g. Untertürkheim, Berlin, Hamburg and Kölleda (Germany), Sebes (Romania), Beijing (China) to the worldwide Mercedes assembly plants e.g. plant Sindelfingen, Bremen and Rastatt (Germany), Tuscaloosa (USA), Beijing (China) or East London (South Africa). The responsibility of the department consists of:

- planning the processes with logistical departments;
- planning and developing the systems according to the requirements of the Business partners;
- implementing and introducing the new systems;
- maintaining the systems and SW-solutions.

The past projects in the department have been implemented with classical project approaches. But in the last year agile projects have been initiated for local and international processes and solutions, hoping to achieve faster project progress, better and more appropriate specifications and a better quality of the implemented software solutions.

During the project implementation, Daimler adopts most solutions based on standard software, e.g. SAP products, to reduce the time for development from specification to implementation. Since the most development has been done by external partner companies, the development effort from Daimler side can be greatly reduced.

Project PBK is the first agile project in this department, which intends to implement a new SW-Tool for the planning of the assembly program of powertrain products, e.g. engines, transmissions and axles. The plan of the assembly program is executed depending on the demands of the Mercedes-Benz Car assembly plants in the powertrain production network. This project has started at the beginning of 2016 and will be completed in 2018.

The case study was focusing on project PBK at the beginning. As the study moves on, it was necessary to add cases depending on practical constraint the objectives of the research. Therefore, more cases in agile development processes are added into the case study. Backup and contingency cases are also prepared in the case study. The unit of analysis from NTT Data is prepared for Backup. All projects or organizations are listed in Table 4.

Table 4: Projects and organizations for units of analysis

Project/ Organization Name	**Number of Person**	**Tasks/ Developed product**	**Reason of selection**
Project PBK PT,	About 90	Development of a	Original cooperation

Daimler AG		planning tool	partner to this thesis
Test Factory, Daimler AG	About 10	Testing service	Implemented CI process
Fleetboard GmbH	Over 30	Service for Daimler Trucks	Professional CI process
Scrum Event GmbH	Over 200	Development of Financial product	Professional CI process
NTT Data GmbH	Less than 10	Software Service for Daimler	As Backup

The goal of project PBK (program, demand and capacity) Powertrain is to realize automatically update of demands and capacities with other programs in real time, in order to create an efficient and reliable program plan for Powertrain. This is the first project developed with agile development process in this filed in Daimler. This project has cooperated with SAP and implemented on the basis of SAP Hana in the cloud, in which the product SAP IBP will be configured und further developed from needs of Daimler.

The ITM/S test factory located at Service Delivery Center (SDC) India currently supports ~20 projects with various testing needs (for e.g.: test automation, regression testing, functional testing, and integration testing etc.). Since 2014, test factory has gained the experience of providing testing services for projects running in agile methodology. ITM/S test factory has made remarkable progress in providing high quality and cost-effective testing service.

Daimler FleetBoard GmbH is a 100% subsidiary of Daimler AG, which provides Internet-based telematics services for modern fleet management. Since launch in 2000, FleetBoard has equipped more than 150,000 trucks with more than 4,000 customers worldwide. FleetBoard has over 7 years' experience with CI and has built profession CI process in the agile development process. Over 50 Jenkins is running every evening. 50 slaves are controlled by 1 master.

Scrum Event GmbH provides its customers Trainings, Certification and Consulting & Coaching for Scrum. 2 Scrum coaches have been invited by project PBK, in order to improve the agile development process in project PBK. Both coaches have very good experience in agile development process. So interviews have been organized to study CI process from their experience. One standard CI process from a third company developing financial product has been investigated. In addition, opinions about building CI process in project PBK have also been researched.

NTT DATA GmbH focuses on developing IT solution in different industries. One of the biggest customers is Daimler AG. To deliver high quality IT solutions to the customer efficiently, NTT Data has implemented professional CI process in the agile development process.

Research Strategy: Exploratory

In this case study, there is an absence of established theory. Duvall et al have described standard CI process in "Continuous Integration Improving Software Quality and Reducing Risk". But in the practice, CI processes are built based on the individual requirements. The researcher begins with asking like "How is your CI process?" as a prelude. After that, questions like "What effects have CI to your projects" would be asked to research the effects of CI in the practice. Based on the answers for prelude questions, hypotheses and advice for further optimization would be afforded from the researcher. Therefore this thesis has chosen the purpose of *Exploratory* as the research strategy, which finds out what is happening, seeking new insights, and generating ideas and hypotheses for new research [9].

Research Category: Inductive Enquiries

As described, the theory of this case study is induced from the observations. The researcher first observes with an open mind, identifies patterns in the observations, sets up tentative hypotheses, and finally relates them to existing theory or develops new theory [9]. Therefore this case study is an *inductive* research.

Triangulation

With triangulation multiple perspectives towards the studied object can be collected, a broader picture of the studied object can be provided. Therefore the precision of the research has been increased and the validity of empirical research has been strengthened. There are four different types of triangulation:

Data (Source) Triangulation: More than one data source are used, or same data are collected at different occasions. *Observer Triangulation*: there is more than one observer in the study. *Methodological Triangulation*: different types of data collection. *Theory Triangulation*: alternative theories or viewpoints are being used [25].

This research has implemented Data (Source) Triangulation, Methodological Triangulation, and Theory Triangulation. In Data (Source) Triangulation, multiple interviewees are interviewed. In Methodological Triangulation, both qualitative and quantitative methods are being used. In Theory Triangulation, as an alternative theory, CI Process with variations is beneficial to be built in the agile development process.

Replication

A case study can be literally replicated, in which the case is selected to predict similar results. In theoretically replication, the case is selected to predict contrasting results for predictable reasons [10]. In this case study, the most units of analysis have implemented standard CI process. Therefore, the most replication cases are focusing on finding similar results, confirming earlier findings, which is kind of literal replication. But several units of analysis implemented CI process with variations. For example, different with the unit of analysis, which are focusing on development with a lot of coding tasks, Project PBK is focusing more on the requirements analysis and configuration of standard software. Hence, the build of CI process is different with the most standard CI process, which is kind of theoretical replication. Therefore this case study has been both literally and theoretically replicated.

3.4 Research Approach for data collection

In this case study, the researcher intends to gather information from software engineers. So interview and questionnaire surveys are used as direct methods of data collection, where the researcher is in direct contact with the interviewees and collect data in real-time [9]. Data has been collected with interview protocol analysis and questionnaire answers.

Therefore, collected data in this case study are combinations of both quantitative and qualitative data, which is called mixed methods [12]. Quantitative data is from the questionnaire, which involves numbers and brief answers. Qualitative data is from an interview, which involves words, descriptions, pictures, diagrams. Questionnaire questions are closed, while interview questions are open.

This thesis is in a flexible design process, key parameters of the study have been changed during the course of the study.

The interview was conducted in the semi-structured way, in which questions are planned but not necessarily asked in the same order as they are listed. Semi-structured interviews allow for improvisation and exploration of the issues raised in the conversation, which is common in software engineering case studies [9].

There are three general principles of asking questions: Funnel model begins with open questions and moves toward more specific questions. Pyramid model begins with specific ones and opens the questions during the course of the interview. Time-glass model begins with open questions, straightens the structure in the middle, and opens up again toward the end of the interview [9]. In this case study, questions in the interview are asked in the time-glass model. For example:

- Open questions: How is your CI process?
- Close questions: Do you have VCR, CI Server, and feedback framework?
- Open questions: How do you think of your CI process?

Interviews are taken by the only one researcher. Most interviews are taken in the telephone conference. After an interview, the protocol is written to record the interview. The archival data for quality metrics from test factory is also used as data source. Table 5 has been created as a summary of all above mentioned research approaches.

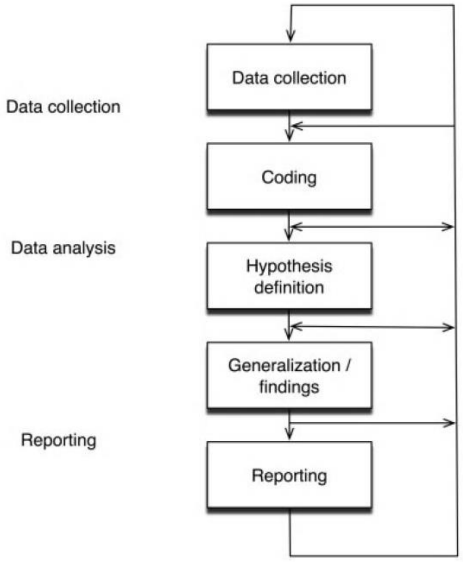
Table 5: Summarize of Research Approach

Research strategy		Exploratory
Research category		Inductive
Triangulation	Data (Source)	multiple interviewees may be interviewed
	Methodological	Using qualitative(interview) and quantitative(questionnaire)methods
	Theory	As alternative, Variations in CI Process and Part of CI process are beneficial to be built in the agile development process
Replication		literally and theoretically
Data collection	Methods	Direct(interview and questionnaire)
	Data type	Qualitative and quantitative
	Type of questions	Open(Interview) /closed(Questionnaire)
	Key parameters design	Flexible
	Way to conduct interviews	Semi-structured
	Model to ask interview questions	Time-glass model
	Way to take interview questions	Telephone conference

4 Data Analysis Methods

4.1 Data Analysis Procedure

The purpose of data analysis is to identify generalizations from the data in terms of patterns, sequences, relationships and so on [9]. The data analysis is done by working with data in the way of summarizing, coding, categorizing, and commenting it. The data analysis procedure for this case study has followed this structure.



(Source: Runeson et al, 2011)

Figure 5: Main steps of data analysis [9]

The data analysis procedure of specific step in this case study will be described in detail in the following section. Since both quantitative and qualitative data have been collected in the data collection phase, the data analysis procedure will be introduced from two parts: Quantitative Data Analysis Procedure and Qualitative Data Analysis Procedure.

4.1.1 Quantitative Data Analysis Procedure

(1) Collect questionnaire results

Questionnaire results from questionnaire tool "easyfeedback" homepage, have been collected separately for all key parameters.

(2) Coding process for questionnaire results

Step1: To display the data in a way, which can be used to analyze RQs. The responses are manually transferred from the questionnaires into a spreadsheet. Each question has been put into a column heading. And each person's answers have been used as a row. Then each possible answer has been assigned a number or "code". Go through each respondent's questionnaire in turn, filling the numbers or codes. Enter all questionnaire data into a spreadsheet, which has been dived into several sub spreadsheets [19].

Step2: After ensuring all the data has been coded and corrected, calculate how many people selected each response. The "sum of all respondents" and "sum of respondents with CI" have been calculated to take the statistical analysis. Based on coding, a set of bar charts have been set up to give a clearer image of the data analysis.

Step3: Besides the information of "how many people selected each response", the information of "What response has been chosen for every question?" This information can also be coded from the questionnaire results to be displayed in the spreadsheet. In the data analysis phase, the coded data can be displayed in tables or graphs.

The questionnaire results have been dived into 2 sub-groups. One sub-group consists of respondents, who have used CI. Another sub-group consists of respondents, who have never used CI. Therefore the effects of CI can be analyzed rationally based on the answers from respondents, who have the experience with CI. And a comparison between 2 sub-groups can also be constructed.

Step4: Apart from coding the questionnaire results with a coding table, which gives the questionnaire results a number. The coding process can also be done with summarizing the results to build a new table with relative numbers. Later on, the summarized and coded data can be displayed in charts, to give a clear view on the usage situation of different CI tools.

(3) Present coded data in charts

After processing the collected data into tables, several charts are created to give a clear view on each research aspect.

(4) Generations/findings from the tables and charts

It will be described in details in the Section 6 Results.

4.1.2 Qualitative Data Analysis Procedure

In this case study, the case of "CI in the agile development process" has been studied. Seven semi-structured interviews have been conducted. Initial qualitative analysis of the data has already started after a half process with data collection.

The qualitative data analysis is carried out as an iterative process, based on Robson's guidelines [12]. Collected documentations from interviews have been gone through several times, and each time to identify or adjust codes, categories and so on. A grounded theory approach has been applied in the data analysis process, in which three steps for data analysis have been defined by Kasurinen et al. [13]:

- Open coding, categories and their related codes are extracted from the data;
- Axial coding, connections between the categories and codes are identified;
- Selective coding, the core category is identified and described.

Based on these three steps from grounded theory, specific steps in the data analysis process for this case study have been designed as follows:

(1) Data collection

Before starting data analysis, there is a set of data has been collected by the researcher from interviews. After the interviews, the answers to interview questions are transcribed into text as the protocol of interviews.

(2) Coding process

Step1: Collected documentations from interviews have been studied in detail by the researchers. Even if the material has already been studied during interviews, document selection and transcription are still necessary to be read again.

Step2: Initial set of codes have been identified from the goals of the study, research questions, and predefined key parameters [14]. Some codes are further defined on the basis of collected data. These coding words were selected to reflect the relevant part of the interview text to a specific part of the studied phenomenon.

Step3: Read all the texts and mark where the codes fit the contents. All pieces of text that relate to the same code are clustered together, in order to be analyzed in a consistent and systematic way. The qualitative data analysis process is iterative. During this process, it is necessary to consider whether more interviews need to be collected and whether codes and codes must be reconsidered. It is often in the case that, some codes must be reformulated or split into more codes.

In the coding process for this study, some codes have been emerged, because some specific processes are not explicitly addressed in the questionnaire. For example, codes for

"parallel process to CI" and "CI process in wish" have been added to the code list later on.

Step4: The results were sent to the participants who were able to comment on them before the final set of conclusions were formulated. The data analysis is focused on building an explanation of the case, leading to formulate the results.

(3) Hypotheses definition

In Data analysis, hypotheses are dived into two parts: Hypothesis generation and Hypothesis confirmation. Hypothesis generation is intended to find hypotheses from the data. Hypotheses generating techniques consist of constant comparisons and cross-case analysis [14]. Hypothesis confirmation denotes some techniques to confirm that a hypothesis is supported. For example, through analysis of more data, the hypothesis can be confirmed. Triangulation and replication are typical approaches for hypothesis confirmation [14].

(4) Generalizations/findings

During the iterative process, a set of generalizations can be formulated, which is the final result of the research. Base on the coding results, 2 aspects are written by the researcher to analyze the processed data.

- Objective description of the coding results about how many people have the positive opinion, and how many people have the negative opinion.
- Subjective analysis from the researcher including speculation on the causes and effects.

(5) Reporting

Reporting for qualitative data analysis will be done along with quantitative data analysis. This case study is reported inside the structure of this master thesis, which has been introduced in the outline in Section 1.4.

4.2 Data Analysis Techniques

4.2.1 Quantitative Data Analysis Techniques

Following methods have been used to make sense of quantitative data for this study.

Technique 1: Take comparison between sub-groups

In this quantitative analysis process, Sub-group of CI and non-CI have been constructed. So that comparisons between sub-groups can be undertaken later.

Technique 2: Process collected data with numbers

A number is given to each response category. Number “5” is given to the response of “excellent”. If evaluation of an effect of CI is “excellent”, it means this response contributes a high score to this effect. Thus the highest number “5” is given to this response. In the same theory, Number”3” is given to the response of “good”, Number”1” is given to the response of “middle”, which means the respondent holds a neutral view: there is no great influence from CI to this effect. So score”1” is given. “Bad” means, the respondents think, CI has a negative effect to this effect. So a “-1” is given to this response. “kein angaben” means the respondent didn´t answer this question. So that, “0” is given to this response.

Technique 3: Analyze the average

Average has been used very often in the quantitative data analysis. For example, for the evaluation of the effect of CI process.

Technique 4: Analyze causation and correlation

The difference between correlation and causation is: causation is when one factor causes another, while correlation is when two variables move together, but one does not influence or cause the other [19]. In this case study, which process from CI leads to what effects on the project, is the causation.

One effect on the project may consist of the correlation of several CI processes.

Technique 5: Regression analysis

To further examine the relationship between variables in the survey, a regression analysis needs to be performed [19]. Regression analysis is an advanced method of data analysis that allows looking at the relationship between two or more variables, and the influence of one or more independent variables on a dependent variable.

Regression analysis can help to determine if this is indeed the case. In this case study, regression analysis has helped to analysis and find out the answer for “Which process leads to which effect”. This result would be very meaningful to the practitioners. Which effect is required from the agile project, practitioners should build the relative process.

4.2.2 Qualitative Data Analysis Techniques

As a wide range types of qualitative data are collected in the case study. Therefore the researcher must employ various qualitative data analysis techniques to make sense of the data. Based on the data analysis techniques from Yin [10], following qualitative data analysis techniques have been implemented in this case study:

Technique 1: Pattern matching

It contains the comparison between empirically observed pattern and predicate pattern further. In RQ1, the comparison between standard CI process and CI process with variations has been undertaken, which leads to propose for optimizations of CI process.

Technique 2: Explanation building

In this kind of analysis, patterns are identified based on cause–effect relationships and the underlying explanations are sought. This technique has been used widely in the research for in RQ2 and RQ3.

In RQ2, based on the research results from RQ1, a set of hypothesis have been generated to explain the usage situation of CI and the challenges to build CI in the agile development process.

In RQ3, the similar process to RQ2 has been undertaken. Based on the RQ1, the hypotheses about CI effects in the agile development process have been confirmed.

Technique 3: Logic models

Logic models are used to explain more complex series of cause–effect relationships. Theoretical predictions of related events can be compared with empirical observations. In RQ1, the theoretical predictions of CI process consist of version control, CI server, and Feedback. In addition, CI process should be started after every committed change. But in the empirical observations, the CI process has been adjusted to the real developing environment. This finding leads to the further analysis of RQ2 and RQ4.

Technique 4: Cross-case synthesis

In this kind of analysis, firstly, different cases can be compared. Secondly, how to take the analysis in a certain case study will be decided. This technique has been used through the whole case study for all RQs. Because multiple units of analysis has been used in the case study.

4.3 Tools support for Data Analysis

4.3.1 Tool support for Quantitative Data Analysis

Questionnaire tool “easyfeedback” has played an important role in data collection for questionnaire results. In addition, it has also provided an initial thinking for data analysis. After the coding process, more quantitative data analysis has been processed with excel depending on the research goals from the researcher.

4.3.2 Tool support for Qualitative Data Analysis

The qualitative analysis in this case study involves working with the text in the form of interview transcripts and quality assurance documents. Therefore, text-based data analysis tool is necessary. In this case study, qualitative data analysis software - QDA Miner has been used. QDA Miner is an easy-to-use qualitative data analysis software package for coding, annotating, retrieving and analyzing small and large collections of documents and images [20]. QDA Miner qualitative data analysis tool may be used to analyze interviews or focus group transcripts, legal documents, journal articles, speeches, even entire books, as well as drawings, photographs, paintings, and other types of visual documents. For this case study, QDA Miner has been used for coding, retrieving and analyzing collections of documents from interviews.

5 Case Study

5.1 Case Study Design

This case study has been conducted in both questionnaire and interview, in order to research the four research questions which are mentioned in Section 3.2 Research Questions. The questions asked during the case study are called Case Study Questions. They were designed based on the four research questions. Additionally, the general information has also been asked to know the background of the respondents. The Case Study Questions are related to the corresponding codes, which have been collected from interviews or/and questionnaire. All Case Study Questions have been summarized in Table 6.

Table 6: Summarize of Case Study Questions

Category	**Case Study Questions**	**Relative Codes**	**Collection Method**
General information	1. What is your current role in your organization?	About interviewee	Interview
	2. Which product(s) are you currently working on?	About project/ organization	Interview
	3. Could you briefly describe which agile software development methodology your current team has used?	About project/ organization	Interview
	4. Are you familiar with the concept of Continuous Integration? ---IF YES: ---IF NO:	About interviewee	Interview
RQ2: Challenge to	5. Has Continuous Integration been used in your development process?	Usage of CI	Interview & Questionnaire

build CI process	---IF YES: ---IF NO:		
RQ1: CI in agile development process	6. How has the Continuous Integration process been built in the development process?	CI process with variations	Interview & Questionnaire
		Standard CI process	
	7. For what purpose is Continuous Integration being used? ---mention Unit/Component/System test, Regression test, Functional test, Acceptance test	Purpose for CI process	Interview & Questionnaire
	8. What tool is used for Version Control Repository, automated build, automated Deployment, automated test?	Tools for CI process	Interview & Questionnaire
	9. How has feedback been given?	Methods of feedback	Interview & Questionnaire
	10. How often is Continuous Integration being run in the development process?	Frequency of CI process	Interview & Questionnaire
RQ2: Challenge to build CI process	11. (CI being used) Are there any challenges you have faced to build CI?	Difficulty to build CI	Interview
	12. (CI not being used) Do you feel that you and/or your work colleagues should use Continuous Integration? (why/why not)	Usage of CI	Interview
	13. How well does the Continuous Integration process suit the current software product you are working on?	Difficulty to build CI	Interview

RQ3: Effects of CI process on Software quality	14. What kind of effects has Continuous Integration brought to software quality? From the aspects of Functionality, Reliability, Usability, Maintainability, Efficiency, and Portability?	Software Quality	Interview & Questionnaire
RQ3: Effects of CI process on Communication	15. What kind of effects has Continuous Integration brought to Communication? From communication in the development team and communication with stakeholders.	Communication	Interview & Questionnaire
RQ3: Effects of CI process on Development process	16. What kind of effects has Continuous Integration brought to development process? From time to run build/test, time for development, cost for development?	Development process	Interview & Questionnaire
RQ3: Effects of CI process on Risk Management	17. What kind of effects has Continuous Integration brought to reduce risk to the project?	Risk Management	Interview & Questionnaire
RQ4:Optimization of CI process	18. From all CI effects, which effect is especially important for the agile project?	Not coded	Interview
	19. How would you like to implement CI in your development process, to reach this important effect?		Interview
	20. How is the development process with SonarQube in your agile development process?		Interview
RQ5:To validate the optimization of CI process in the practice	21. In comparison with the CI process without Sonar in the past, what are the differences now with Sonar?	Not coded	Interview

5.2 Case Study Implementation

5.2.1 Case Study with Interviews

Seven semi-structured interviews have been conducted. An overview of each interview has been introduced in Table 7.

Table 7: Overview of interviews in the Case Study

Interviews	**Name for Unit of Analysis**	**Interview Participant**			**Date 2016**	**Duration of interview**
		Name	Company	Position		
Interview1	PBK with Daimler	Heiner Lösel	Daimler AG	IT Architect	04.03 11.03 18.03	180min
Interview2	PBK with SAP	Stefan Adam & Ralf Hasemann	SAP AG	Software Developer	20.04	120 min
Interview3	Scrum Event (Marco)	Marco Tomma-sone	Scrum Event GmbH	Agile Coach	18.05	60 min
Interview4	Scrum Event (standard CI)	Confiden-tial	Scrum Event GmbH	Agile Coach	18.05	60 min
Interview5	Daimler Test Factory	Kalyan Boggarapu	Daimler AG	Test Manager	25.05 22.06	120 min
Interview6	Daimler Fleetboard	Markus Hühn	Daimler Fleetborad GmbH	Product Owner	22.06 02.08	120 min
Interview7	NTT Data	Confiden-tial	NTT Data	Software Developer	27.06	30min

Section 3.2 Research Methodology described that, multiple units of analysis has been selected for this case study. In Section 3.2, each researched project was treated as a unit of analysis. During the research, it has been found that one project is developed from more than one development organization. Different development organizations have implemented different CI process. Therefore it is better to treat every interview as a unit

of analysis. Name for unit of Analysis has been listed in the table, which will be used frequently in the following sections, especially in Section 6.

5.2.2 Case Study with Questionnaire

Based on the Case Study Questions from Section 5.1, an online questionnaire has been designed with the questionnaire designing tool "easyfeedback". The link to questionnaire has been published on different public social media, in order to get a wide range of answers. An overview of the general information about the questionnaire is listed in the Table 8.

Table 8: General information about questionnaire in the Case Study

Number of people finished questionnaire	16
Positions of interviewees for questionnaire	Software developer
Duration of questionnaire being published	Since 13.06.2016
End of data collection for Questionnaire	09.08.2016
Ways of questionnaire being published	different agile groups from Xing, Daimler Intranet
Other invitation methods to answer Questionnaire	Separately invitation based personnel contact

5.3 Data Coding for collected data

5.3.1 Coding process for questionnaire results

According to the theory of coding process for questionnaire results in Section 4.1.1 Quantitative data analysis, the collected questionnaire results are coded. In Section 4.2.1, different techniques for quantitative data analysis have been listed, which will be taken into consideration in this section. To take comparison between sub-groups, the questionnaire respondents have been dived into 2 sub-groups: CI and non-CI users. To be mentioned, in the collected data, answers provided by non-CI users have been marked with red color.

The coding process, collected data, and the corresponding coding results from this case study are presented in the following.

Coding step1: Coding questionnaire results with filling the spreadsheet

If the respondents have chosen the choice for the questions, the Code "1" is given to this unit. Otherwise "0" is filled in. After ensuring all the data has been coded and corrected, the number of how many people have selected each response has been counted. The "sum of all respondents" and "sum of respondents with CI" have been calculated to take the statistical analysis. Figure 6, Figure 7 and Figure 8 have been created in this method.

Based on coding results, a set of bar charts will be set up to give a clearer image of data analysis, which will be presented in the Section 6 Results.

Figure 6 is a part of the spreadsheet, which presents the coding results for Usage of CI and CI processes being implemented.

Respondents	Q1: Have you ever used CI		Q2:Do you have the following process (in or without CI)					
	YES	NO	1.Version control	2.CI server	2.1Automated build	2.2Automated deployment	2.3Automated test	3.Feedback Mechanism
Respondent 1	1	0	0	0	1	0	0	0
Respondent 2	1	0	1	1	1	0	1	1
Respondent 3	1	0	1	1	1	1	1	1
Respondent 4	1	0	1	1	1	0	1	1
Respondent 5	0	1	-	-	-	-	-	-
Respondent 6	0	1	1	0	1	0	1	0
Respondent 7	1	0	1	1	1	0	1	0
Respondent 8	1	0	1	0	1	1	1	0
Respondent 9	1	0	1	0	0	1	1	1
Respondent 10	1	0	1	1	1	0	1	1
Respondent 11	1	0	1	1	1	1	0	0
Respondent 12	1	0	1	1	1	1	1	0
Respondent 13	1	0	1	1	1	1	1	1
Respondent 14	1	0	1	1	1	1	1	0
Respondent 15	1	0	1	1	1	0	1	0
Respondent 16	1	0	1	1	1	1	1	0
Implemented process(CI and non-CI)	16	-	14	11	14	8	13	6
Implemented process in CI	14	-	13	11	13	8	12	6

Figure 6: Coding results for Usage of CI and CI processes being implemented

Figure 7 is a part of the spreadsheet, which presents the coding results for Purpose of CI.

Respondents	Q4:Purpose of CI					
	1.1Unit test	1.2Component test	1.3System test	2.Regression test	3.Functional test	4.Acceptance test
Respondent 1	0	1	1	0	0	0
Respondent 2	1	0	1	1	0	0
Respondent 3	1	0	1	1	0	1
Respondent 4	0	1	0	1	0	0
Respondent 5	-	-	-	-	-	-
Respondent 6	1	1	1	1	1	0
Respondent 7	1	1	0	0	1	0
Respondent 8	1	0	0	0	0	0
Respondent 9	0	0	0	1	1	1
Respondent 10	1	1	0	1	1	0
Respondent 11	1	0	0	0	0	0
Respondent 12	1	0	0	0	0	0
Respondent 13	1	1	1	1	1	1
Respondent 14	1	0	1	1	1	0
Respondent 15	1	0	1	0	0	0
Respondent 16	1	0	0	1	1	0
Implemented process(CI and non-CI)	12	6	7	9	7	3
Implemented process in CI	11	5	6	8	6	3

Figure 7: Coding results for Purpose of CI

Figure 8 is a part of the spreadsheet, which presents the coding results for Purpose of CI.

Respondents	Q5:frequency of CI			
	after every commit change	≥1 time per day	≥1 time per week	it depends
Respondent 1	0	0	0	1
Respondent 2	1	0	0	
Respondent 3	1	0	0	0
Respondent 4	0	0	1	0
Respondent 5	-	-	-	-
Respondent 6	0	0	0	1
Respondent 7	0	0	1	0
Respondent 8	1	0	0	0
Respondent 9	0	0	1	0
Respondent 10	1	1	0	1
Respondent 11	1	0	0	0
Respondent 12	0	0	0	0
Respondent 13	1			
Respondent 14	0	0	0	1
Respondent 15	1	0	0	0
Respondent 16	0	0	1	1
Implemented process(CI and non-CI)	7	1	4	5
Implemented process in CI	7	1	4	4

Figure 8: Coding results for Frequency of CI

Coding step2: Coding questionnaire results with coding table

As described in the techniques for quantitative data analysis in Section 4.2.1, the collected questionnaire results for all four effects of CI process will be processed with a coding table from Table 9. The collected questionnaire results, coding table, and the coding results will be presented in the following.

Figure 9 displays the collected questionnaire results for Software quality.

Repondents	1.Improve Software quality assurance	1.1Functionality assurance	1.2Increase Reliability	1.3Increase Usability	1.4Increase Maintainability	1.5Increase Efficiency	1.6Increase Portability	1.7Reduce erorr rate
Respondent 1	good	good	good	middle	kein angaben	kein angaben	kein angaben	excelent
Respondent 2	middle	good	good	kein angaben	kein angaben	kein angaben	kein angaben	good
Respondent 3	middle	good	middle	middle	kein angaben	kein angaben	kein angaben	good
Respondent 4	good	good	excelent	middle	middle	middle	kein angaben	excelent
Respondent 5	-	-	-	-	-	-	-	-
Respondent 6	excelent	excelent	good	good	excelent	middle	middle	good
Respondent 7	good	good	good	middle	good	good	middle	excelent
Respondent 8	excelent	excelent	excelent	excelent	excelent	excelent	excelent	bad
Respondent 9	good	good	excelent	middle	good	excelent	good	good
Respondent 10	good	good	good	good	good	good	good	good
Respondent 11	excelent	excelent	excelent	excelent	excelent	excelent	excelent	excelent
Respondent 12	excelent	good	good	kein angaben	good	middle	middle	good
Respondent 13	good	good	middle	good	good	excelent	middle	excelent
Respondent 14	good	good	good	middle	middle	middle	middle	good
Respondent 15	good	good	good	good	excelent	excelent	middle	excelent
Respondent 16	good	good	excelent	middle	good	good	kein angaben	excelent

Figure 9: Collected quantitative data for evaluation of Software quality

Figure 10 displays the collected questionnaire results for Communication, Development process, and Risk management.

Repondents	2.Improve Communication	3.Improve Cooperation	4.Improve Development process	4.1Reduce Time to run build	4.2Reduce Time for development	4.3Reduce Cost for development	5.Reducing risk	5.1Reduce rate of refactoring
Respondent 1	excelent	excelent	excelent	excelent	middle	good	excelent	middle
Respondent 2	bad	bad	good	bad	middle	middle	middle	bad
Respondent 3	good	excelent	excelent	good	excelent	bad	good	middle
Respondent 4	good	good	good	good	good	middle	excelent	good
Respondent 5	-	-	-	-	-	-	-	-
Respondent 6	excelent	good	excelent	good	good	good	good	good
Respondent 7	good	excelent	good	excelent	good	good	good	middle
Respondent 8	bad	excelent	excelent	bad	excelent	excelent	excelent	excelent
Respondent 9	middle	middle	middle	excelent	middle	good	excelent	middle
Respondent 10	kein angaben	kein angaben	good	good	good	good	good	good
Respondent 11	good	excelent	excelent	excelent	middle	middle	excelent	middle
Respondent 12	kein angaben	kein angaben	good	excelent	middle	middle	good	bad
Respondent 13	good	excelent	good	excelent	excelent	good	excelent	middle
Respondent 14	middle	middle	good	good	middle	middle	kein angaben	kein angaben
Respondent 15	good	good	good	excelent	excelent	excelent	middle	middle
Respondent 16	good	good	good	bad	excelent	excelent	good	good

Figure 10: Collected quantitative data for evaluation of rest CI effects

Table 9 is the coding table for effects of CI. With the coding table, the evaluation on the effects of CI process will be replaced with a code number. Thus, the evaluations on the effects of CI process can be presented in bar charts in the later Section, to give a much clearer view of the Research Questions.

Table 9: Coding table for effects of CI

Questionnaire choice	**Code for choice**
excellent	5
good	3
middle	1
bad	-1
kein angaben	0

After coding the questionnaire results with the coding table in Table 9, the coding results for effects of CI on Software quality is displayed in Figure 11.

Respondents	1.Improve Software quality assurance	1.1Functionality assurance	1.2Increase Reliability	1.3Increase Usability	1.4Increase Maintainability	1.5Increase Efficiency	1.6Increase Portability	1.7Reduce erorr rate
Respondent 1	3	3	3	1	0	0	0	5
Respondent 2	1	3	3	0	0	0	0	3
Respondent 3	1	3	1	1	0	0	0	3
Respondent 4	3	3	5	1	1	1	0	5
Respondent 5	-	-	-	-	-	-	-	-
Respondent 6	5	5	3	3	5	1	1	3
Respondent 7	3	3	3	1	3	3	1	5
Respondent 8	5	5	5	5	5	5	5	-1
Respondent 9	3	3	5	1	3	5	3	3
Respondent 10	3	3	3	3	3	3	3	3
Respondent 11	5	5	5	5	5	5	5	5
Respondent 12	5	3	3	0	3	1	1	3
Respondent 13	3	3	1	3	3	5	1	5
Respondent 14	3	3	3	1	1	1	1	3
Respondent 15	3	3	3	3	5	5	1	5
Respondent 16	3	3	5	1	3	3	0	5
Sum of all repondents	49	51	51	29	40	38	22	55
Sum respondents with CI	44	46	48	26	35	37	21	52
Average of respondents with CI	3,1	3,3	3,4	1,9	2,5	2,6	1,5	3,7

Figure 11: Coding results for evaluation of Software quality

Similarly, after coding the questionnaire results with the coding table in Table 9, the coding results for effects of CI on Communication, Development process, and Risk management is displayed in Figure 12.

Respondents	2.Improve Communication	3.Improve Cooperation	4.Improve Development process	4.1Reduce Time to run build	4.2Reduce Time for development	4.3Reduce cost for development	5.Reducing risk	5.1Reduce rate of refactoring
Respondent 1	5	5	5	5	1	3	5	1
Respondent 2	-1	-1	3	-1	1	1	1	-1
Respondent 3	3	5	5	3	5	-1	3	1
Respondent 4	3	3	3	3	3	1	5	3
Respondent 5	-	-	-	-	-	-	-	-
Respondent 6	5	3	5	3	3	3	3	3
Respondent 7	3	5	3	5	3	3	3	1
Respondent 8	-1	5	5	-1	5	5	5	5
Respondent 9	1	1	1	5	1	3	5	1
Respondent 10	0	0	3	3	3	3	3	3
Respondent 11	3	5	5	5	1	1	5	1
Respondent 12	0	0	3	5	1	1	3	-1
Respondent 13	3	5	3	5	5	3	5	1
Respondent 14	1	1	3	3	1	1	0	0
Respondent 15	3	3	3	5	5	5	1	1
Respondent 16	3	3	3	-1	5	5	3	3
Sum of all repondents	31	43	53	47	43	37	50	22
Sum respondents with CI	26	40	48	44	40	34	47	19
Average of respondents with CI	1,9	2,9	3,4	3,1	2,9	2,4	3,4	1,4

Figure 12: Coding results for evaluation of rest CI effects

Coding step3: Coding the collected questionnaire results in a way of summarizing the results, in order to build a new table with counted number.

Figure 13 presents the collected data for Tool for CI process.

Respondents	Q3: Tool for process in Q2					
	1.Version control	2.CI server	2.1Automated build	2.2Automated	2.3Automated test	3.Feedback Mechanism
Respondent 1	-	-	Jenkins	-	-	
Respondent 2	Git	Jenkins	Maven	-	Junit	Email
Respondent 3	Bitbucket	Jenkins,Bamboo	Jenkins,Bamboo	Jenkins, Bamboo	Junit,Jenkins,Bamboo	Extreme Feedback
Respondent 4	Subversion	Jenkins	Jenkins	-	Jenkins	Jira
Respondent 5	-	-	-	-	-	-
Respondent 6	clear case	-	?	-	Jenkins	-
Respondent 7	SCM Manager,	Jenkins	Maven, Docker	-	Maven	-
Respondent 8	Github	TravisCI	gradle	ansible	gradle	-
Respondent 9	jazz RTC			ANT	Selenium	Sharepoint
Respondent 10	Git	Jenkins	-	-	Junit,Test Rail,Squish	RTC
Respondent 11	Team Foundation	Team City	Team City	Octopus	-	-
Respondent 12	Subversion	Jenkins	Jenkins	Jenkins	Jenkins	-
Respondent 13	TFS,GIT+Gerrit	Jenkins TFS	Jenkins TFS	Jenkins TFS	Selenium,Java, C#	Homemade
Respondent 14	Git	Jenkins	Maven, Gradle	Jenkins, Nexus	JUnit, SonarQube, Appium-	
Respondent 15	Git Svn Cvs	Gerrit	Jenkins	-	Jenkins	-
Respondent 16	Team Foundation	Team Foundation	Team Foundatio	Custom tools	MSTest, OpenCover	-

Figure 13: Collected data for Tool for CI process

Based on this table with the tools from all respondents, the answers from respondents with CI experience have been selected out. A table for the usage of CI server and its processes has been created. Figure 14 presents the coding results for Tool for CI process. In the same, Figure 15 and Figure 16 display the coding results for the tool for Version control and Feedback framework.

Tool for CI process	CI server	Automated build	Automated deployment	Automated test
Jenkins	8	6	4	5
Maven	0	3	0	1
Bamboo	1	1	1	1
Docker	0	1	0	0
Ansible	0	0	1	0
Team City	1	1	0	0
TravisCI	1	0	0	0
TFS	2	2	1	0
gradle	0	2	0	1
Ant	0	0	1	0
Octopus	0	0	1	0
Junit	0	0	0	4
selenium	0	0	0	2
Test Rail	0	0	0	1
Squish	0	0	0	1
Gerrit	1	0	0	0
Nexus	0	0	1	0
Custom tools	0	0	1	0
SonarQube	0	0	0	1
MSTest, OpenCover	0	0	0	1
Appium-	0	0	0	1
Number of implemented in CI	11	13	8	12

Figure 14: Coding results for Tool for CI server

Tool for version control	Number of Usage
Git	5
Bitbucket	1
SVN	3
Github	2
Gitlab	1
Jazz RTC	1
TFS	3
GIT+Gerrit	1
CVS	1
number of VCR	13

Figure 15: Coding results for Tool for version control

Feedback framework	Number of Usage
Self developed tool	2
Email	1
Jira	1
Jazz RTC	1
Sharepoint query	1

Figure 16: Coding results for Feedback framework

Figure 17 shows the collected questionnaire results about, what CI process leads to the corresponding CI effect.

	Which process brings this effect																
	1.Improve software quality assurance	1.1Functionality assurance	1.2Increase Reliability	1.3Increase Usability	1.4Increase maintainability	1.5Increase efficiency	1.6Increase portability	1.7Reduce erorr rate	2.Improve communication	3.Improve cooperation	4.Improve development process	4.1Reduce time to run build	4.2Reduce time for development	4.3Reduce cost for development	5.Reducing risk	5.1Reduce rate of refactoring
Respondent 1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 4	Automated Test	Automated Test	Automated Test	NO	-	-	-	Automated Test	Version control/Feedback System	Version control/Feedback System	Automated Test	Automated Build	Automated Test	NO	Automated Test	Automated Test
Respondent 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 7	Automated Testing	Automated Testing	Automated Testing	Automated Testing	-	-	-	Automated Testing	Version Control, Automated	Version Control	all	CI Server: Automated Build,	all	all	Automated Testing	Automated Testing
Respondent 8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 9	Improvement mainly due to automatic testing	faster customer feedback, shorter sprints	Regression testing (Less errors "this used to work and	No real improvement	-	-	-	reduced rate in regression, but no improvement in new	No real improvement	No real improvement	multiple software partners. no real improvemen	automatic build reduced packaging by 4hours each	No real improvement	reduced cost due to faster builds not from development	More regression tests -> better software	No real improvement
Respondent 10	CI Server	CI Server,VCS	CI Server,VCS	CI Server,VCS	-	-	-	CI	CI	CI	CI	Auto Build	automation processes	automation processes	CI,Feedback mechanism	CI
Respondent 11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Respondent 13	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban	agile kaban
Respondent 14	Automated test	Automated test	Automated test	Automated deployment	-	-	-	Automated test	CI server, Automated deployment	CI server, Automated deployment	CI server, Automated deployment	Automated test	Automated test	Automated test	-	-
Respondent 15	Jenkins(CI server)	Jenkins	Jenkins(VCR +CI Server)	Jenkins	-	-	-	Jenkins	Gerrit	Git	Jenkins	Jenkins	Jenkins	Jenkins	Gerrit + Jenkins	Gerrit
Respondent 16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 17: Collected data for CI process and CI effect

After the similar coding process like for the CI tools, coding results for "What CI process leads to the corresponding CI effect" have been presented in Figure 18.

CI process	1.Improve Software quality assurance	1.1Functionality assurance	1.2Increase Reliability	1.3Increase Usability	1.4Increase Maintainability	1.5Increase Efficiency	1.6Increase Portability	1.7Reduce erorr rate	2.Improve Communication	3.Improve Cooperation	4.Improve Development process	4.1Reduce Time to run build	4.2Reduce Time for development	4.3Reduce Cost for development	5.Reducing risk	5.1Reduce rate of refactoring
VCR	0	1	2	1	-	-	-		3	3					1	1
CI server	2	2	3	2	-	-	-	3	2	2	3	2	1	1	3	1
Automated test	4	3	3	1	-	-	-	2	0	0	1	1	2	1	2	2
Automated build	0	0	0	0	-	-	-	0	0	0	0	4		1	0	0
Automated process	0	0	0	0	-	-	-	0	0	0	0	0	1	1	0	0
Automated deployment	0	0	0	1	-	-	-	0	1	1	1	0	0	0	0	0
Feedback fram work	0	1	0	0	-	-	-		1	1					1	0
Agile method(Kanban)	1	1	1	1	-	-	-	1	1	1	1	1	1	1	1	1

Figure 18: Coding results for CI process and CI effect

For "What CI process leads to the corresponding CI effect", the main aspects of CI effects is with more significance to be researched. The number from sub aspects has been summed to the main aspects. After further coding process, a table with only 4 main aspects of CI processes has been created. Figure 19 presents the further coding results.

CI process	Software quality	Communication	Development process	Risk management
VCR	4	6		2
CI server	12	4	7	4
Automated test	13	0	5	4
Automated build	0	0	5	0
Automated process	0	0	2	0
Automated deployment	1	2	1	0
Feedback framework	1	2	0	1

Figure 19: Further coding results for CI process and CI effect

5.3.2 Coding process for interview results

Coding step1: Upload the collected interview results to coding tool – QDA Miner

Coding step2: Construct categories and codes in QDA Miner

For this study, the initial codes were put into four categories, which based on the general information and three Research Questions.

- Category1: General information
- Category2: RQ1, CI in agile development process
- Category3: RQ2, Challenge to build CI process
- Category4: RQ3, Effects of CI process

RQ3 was divided into 4 sub-research questions. Therefore Category3 was also accordingly divided into 4 sub-categories containing sub-themes. Each sub-theme contained all related codes.

- Category3.1: Software Quality
- Category3.2: Communication
- Category3.3: Development process
- Category3.4: Risk Management

RQ4, Optimization of CI process and RQ5, to validate the optimization of CI process in the practice, should belong to category 4 and category5. But they have been directly analyzed from the collected data, instead of being coded first.

Finally, following categories and codes in Figure 20 are defined in the coding process:

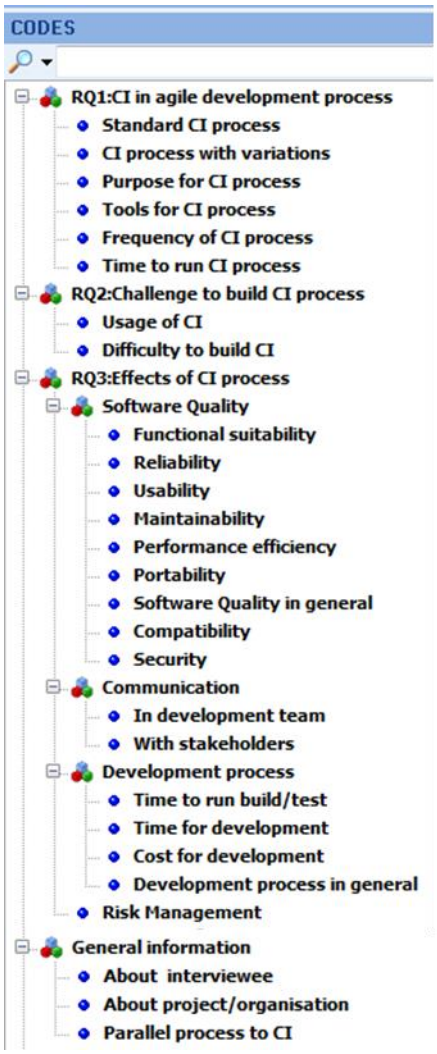


Figure 20: Overview of the codes in QDA Miner

Coding Step3: Interview results are coded to the correspond categories and codes

Some interview results indicate clear, which codes they belong to. In this study, it is called “Direct coding”. But some collected interview results need to be analyzed first, and then can be coded to the relative codes, which called “Indirect coding” in this study.

(1) Direct coding

For instance: Category “General information” contains the codes like “about interviewee” and “about project and organization” and “parallel process to CI”, which introduces the background of the unit of analysis.

Sentences like “Every 4 weeks. It depends on the sprint cycle. It will be tested at the end of one sprint.” and “Stage2: Everyday.” could be classified with the Frequency of CI

process code. This code belongs to the category RQ1. So the sentences from this code can be analyzed to answer RQ1.

Some codes can be combined to analyze one research question. For RQ2: What are the challenges to build CI in agile development processes? In this case, all text related to codes like “usage of CI”, ” Difficulty of CI” and “CI process variations” are combined to analyze RQ2.

(2) Indirect coding

Collected data for coding is not only sentence with obvious explanations to RQs, but also quality metrics and key performance indexes (KPIs). For example, the data collected from test factory to analyze RQ3 are quality metrics and KPIs. This kind of data needs to be interpreted to the main aspects and sub aspects of CI effects firstly, and then be coded to the relative CI effects, namely the categories and codes defined from the researcher for this case study.

In the following, the interpretation from KPIs and quality metrics to 4 effect aspects is described in details:

Firstly, Table 10 contains KPIs and Quality Metrics which are relevant to CI is created.

Table 10: KPIs and quality metrics from test factory

KPIs	**Formula**	**Target State**
Defect Identification Efficiency (%)	Total number of valid defects found by test team /(Total valid defects logged in the project life.cycle + Number of defects identified by test team but rejected by the customer)	90%
% Automation	Number of Automated regression test cases / Total number of regression test cases	>=50%
Effort Variance	(Actual Effort- estimated effort)/Estimated Effort	<=10%
Schedule Variance	(Actual End date-Planned End date)/(Planned End date-Planned Start date)	<=10%
Cost Variance	(Actual Cost-Estimated Cost)/Estimated	<=10%

	Cost	
Quality Metrics	**Formula**	**Target State**
Test Coverage	Total number of testable requirements mapped to test cases/Total number of baselined testable requirements	100%
Defect Leakage(UAT)	Defects identified in UAT/Total defects	4%-7%
Test execution productivity	Number of test cases executed/Effort in Person days	From 5-7 Test cases (manual) To 15-20 Test cases (automated)
Requirement stability	Number of Change Requests/Total number of requirements	10%

Secondly, the KPIs and quality metrics are analyzed one by one, in order to be mapped to the main and sub aspects of the CI effects. During the analysis, the quality model in the Section 2.4.1 is being used to analyze the sub aspects of software quality aspects. The additional effects of CI process are the basis to analyze Communication, Development process and risk management. After the analysis, KPIs and quality metrics are mapped to categories and codes defined in Figure 20 in the Section 5.3.2.

KPI 1: Defect Identification Efficiency

Table 11 presents the evaluation from Test factory on the KPI of Defect Identification Efficiency, which is the basis of analysis about effects of CI process.

Table 11: Evaluation on Defect Identification Efficiency

Unit of Analysis	**Evaluation text from the interviewee**
Daimler Test Factory	Defect identification will be improved through CI. In the unit test, after each commit, CI will be triggered. Every successful unit test will also be tested in automatically regression test. And for every test in automatically regression test, a relevant test case is designed. So that, the Defect identification can be as high as possible with CI, almost 90%.

Reliability

- **Recoverability:** The test factory has mentioned the unit test in CI process, which has great effect to improve Defect Identification Efficiency. In fact, after private build from developer and before unit test, the developer uploads the newest version to the VCR tool. The software version is controlled with a new version number. CI server will call up this new code piece and test with existed parts. If the uploaded code part has an error or even causes system crash, it is easy to roll back to the older software version. Therefore, with CI the recoverability of software is greatly increased.
- **Maturity:** Based on the statement from test factory, after private build, the newly developed code part has to pass through Unit test, Component Test, automated Regression test and Functional test. The frequency of software failure is decreased with CI. The statement from test factory also indicated that the defects found in the user acceptance test have been greatly decreased.

Maintainability

- **Modularity:** In CI process, the test phases have been dived into different stages. After private build, the newly developed code part has to pass through Unit test, Component Test, automated Regression test and Functional test stage by stage. During the unit test, a set of classes will be tested in a module. In the component test, the connection between different modules will be tested in a component and so on. As the test is running from unit to unit, module to module, a change to one component has minimal impact on other components. Therefore the modularity of the software is highly improved.
- **Analysability:** From the statement of test factory, the Defect identification is very high, almost 90%. It means, with CI, the test team has identified the most real defects. It indicates that, with CI the test team can diagnose a product for deficiencies or causes of failures, or to identify parts to be modified in a right way. CI has provided good mechanisms for the product or system to analyze its faults and provide reports to a failure.
- **Modifiability:** Based on the statement from test factory, with the automated Regression test from CI process, the development of software can be effectively and efficiently modified without introducing defects or degrading existing product quality.

Besides, Modularity and Analyzability can influence modifiability.

Functional suitability

- **Functional completeness and Functional correctness:** From the statement of test factory, "the Defect identification can be as high as possible with CI, almost 90%." It indicates that with CI over 90% real defects will be found. Functional completeness and Functional correctness of the software can be assured.

Risk management

As over 90% real defects have to be found. The risk of running software has been greatly reduced.

In conclusion, this KPI reflects 2 aspects of CI effects in Table 12.

Table 12: KPI Defect Identification Efficiency mapped to aspects of CI effects

Software Quality			**Risk management**
Reliability -Recoverability -Maturity	**Maintainability** -Modularity -Analysability -Modifiability	**Functionality** -Functional completeness -Functional correctness	Reducing risk

KPI 2: Percentage of Automation

Table 13 presents the evaluation from Test factory on the KPI of Percentage of Automation, which is the basis of analysis about effects of CI process.

Table 13: Evaluation on Percentage of Automation

Unit of Analysis	**Coded text**
Daimler Test Factory	Percent of automation is great improved through CI, because of the automated build /test process in CI. This aspect indicates that; the development process is improved.

Reliability

- **Maturity:** With the automated build /test process in CI, it is easier, faster and cheaper to test the software repeatedly. The errors and failure of the development will be corrected on time. In the end, the frequency of failure has been greatly reduced.
- **Fault tolerance:** After a large amount of automated test in different stages, the finished product which has passed the entire test will have a high Fault tolerance, namely the software can operate as intended despite the presence of hardware or software faults.

Maintainability

- **Testability:** With the automation build /test process, all established test criteria can be checked efficiently, whether it has been met by the developed software.
- **Adaptability:** After each change on the development, the automation process to check the change will be triggered. Therefore the development of the software can be effectively and efficiently adapted for different hardware, software, operational, and usage environments.

Development process

It is clear that, with the automated process, time to run build/test and cost for development have been decreased. Therefore "the development process is improved."

In conclusion, this KPI reflects 2 aspects of CI effects in Table 14.

Table 14: KPI Percentage of Automation mapped to aspects of CI effects

Software Quality		**Development Process**	
Reliability - Maturity - Fault tolerance	**Maintainability** - Testability - Adaptability	**Time to run build/test**	**Cost for development**

KPI 3, 4, 5: Effort Variance, Schedule Variance, Cost Variance

Table 15 presents the evaluation from Test factory on the KPI of Effort Variance, Schedule Variance and Cost Variance, which is the basis of analysis about effects of CI process.

Table 15: Evaluation of Effort, Schedule, and Cost variance

Unit of Analysis	**Coded text**
Daimler Test Factory	Effort variance, schedule variance, and cost variance will also be improved through CI. CI can reduce the planed effort and cost to test, and finish the test ahead of time. CI cannot control the actual situation. But CI can keep the planned Effort, schedule date and cost to a stable level, because of the standard process. For different CI process, time to run build, time to deployment and time for test can vary from each other. But for one process, the value for time to run build, time to deployment and time keep in an almost static value, which is a result from all other interview.

Based on the statement from test factory, "CI can keep the planned Effort, schedule date and cost to a stable level, because of the standard process." In a CI process, "value for time to run build, time to deployment and time keep in an almost static value".

Therefore the development process can be continued in a stable situation. This indicates that the risk of development can be reduced.

In conclusion, these three KPIs can be only mapped to Risk management.

Quality Metric1: Test Coverage

Table 16 presents the evaluation from Test factory on the Quality Metric of Test Coverage, which is the basis of analysis about effects of CI process.

Table 16: Evaluation of Quality Metric-Test Coverage from Test factory

Unit of Analysis	**Coded text**
Daimler Test Factory	Test coverage is improved through CI. Every requirement will be tested through the automatically test. No requirement will be ignored through CI. Every change will be integrated to the whole parts.

Through CI process, every requirement will be tested together with the whole requirements through the automatically test. "Every change will be integrated to the whole parts." Therefore high test coverage will be reached through CI process.

Maintainability

- **Testability:** All test cases have been transferred into script being running in CI process. Test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met.

Reliability

- **Maturity:** Based on the statement from test factory, with the high test coverage, the potential failure will be identified during test phase; the frequency of failure of the finished software is decreased with CI.
- **Fault tolerance:** With the high test coverage, the finished product which has passed all tests will have a high Fault tolerance namely the software can operate as intended despite the presence of hardware or software faults.

Functional suitability

- **Functional completeness and Functional correctness:** If some testing results will not be generated from test, namely the specific function of software is not existing, or If software cannot generate the right results from the test, namely function of software is not running correctly, with the high test coverage, it is efficient to identify the problem and to solve it later on. In a result, Functional completeness and Functional correctness can be assured.

In conclusion, Quality Metric1: Test Coverage can be mapped to the following aspects of CI effects in Table 17.

Table 17: Quality Metric Test Coverage mapped to aspects of CI effects

Software Quality		
Maintainability	**Reliability**	**Functional suitability**
- Testability	- Maturity	-Functional completeness
	- Fault tolerance	-Functional correctness

Quality Metric2: Defect Leakage (UAT)

Table 18 presents the evaluation from Test Factory on the Quality Metric of Defect Leakage (UAT), which is the basis of analysis about effects of CI process.

Table 18: Evaluation on Defect Leakage (UAT)

Unit of Analysis	**Coded text**
Daimler Test Factory	Defect leakage (UAT) will be reduced greatly through CI. (1)Before release, the requirements will all be tested more than once, some important requirements, which developed at the beginning are tested a lot of times. Because the finished parts will be tested with the newly coming parts again and again during CI. So the defects found by customer will be reduced greatly, only about 4%-7%. (2)After each integration, the finished parts can be showed to customer in necessary. Then the customer can always check in side one sprint, if the development is going in the right direction. So after the final release, the software will act as exactly as possible with the expectation from customer.

Defects identified in UAT means, the defects identified in the acceptance phase. From the statement from test factory, Defect leakage (UAT) has been reduced greatly through CI process. And how CI process helps to reduce this index has also been explained from 2 points.

Functional suitability

The very low Defect leakage (UAT) indicates that Functional completeness and Functional correctness are kept at a high level through CI. Because of the interaction with stakeholder, functions of software can facilitate the accomplishment of specified tasks and objectives, namely functional appropriateness can be increased.

Usability

- **Appropriateness recognizability:** After each integration, the finished parts can be showed to customer. The stakeholders can recognize whether a product or system is appropriate for their needs.
- **Operability:** Through the continuous feedback to customer, after the final release, the software will act as exactly as possible with the expectation from customer. Therefore the operability of the software will be greatly increased.

Maintainability

- **Maturity:** Few defects have been found in the acceptance test, namely by the customer side. It means the software can meet needs for reliability under normal operation.
- **Availability:** It means the software is operational and accessible when required for use.

In conclusion, Quality Metric2: Defect Leakage (UAT) can be mapped to the following aspects of CI effects in Table 19.

Table 19: Quality Metric- Defect Leakage (UAT) mapped to aspects of CI effects

Software Quality		
Functional suitability -Functional completeness -Functional correctness -Functional appropriateness	**Usability** -Appropriateness recognizability -Operability	**Maintainability** -Maturity -Availability

Quality Metric3: Test execution productivity

Table 20 presents the evaluation from Test factory on the Quality Metric of Test execution productivity, which is the basis of analysis about effects of CI process.

Table 20: Evaluation on Test execution productivity

Unit of Analysis	**Coded text**
Daimler Test Factory	Test execution productivity will be greatly improved through CI, because of the high automation process in CI.

Test execution productivity means Number of test cases executed/Effort in Person days. It is clear that, with the automation process in CI, with the same effort in Person days, much more test cases will be executed.

Maintainability

- **Testability:** With the automation build /test process, all established test criteria can be checked efficiently, whether they have been achieved in the development.

Reliability

- **Maturity:** with the automated build /test process in CI, it is easier, faster and cheaper to test the software repeatedly. The errors and failure of the development will be corrected on time. In the end, the frequency of failure has been greatly reduced.

Development process

CI saves time to run builds /test and saves cost for development further.

In conclusion, Quality metric3: Test execution productivity can be mapped to the following aspects of CI effects in Table 21.

Table 21: Quality Metric-Test execution productivity mapped to aspects of CI effects

Software Quality		**Development Process**	
Maintainability - Testability	**Reliability** - Maturity	**Time to run build/test**	**Cost for development**

Quality Metric4: Requirement stability

Table 22 presents the evaluation from Test factory on the Quality Metric of Test execution productivity, which is the basis of analysis about effects of CI process.

Table 22: Evaluation of Quality Metric- Requirement stability from Test Factory

Unit of Analysis	**Coded text**
Daimler Test Factory	Requirement stability will be affected through CI. The change requirements come from 2 ways: (1) Customer changes the requirements initially. (2) Requirement is developed not in the right direction like the customer really wants. From the second way, CI assures in this point that, the requirements will be right developed as the customer really wants. Then the requirements stability can be assured. Because after each integration, the finished parts can be showed to customer in necessary. Then the customer can always check in side one sprint, if the development is

	going in the right direction. So after the final release, the software will act as exactly as possible with the expectation from customer.

Functional suitability

After each integration, the newest software version can be displayed to stakeholders. Then the stakeholders can check, whether the development has been taken in the right direction. So that, the set of functions can cover the specified tasks, the software can also provide the correct results with the required precision, and functions of software can facilitate the accomplishment of specified tasks and objectives.

Therefore, functional completeness, functional correctness, and functional appropriateness can be assured.

Usability

- **Appropriateness recognizability:** After each integration, the finished parts can be displayed to customer. The stakeholders can recognize whether a product or system is appropriate for their needs.
- **Operability:** Because of the continuous feedback to customer, after the final release, the software will act as exactly as possible with the expectation from customer. Therefore the operability of the software will be greatly increased.

Maintainability

- **Testability:** Based on stable requirements, test team can design the relative test criteria and test cases. If the requirement stays unstable, it would be difficult for the test team to design the relative test criteria and test cases in a good quality on time.

So that CI process has helped the test team indirectly to establish test criteria for the software on time and in good quality. Tests can be performed to determine whether those criteria have been achieved. Therefore Testability can be assured.

Communication

High requirement stability also indicates that the Communication with stakeholders runs well.

In conclusion, Quality metric4: Requirement stability can be mapped to the following aspects of CI effects in Table 23.

Table 23: Quality Metric- Requirement stability mapped to aspects of CI effects

Software Quality			**Communication**
Functional suitability -Functional completeness -Functional correctness -Functional appropriateness	**Usability** -Appropriateness recognizability - Operability	**Maintainability** - Testability	**Communication with stakeholders**

Based on the above mapping relationship and interpretation from the researcher, interview protocol from test factory has been coded to the relative CI effects. In the coding table, the same explanation for these KPIs and Quality metrics will not be displayed repeatedly. Based on the allocation to the relative CI effects, the analysis for RQ3 is undertaken.

A screenshot for an overview of the coding process is in the Figure 21.

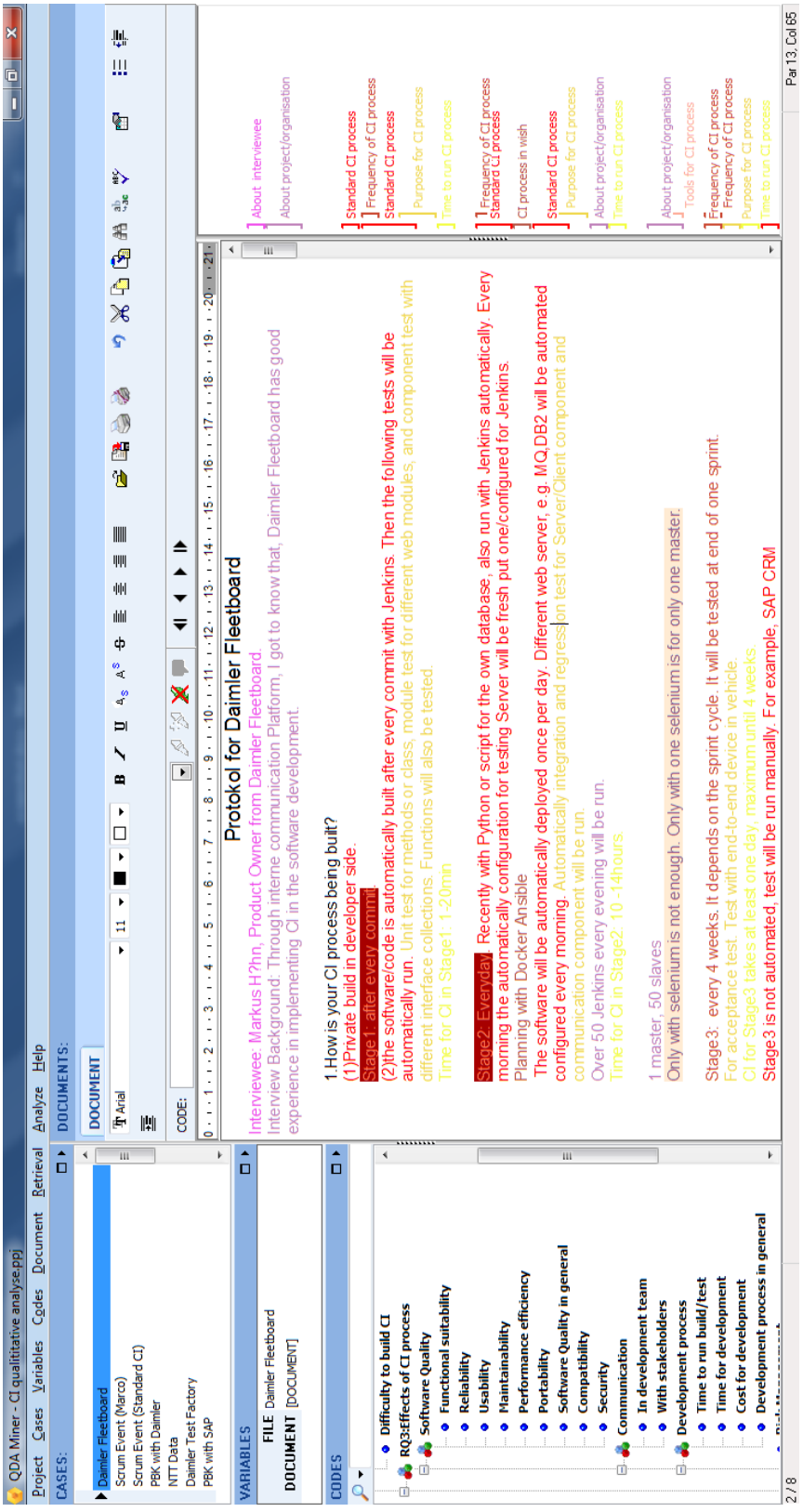


Figure 21: Coding process with QDA Miner

After all these coding steps, the coding results for qualitative data are demonstrated in the coding table, which will be described in more details in Section 6 Results.

5.4 Hypotheses Generation and Confirmation

In this case study, a set of hypotheses are generated from literature, which can be confirmed after the coding process. Another set of hypotheses have been generated after coding process, which will be confirmed when further data collection is conducted in the field. This process results in an iterative approach, in which data collection and analysis are conducted in parallel.

During the literature review, hypothesis1 (H1) and hypothesis2 (H2) have been generated. Similarly, during the data collection and data analysis in project PBK, hypothesis3 (H3) has been generated.

H1: A standard CI process often contains 3 components: VCR, CI server, and feedback framework.

H2: Through CI process, software quality can be assured and risk can be reduced.

H3: CI is not suitable to be built in all agile projects. There are challenges to build CI in some agile projects

Hypothesis confirmation is done in the following case study or within data analysis from other units of analysis. During the data collection process and data analysis process, above mentioned H1 and H2 are confirmed. During the data analysis from another unit of analysis, H3 is confirmed. This process can be showed with Table 24.

Table 24: Hypotheses generation and confirmation process

Hypotheses definition	**Techniques**	**Done in which phase of this case study**	**Examples of Hypotheses(H)**
Hypothesis generation	Constant comparisons & Cross-case analysis	Literature review	**H1:** A standard CI process often has 3 components: VCR, CI server, and feedback framework. **H2:** Through CI process, software quality can be assured and risk can be reduced.
		First cycle of unit of analysis	**H3:** CI is not suitable to be built in all agile projects. There are challenges to build CI in some

			agile projects
Hypothesis confirmation	Triangulation & Replication	First cycle of unit of analysis	**H1** and **H2**
		Following cycle of unit of analysis	**H3**

5.5 Validity Procedure

The validity of a study denotes the trustworthiness of the results, and to what extent the results are not biased by the researchers' subjective point of view [9].

There are different ways to classify aspects of validity and threats to validity in the literature. In this case study, a classification scheme from Yin [10] has been chosen. This scheme distinguishes between four aspects of validity, construct validity, internal validity, external validity, and reliability [9].

The four validity categories are introduced briefly with the corresponding validity threats from this case study. In addition, the relative validity measures to each validity threat, which have been undertaken during the case study, will also be described in the following.

Category 1: Construct Validity

To what extent the operational measures that are studied really represent: what the researcher has in mind and what is investigated according to the research questions [9].

Table 25: Threats and measures to Construct Validity

Threat	**Measure**
The concept of CI is not the same between researcher and the interviewed persons.	Improve understanding Following key concepts has been introduced: 1.Continuous integration 2. Purpose of the study 3. Some general information about the interview and its structure.

Category 2: Internal Validity

To what extent the researcher is aware of the risk that, the investigated factor is also affected by a third factor.

Table 26: Threats and measures to Internal Validity

Threat	**Measure**
The researched Effects of CI may be affected not only through CI process, but also through other agile methods, which have been implemented parallel in an agile process.	Check of potential factor Interviewees are asked, whether there is parallel process to CI.

Category 3: External Validity

To what extent it is possible to generalize the findings. And to what extent the findings are of interest to other people outside the investigated case [9].

Table 27: Threats and measures to External Validity

Threat	**Measure**
The challenges in RQ2 may be not so popular in the whole agile world than in the studied case studies.	Multiple data sources This study included interviews from different companies with different roles and teams to benefit the external validity of this research; The generalizability of the results has been improved positively.
	Digital storage All data collected and documented in this study were digitalized and stored, e.g. interview notes, audio recordings. It allows for future external assessment.

Category 4: Reliability

To what extent the findings of the study are dependent on researchers that executed the research. And to what extent can the study be reproduced with similar results [9].

Table 28: Threats and measures to reliability

Threat	**Measure**
Collected data is coded in an improper way.	Improve coding process 1. No pre-determined codes were applied to the collected data. Thus avoiding forcing the data into a specific code or category. Codes were instead induced from the data collected [25]. 2. After coding, interview protocol is given back to interviewee, to be checked and commented. 3. All data collected and documented in this study were digitalized and stored, such as interview notes, pictures help to understand important concepts.
Questionnaires or interview questions are unclear to the responders.	Member checking 1.Developed transcripts of interviews are sent back to interviewee to check misunderstanding, change of view, and correctness of products and organizations mentioned in the interviews 2.Review of explanations, models, conclusions and recommendations by interviewee In order that, the risk of having fault in the transcripts and analysis will be reduced.
Interviewees for questionnaire may come from the same organization, which may lead to non-general results about CI process.	Information recording 1. Basic record of invited person for questionnaire has been undertaken. In order to ensure that, they are not from the same organization. 2. The person's tasks in the project organization were checked during the interview.

There is Only single interviewer for the case study.	Improve triangulation 1. Interviews are organized with different companies, organizations, teams, and roles, which enriches the triangulation. 2. Two data collection methods have been used: questionnaire and interview. 3. Alternative explanations and theories in the analysis have been formulated to improve the analysis.
Interview protocol is the main source of qualitative data. No enough multiple forms of documentation.	Extend data type 1. Try to get more respondents for the questionnaire. 2. Beside interview protocol, try to get more data from testing reports, projects progress report, e.g. KPIs and Metrics from test factory.

6 Results

6.1 Results for RQs from quantitative data analysis

According to the data analysis procedure described in Section 4, the quantitative data analysis has been undertaken in the following procedure: (1) Collect questionnaire results; (2) Coding process for questionnaire results; (3) Present coded data in charts; (4) Generations/findings from the tables and charts.

This section focuses on presenting the Generations and findings from the tables and charts as results. Quantitative data has been only collected and analyzed for RQ1 and RQ3.

6.1.1 RQ1: How has CI been built in different agile development process

For RQ1, quantitative data can provide data from these aspects:

(1) Implemented process in CI;
(2) Purpose of CI process;
(3) Frequency of CI process;
(4) Tool for CI process;
(5) Tool for Version control;
(6) Feedback framework.

All these 6 aspects will be analyzed and interpreted in details in the following.

(1) Implemented process in CI

There are totally 14 persons have ever implemented CI process in the agile development process from 16 persons who have finished filling the questionnaire.

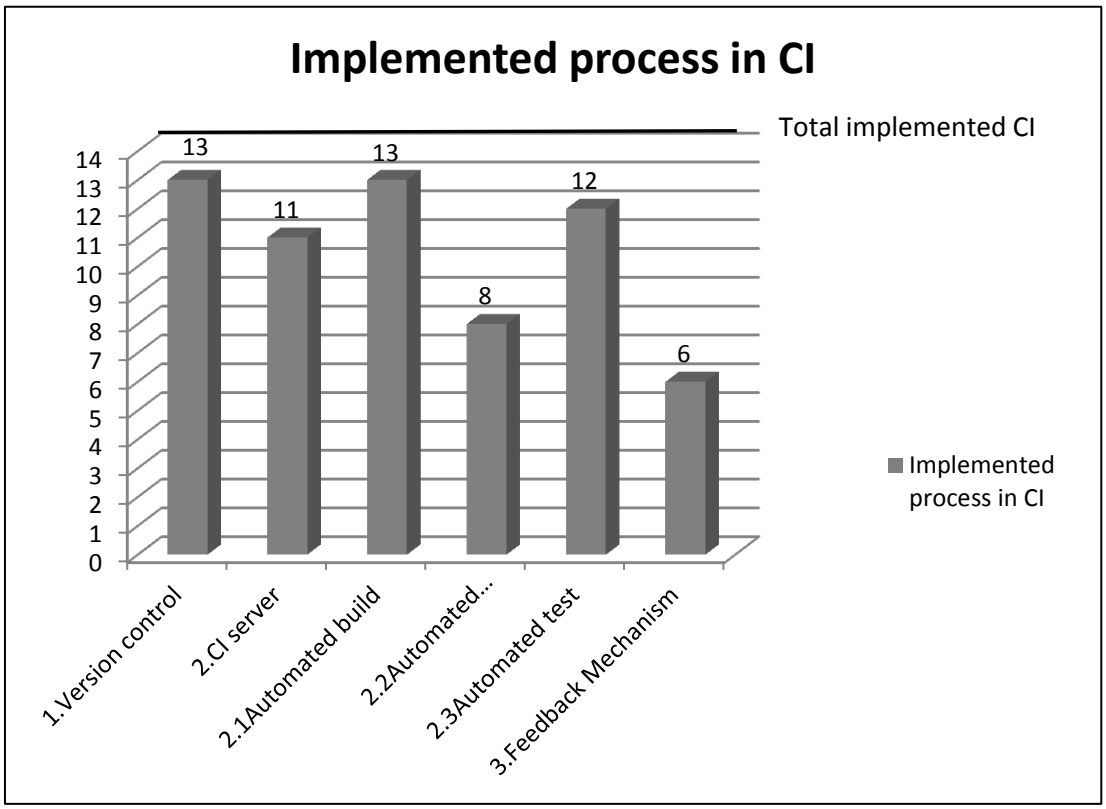


Figure 22: Implemented process in CI

13 people have implemented Version control and automated build. 12 people have implemented automated test. 11 people have separate CI server. 8 people have automated deployment and 6 have implemented feedback framework.

It can be concluded that almost all interviewees have implemented Version control, automated build and automated test in their CI process. Most of them have a separate CI server. More than half of CI users have automated deployment. Since deployment is not a necessary part of every development process. Less than a half CI process has feedback mechanism. Most CI Server can afford feedback mechanism. Besides, there are a lot of methods to give feedback. IT is not necessary to install a separate tool for feedback.

(2) Purpose of CI process

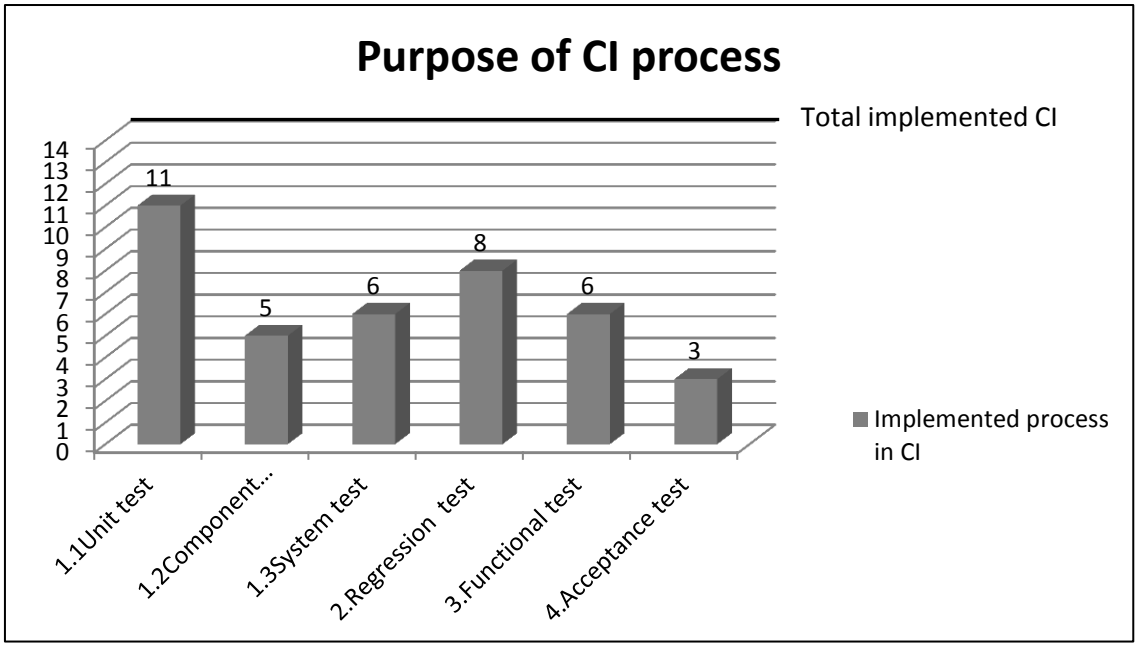


Figure 23: Purpose of CI process

From 14 implemented CI process, 11 CI processes have the purpose for Unit test. 8 CI processes have the purpose for Regression test. 6 CI processes have the purpose for system test and functional test. 5 CI processes have the purpose for Component test. Only 3 CI process are with the purpose for Acceptance test.

In conclusion, Unit test is the most popular purpose of CI process. Regression test is also a very common purpose of CI process. Almost a half CI process has the purpose to realize system test, functional test and component test. CI process has been not often implemented to realize acceptance test.

Based on the definition and mechanism of CI process, it is also easier to implement CI for Unit test, in which modules and classes of the coding part can be checked. The definition of regression test verifies that software previously developed and tested still performs correctly after it was changed or interfaced with other software [26]. The goal of regression test is similar to the goal of CI process. So it is understandable that, CI process is often used for Regression test. To realize system test and functional test, not only code part should be taken into account, but the particular test case should be also be checked, whether the specific function is fulfilled. So it is more difficult to realize System test and Functional test than Unit test. It is also the reason for few purposes for acceptance.

(3) Frequency of CI process

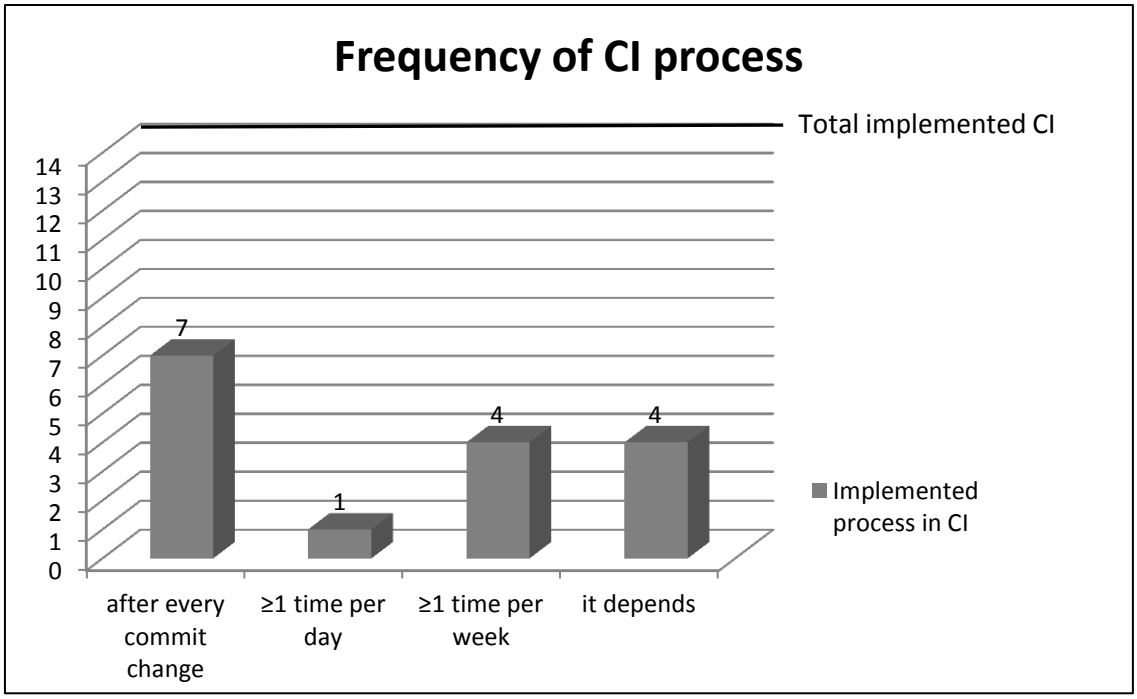


Figure 24: Frequency of CI process

From 14 implemented CI process, 7 CI process have been executed after every commit change. 4 CI process have been executed more than once per week or depend on the needs. Only 1 CI process has been executed more than once per day.

In conclusion, a half CI process has been executed after every commit change, which has been defined in the concept of CI. Fewer than a half CI processes have been triggered more than once in a week. The same number of CI processes have been triggered depend on the actual needs. A few CI process has been triggered more than once per day.

This situation reflects a non-positive use of CI in the practice. In fact, CI process should be executed after commit change as in the definition. Only in this way, the benefits of CI process can be fully achieved. This situation also reflects the difficulty to implement CI from the side. In the agile organization where CI is not executed after every commit change, maybe they don't need such frequent build integration at all, or such frequent build integration is too expensive for them to implement.

(4) Tool for CI process

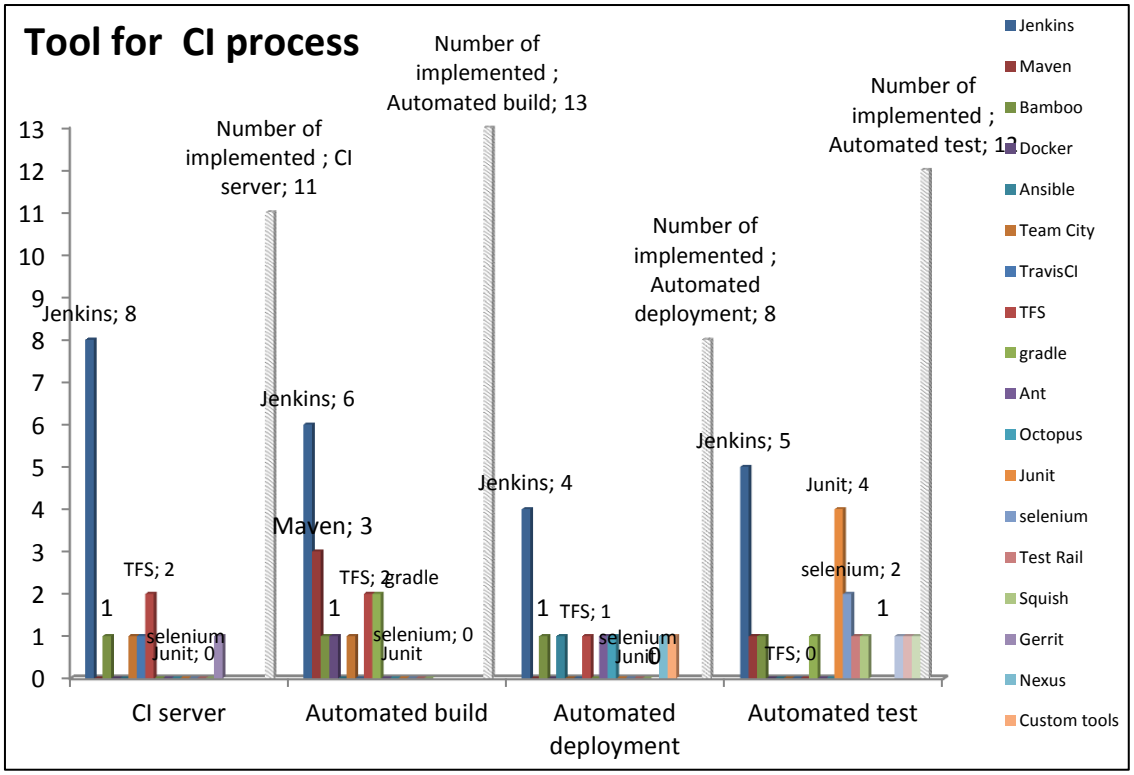


Figure 25: Tool for CI process.

11 CI processes have separate CI server. 8 CI sever have been implemented with Jenkins, 2 with TFS (Microsoft Test Foundation Server). Other Tools, such as gradel, Gerrit and so on have only been used once.

13 CI processes have included automated build. Jenkins has been used by 6 automated build processes. Maven has been used by 3 automated build processes. TFS and gradle have been used by 2 automated build processes.

8 CI processes have included automated deployment. Jenkins has been used by 4 automated deployment processes. TFS, Bamboo, Ansible, Ant and so on have been used once for automated deployment.

12 CI processes have included automated test. Jenkins has been used by 5 automated test processes. Junit has been used by 4 automated test processes. Selenium has been used twice for automated test processes. TFS, Bamboo, SonarQube and Maven have been used once.

It can be seen obviously that, Jenkins is the most popular tool for all CI process. TFS is also a popular tool which is widely used in almost all CI processes. Except of Jenkins

and TFS, Maven and gradle for automated build, Junit and Selenium for automated test are also popular in the practice.

(5) Tool for Version control

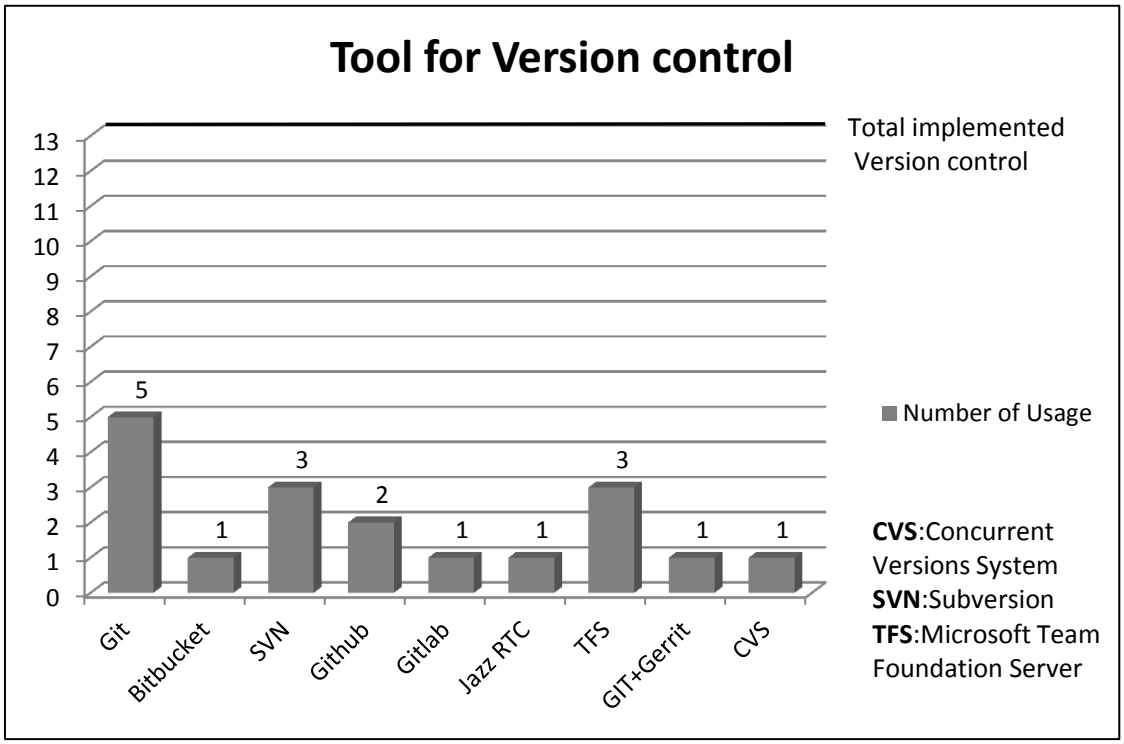


Figure 26: Tool for Version control

11 CI processes have implemented version control repository. This diagram describes the situation of the tool being used for Version control. Git, SVN, TFS, and Github are the most popular tool for version control.

(6) Feedback framework

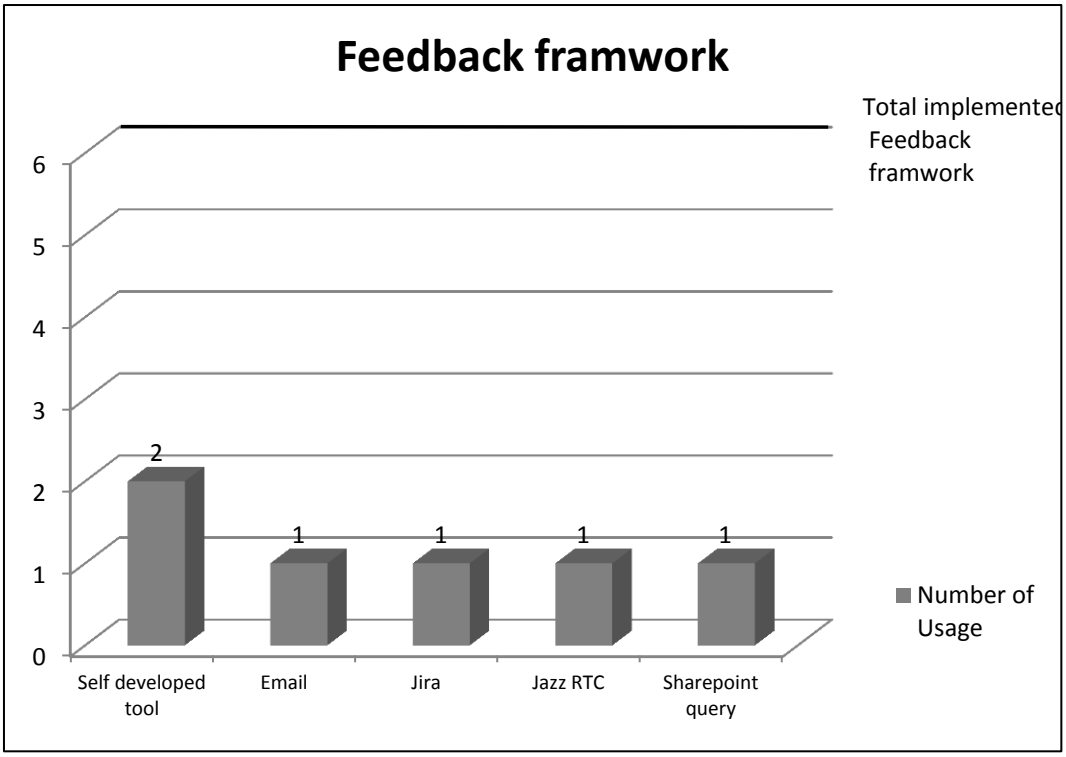


Figure 27: Feedback framework

6 CI processes separate feedback framework. The tool has been used equally distributed. Self-developed tool and email are popular in the practice.

6.1.2 RQ3: What effects have CI process?

For RQ3, quantitative data have been analyzed and interpreted from these aspects:

(1) Evaluation of CI from four main aspects and their sub-aspects;
(2) Four effects of CI process;
(3) Relationship between CI process and effects.

In the following, the analysis of these 3 aspects will be described in the following.

(1) **Evaluation of CI from 4 main aspects and their sub-aspects**

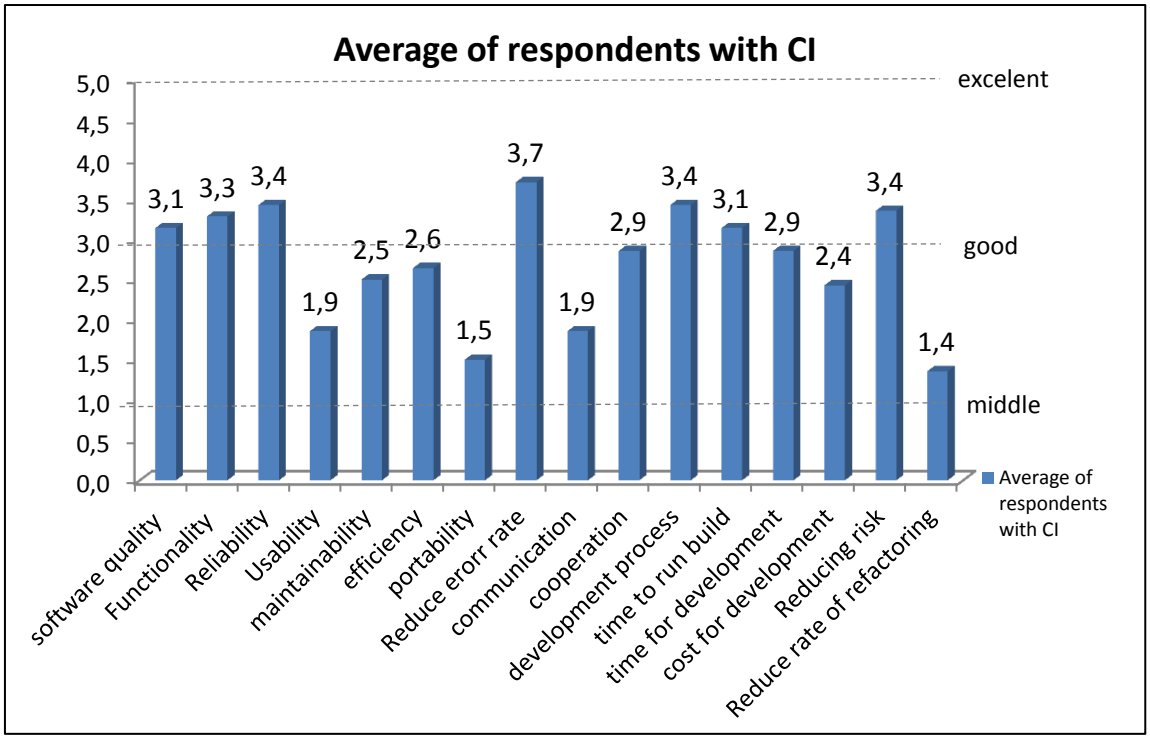


Figure 28: Evaluation of CI from 4 main aspects and their sub-aspects

This diagram describes the evaluation on 4 main aspects and their sub-aspects on effects of CI process. As explained in the coding process, the evaluation of "excellent" is corresponding to the score of 5. The evaluation of "good" is corresponding to the score of 3. The evaluation of "middle" is corresponding to the score of 1.

This diagram displays clearly that the average of the evaluation of all researched aspects of CI process is over the level of "middle". It indicates that, CI process has positive effects on all research aspects.

"Reduce error rate" has the highest evaluation of 3.7. "Reliability", ""development process" and "reduce risk" has the second highest score of 3.4. "software quality", "Functionality", and "time to run build" all have the score over 3, which corresponds to the evaluation of "good". "corporation" and "time for development " have the score of 2.9 , which is slightly less than 3.0.

"Reduce rate of refactoring" has the lowest score of 1.4. But "reducing risk" has a very high score of 3.4. After the analysis of researcher, the score for "Reduce rate of refactoring" is not convincible. Maybe the questionnaire responder doesn't understand the meaning of refactoring rate very well. So that, the evaluation of this index hasn't been

filled in large quantities. “portability”, “usability” and “communication” all have the score under 2.

The 3 index with the highest score are “reduce error rate”(3.7), “reliability”(3.4),”development process” (3.4)and “reducing risk”(3.4). They respect the main aspects of Software quality, Development process, and Risk management. So it can be concluded that, CI process can improve Software quality, Development process and Risk management.

Since the score for “cooperation” (2.9) and “communication” (1.9) is near to good(3.0). It can be completed that, CI process can also improve Communication, but not so effective like the other 3 aspects.

(2) Evaluation of CI from 4 main aspects

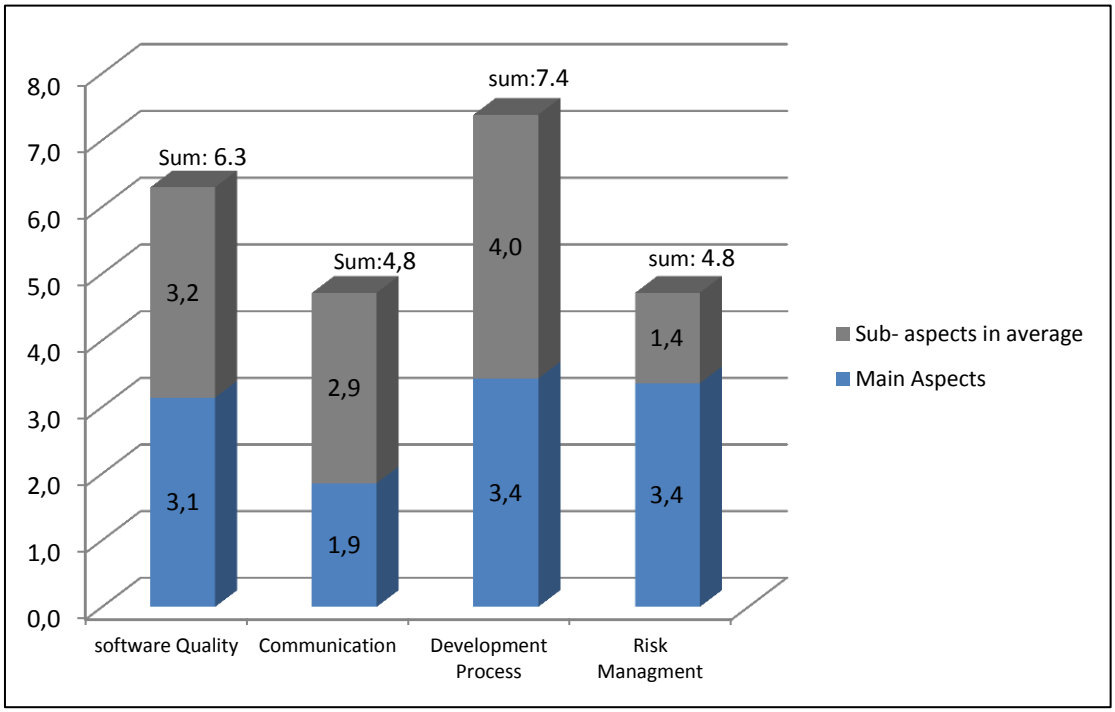


Figure 29: Evaluation of CI from 4 main aspects

This diagram describes the comparison of the evaluation between 4 main aspects. The blue bars are the evaluations of 4 main aspects, namely Software quality, Development process, Risk management and Communication. The red bars are evaluations of their sub-aspects in average. For software quality, the red bar is the average evaluation of “Functionality”, “Reliability”, “Usability”, “maintainability”, “efficiency”, “portability”, and “Reduce error rate”. For Communication, the red bar is the evaluation of “co-

operation". For Development process, the red bar is the average of the evaluation for time to run build, time for development, and cost for development. For Risk Management, the red bar is the evaluation of "Reduce rate of refactoring".

After observation the sum scores of both main aspects and their sub-aspects. The second place is followed by Software quality. The sum of Communication and Risk management are the same, which are in the third place. It can be concluded that CI process has the most significant improvement on the development process, the secondly significant promotion on software quality.

Among the blue bars, Development process and Risk management have the highest score of 3.4. Software quality has the second highest score of 3.1. Then communication has the lowest score of 1.9.

Among red bars, the situation has been slightly changed. Development process has still the highest score of 4.0, which is even higher than the main aspect along. Instead of Risk management, the average of sub-aspects of Software quality is in the second place. The average of sub aspects of communication is much higher than the main aspect. Instead of Communication in the main aspect, the average of sub aspect of risk management is in the last place.

(3) Relationship between CI process and effects

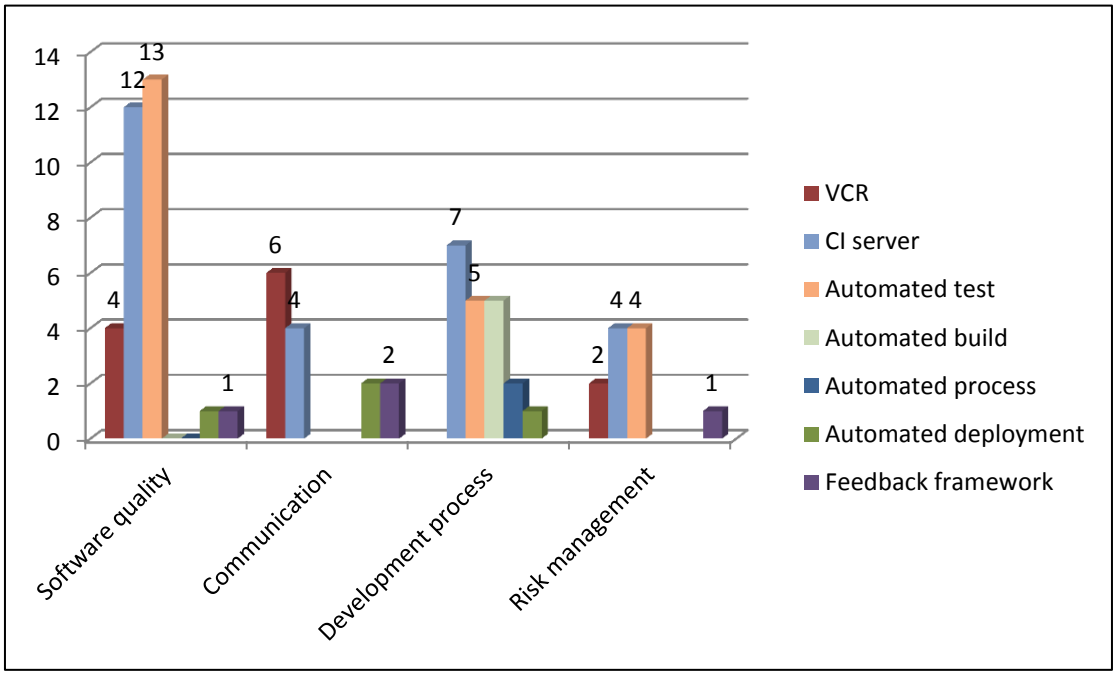


Figure 30: Relationship between CI process and effects

Figure 30 describes what CI process leads to what effect. To improve software quality, 13 people have filled in automated test, 12 people have filled in CI Server, 4 people have filled in Version control repository. Only one person has agreed on automated deployment and feedback framework. The most people agree on that, automated test facilitates the improvement of software quality. Slightly fewer people agree on the CI server. On one side, CI server contains automated test, automated build and so on. On the other side, some agile organization has implemented CI process without CI Server, rather with automated test, which is easier to implement in the CI process. Much fewer people agree on the Version control repository. The other CI process has got little agreement on the improvement of software quality.

To improve communication, 6 people have filled in Version control repository, 4 people have filled in CI server, and 2 people have filled in automated deployment and feedback framework. The most people agree on that, Version control repository facilitates the improvement of communication. Slightly less people agree on the CI server. It is similar with the software quality. A few people agree on the automated deployment and feedback framework.

To improve development process, 7 people have filled in development process. 5 people have filled in automated test and automated build. 2 have chosen general automated process and 1 for automated deployment. The most people agree on that, CI Server improves the development process. Slightly fewer people agree on automated test and automated build. Few people agree on automated deployment.

To improve risk management, 4 people have filled in CI Server and automated test. 2 people have filled in Version Control repository and 1 has filled in feedback framework. The most people agree on that, CI Server and automated test improve the risk management. Fewer people agree on Version Control repository. Few people agree on feedback framework.

It can be concluded that a CI server can promote improvement on all 4 aspects. Accept from CI server, automated test has promoted great improvement on the Software quality, development process and risk management. In addition, Version control repository does a great help to improve communication. Automated build facilitate the improvement of development process.

With this conclusion, agile organization can decide which CI process to implement based on which effect is in need.

6.2 Results for RQs from qualitative data analysis

According to the data analysis procedure, which has been described in Section 4.2.2. The qualitative data analysis has been undertaken in the following procedure:

(1) Collected Data

Collected qualitative data has been stored in the form of interview protocols.

(2) Coding process

Interview protocols have been processed through the coding process, which is described in Section 5.3.2 in details. After coding, four coding tables have been constructed for three RQs and general information separately. In addition, one coding frequency table has been created for all codes. These four tables can found in the attachments. The coded data can be analyzed together with all codes, categories and question classifications.

(3) Hypotheses Definition and Generalization

The Hypotheses definition/confirmation process and the corresponding techniques have been described in Section 5.4. In the following parts, coded data will be analyzed for the relative RQs including hypotheses definition and confirmation.

(4) Generalizations/findings

The generations and findings from the above process, as well as the summarization for each RQ will be presented in this Section6.

6.2.1 RQ1: How has CI been built in different agile development process

The goal of RQ1 is to research, how has CI been built in the different agile development process. This Research question has been research from 6 codes:

(1) Standard CI process;
(2) CI process in variation;
(3) Tool for CI process;
(4) Purpose of CI;
(5) Frequency of CI;
(6) Time to run CI process.

Code1: Standard CI process

After the coding process for interview protocols, the coded data for standard CI process from different interviews is presented in the following.

(1)From Interview 2 with “PBK with SAP”

Table 29: Standard CI process from “PBK with SAP”

Unit of Analysis	**Coded text**
PBK with SAP	In the development process for standard IBP, the development has built CI process.
PBK with SAP	(1) Unit test: build standard CI system with tool Jenkins, which is described in Section 2.3. (2) Component Test: build the connection between classes, through these connections to run component test with the similar CI system for Unit test.
PBK with SAP	In the configuration test of IBP in project PBK, Parse has been used as Version Control Repository. After every commit change, the finished code is uploaded to the version control tool. Jenkins picks this part up and tests this part with the already existed part, to see if it can be integrated to the existed parts. Afterward, mail as feedback will be sent out to the developer.

(2)From Interview4 with “Scrum Event (standard CI)”

Table 30: Standard CI process from “Scrum Event (standard CI)”

Unit of Analysis	**Coded text**
Scrum Event (Standard CI)	All kinds of tests are being automated implemented in the same CI system. The different kind’s tests depend just on different requirements. The CI process is described in the following: 1. VCR: Code upload to VCR tools, e.g. CVS/Git 2. CI Server 2.1 Automated deployment: Self-built Own deployment process. 2.2 Automated build with Docker. 2.3 Automated Test: HP Quick Test Professional (QTP), also known as HPE Unified Functional Testing (UFT). 2.4 Virtualization Platform with Docker. Comment: Have considered using Bamboos as CI Server, but the

	product consists of Oracle, Java, Pair Script, standard CI server. A standard CI cannot cover all the requirements. 3. Feedback: Self-built own build process. When error happens or test failed, feedback will be given to the development team.

(3)From Interview 6 with "Daimler Fleetboard"

Table 31: Standard CI process from "Daimler Fleetboard"

Unit of Analysis	**Coded text**
Daimler Fleetboard	Private build in developer side.
Daimler Fleetboard	Stage1: the software/code is automatically built with Jenkins after every commit. Then the following tests will be automatically run.
Daimler Fleetboard	The software will be automatically deployed once per day. Different web server, e.g. MQ, DB2 will be automated configured every morning. Automatically integration and regression test for Server/Client components and communication components will be run.
Daimler Fleetboard	Stage2: Every day. Recently with Python or script for the own database, also run with Jenkins automatically. Every morning the automatically configuration for testing Server will be fresh put one /configured for Jenkins.
Daimler Fleetboard	Stage3: Test with end-to-end device in vehicle. Not automated, test will be run manually. For example, SAP CRM

(4)From Interview 7 with **"**NTT Data"

Table 32: Standard CI process from "NTT Data"

Unit of Analysis	**Coded text**
NTT Data	Private build upload the code to repository, a unit number is given to this piece code after uploading, After Jenkins is triggered, this piece of code with already existed code will be automated deployed Configuration management runs automatical-

	ly/manually, e.g. databank configuration: add a table to the database. Or server configuration, the server for test has to be newly installed for Unit test.

(5)From Interview 3 with **"**Scrum Event (Marco)"

Table 33: Standard CI process from "Scrum Event (Marco)"

Unit of Analysis	**Coded text**
Scrum Event (Marco)	At the beginning of the development, Test driven develop is used to define the first rough framework. Afterward during the development process, CI (short for Continuous integration) is used to test, if the newly developed part can be integrated to the existed parts. As the correlation between the theory between TTD (short for test driven development) and CI, TTD and CI are being used in the practice very often.

Code2: CI process in variation

After coding interview protocols, the coded data for CI process with variation is presented as following:

CI variation1: CI in requirements engineering

An Interview about how to build CI in PBK with the perspective from agile coach for PBK was organized. The coded result for CI in requirements engineering is presented in Table 34.

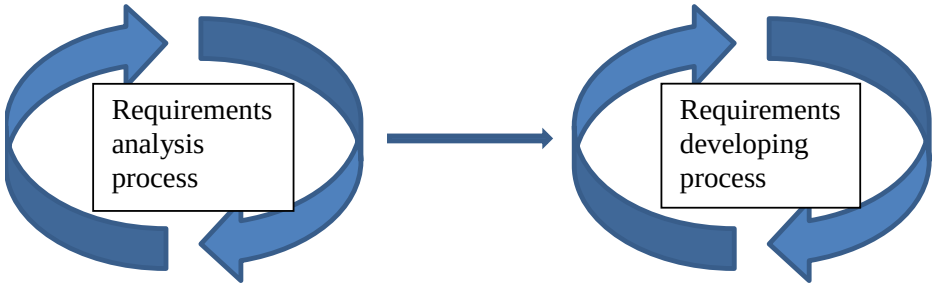
Table 34: CI process with variation in requirement engineering

Unit of Analysis	**Coded text**
Scrum Event (Marco)	CI variations have the same principle with standard CI process is that, after each change the new coming part is tested with the existed parts , to check if the new coming part can be integrated to the existed parts.
Scrum Event (Marco)	CI variation1 : CI in requirements engineering
Scrum Event (Marco)	For the project like PBK, is it also suitable to implement CI in the development process?

	Yes, the prince of the continuous integration can be used in the first cycle to define the requirements and further to create the suitable user stories. The principle of CI can be used as in the following development structure for the project and the corresponding table.(See Figure 22 and Table 29)
Scrum Event (Marco)	Requirement Engineering with CI: (1)Requirement from one work packet describing, acceptance criteria definite is equal to unit test (2)Summarize requirements from different requirements to build one user story is equal to component test (3)Check if the user stories compatible/tolerant with each other is equal to System test. The same key point needed to be implemented is: after every change, the newly developed part will be integrated to the older existed part.

The key feature of standard CI process is that, after every change, the new developed part will be integrated to the older existed part. Based on the information provided from the unit of analysis with "Scrum Event (Marco)" and the key feature of standard process from literature review, a concept of constructing CI process with variation for requirement engineering is built by the researcher and interviewee together. This concept is displayed in the following figure and table.

The Figure 31 shows that, after the Requirements analysis process, which is performed by CI process with variation, the requirements will be developed with standard CI process.



(Source: Own illustration)

Figure 31: CI process from Requirements analysis to Requirements developing

Table 35 demonstrates the mapping relationship between CI process with variation and standard CI process.

Table 35: Relationship between CI process with variation and standard CI process

CI process with variation for Requirements analysis process	**Standard CI process for Requirements developing process**
Requirement from one work packet describing, acceptance criteria definite	unit test
Summarize requirements from different requirements to build one user story	component test
Check if the user stories compatible/tolerant with each other	System test

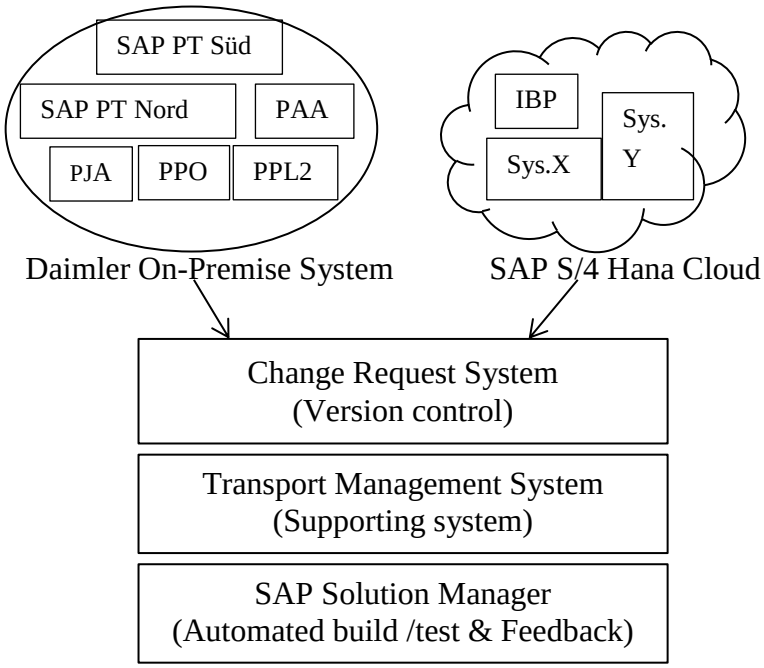
CI variation 2: CI in Acceptance test for PBK

Table 36: CI process with variation in Acceptance test

Unit of Analysis	**Coded text**
Scrum Event (Marco)	CI variation 2: CI in Acceptance test from Daimler Based on the principle from Standard CI process, a CI variation can be built for cross system, namely local system and system in the cloud development process.
PBK with Daimler	Change request system acts as VCR. SAP Consolidation system and Transport system can be used as supporting system. SAP Solution Manager can be treated as automated build/test. Test result as feedback will be given from SAP Solution Manager to the relevant developer.
PBK with SAP	(3)System test: in the investigation, supposing using SAP SolMan and logical component to build the connection between business use case and the real software parts which lying on the cloud or on premise. The CI system can be built based on SolMan, ChaRM, TMS.

Based on this information provided by units of analysis and the feature of CI process from Section 2.3.3, a concept of constructing CI process with variation for Acceptance test in cross system development is built by the researcher and interviewee together. This concept is displayed in the following figure and table.

Figure 32 shows that, to build a suitable CI process with the current and available system from project PBK, in order to realize the acceptance test for cross system development.



(Source: Own illustration)

Figure 32: Acceptance test in cross system development with CI process variation

Table 37 demonstrates the mapping relationship between CI process with variation for Acceptance test and standard CI process.

Table 37: Relationship between Acceptance test with CI variation and standard CI process

CI process with variation for Acceptance test	**Standard CI process**
SAP Change request system	VCR
SAP Consolidation system and SAP Transport system	automated build/test
Test result	feedback

CI variation 3: CI in configuration test from SAP

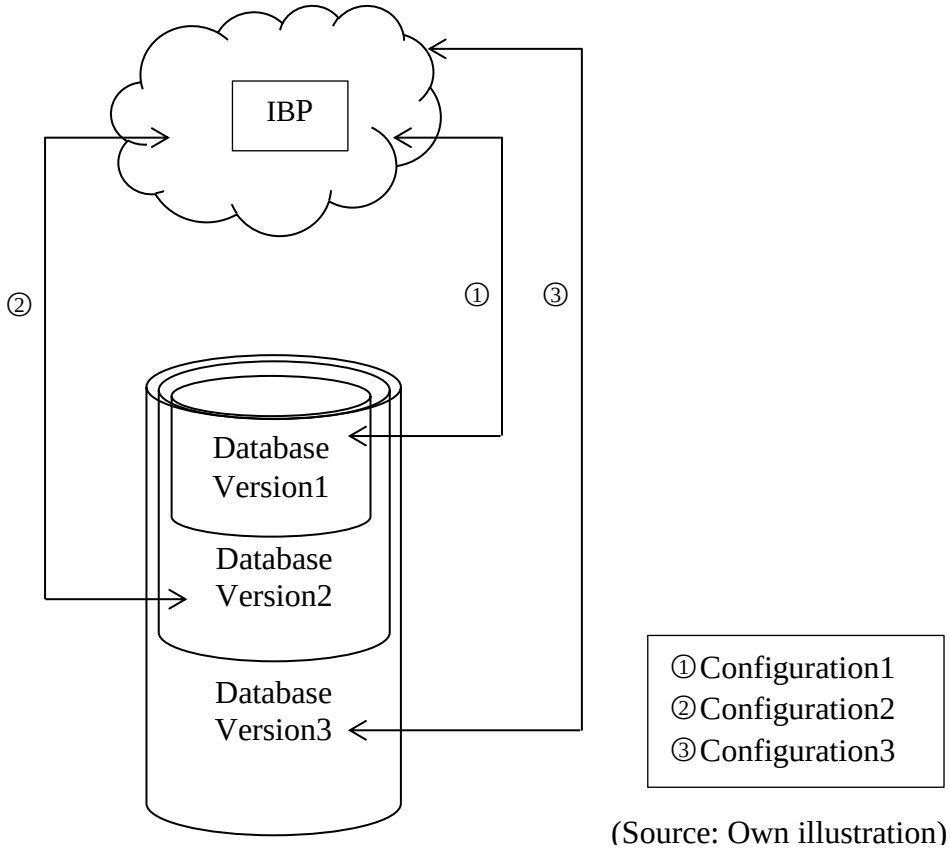
This Interview is about configuration in PBK, namely how to build CI in PBK with the perspective from the Agile coach and SAP Developer.

Table 38: CI process with variation in Acceptance test

Unit of Analysis	**Coded text**
Scrum Event (Marco) & PBK with SAP (Adam)	CI variation 3: CI in configuration test for SAP IBP (1)Every consultant should have the chance to take configuration in IBP, version control can be realized with manually dividing version number (2)An automated tool for functional test is developed for both configuration types, which act as CI Server for Automated build/test (3)Feedback can be given through various ways, emails, or special tool for organizing agile development process, like Version One
Scrum Event (Marco)	The configuration contains: (1) Data bank configuration, the same software will be run with different data input, to see as the extension of data volume, if the functionality of the software can be assured. (2) Different functions are combined in new ways or the hidden functions are activated according to the requirement during the progression of the project. In this case, the same data input should be afforded, to see with the same data, if the output is verified according to the requirements.

Figure 33 and Figure 34 are used to explain the basis for configuration test.

Configuration type1: Database configuration



(Source: Own illustration)

Figure 33: Database configuration

Configuration type2: Function configuration

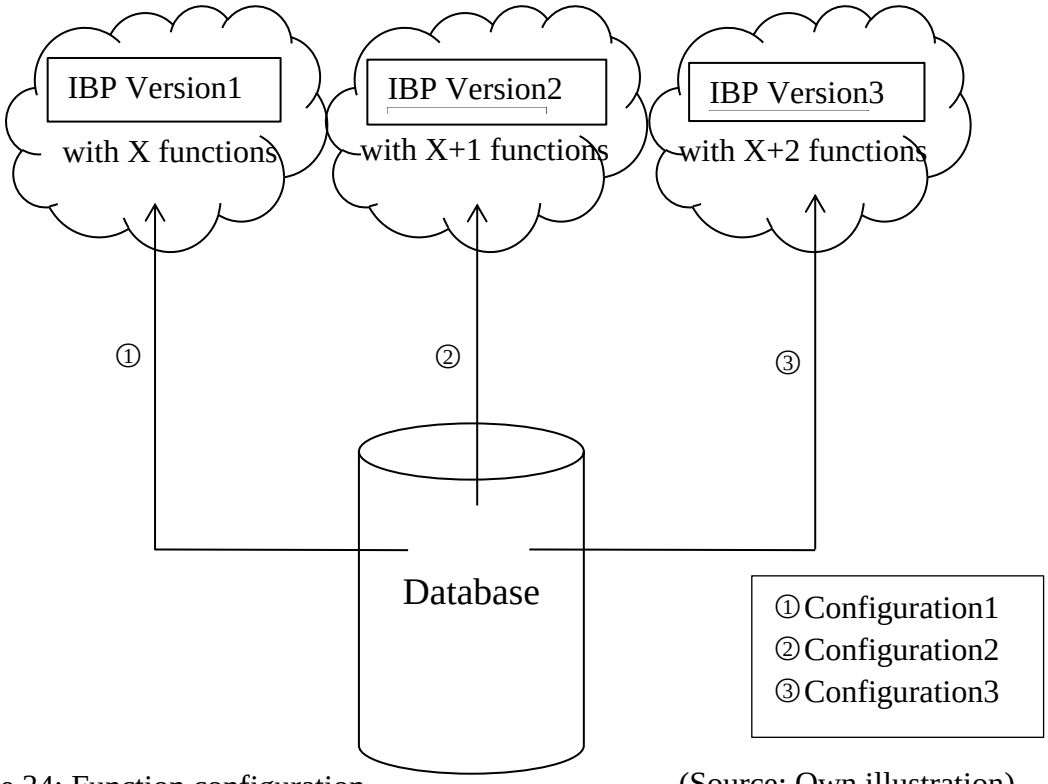


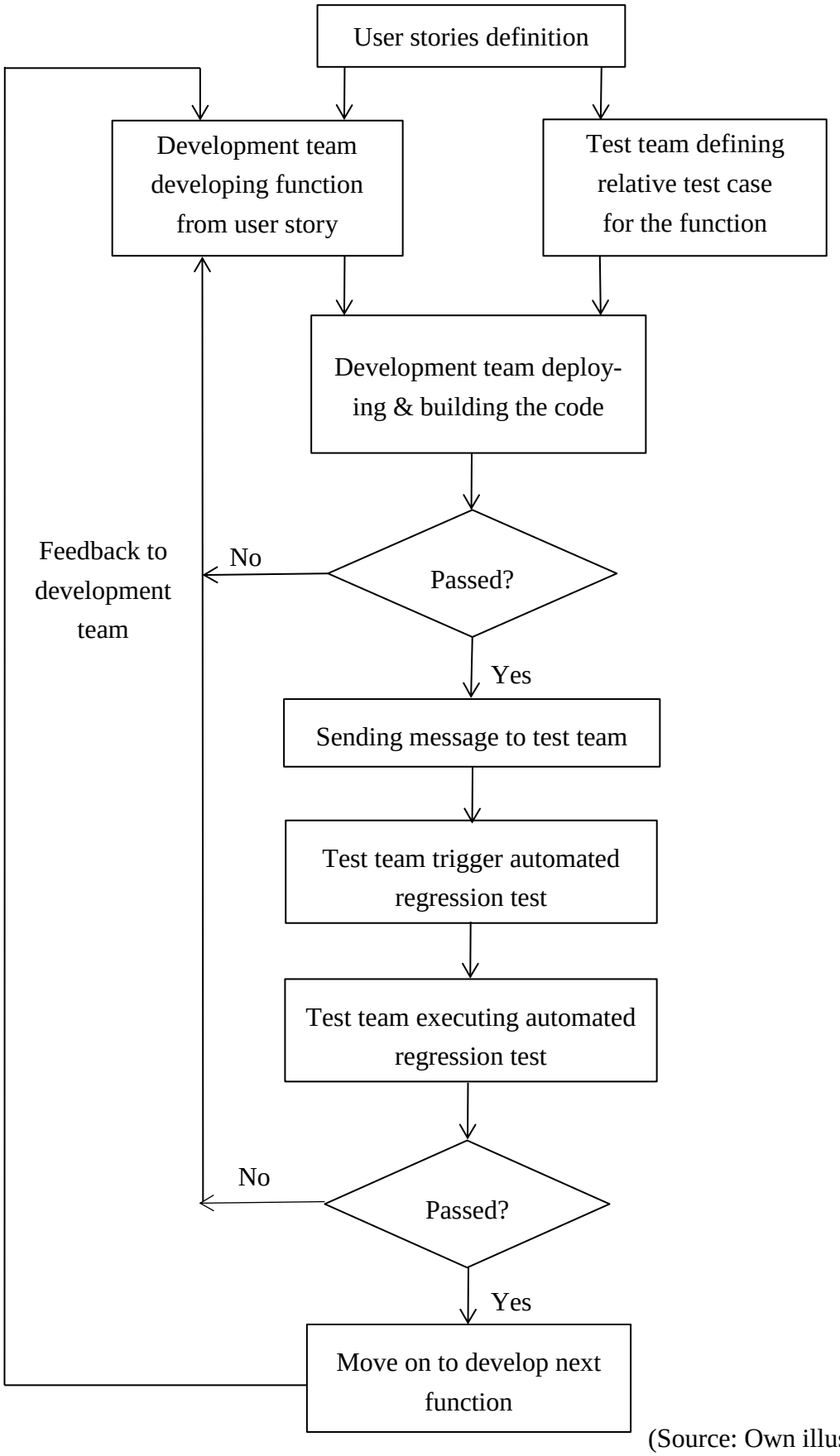
Figure 34: Function configuration (Source: Own illustration)

CI variation4: Functional test in Daimler test factory

Table 39: CI process with variation for Functional test in Daimler test factory

Unit of Analysis	**Coded text**
Daimler Test Factory	We are using agile method in the project. In one Sprint several user stories are defined. During the development team developing the function from user story, the quality assurance team, namely test team works on defining the relative test case for the function. The relative quality requirements for the function are also contained in the test case. The development team deploys and builds the code automatically with Jenkins. When the new deployment and build are successful passed. A message is sent to the test team. After receiving the message, the automated regression test is manually triggered but automatically executed in the test environment. If the test is failed, the defect message is sent to the development team.
Daimler Test Factory	Code is automatically built with Jenkins. As in the project no access to Jenkins is available, a manual trigger is needed. But the test team is trying to build the automated trigger. After the manual trigger, Functional test is executed with CI in variation.

To better present this CI process with variation for Functional test, a flow chart in Figure 35 has been drawn by the researcher based on the information provided by the unit of analysis.



(Source: Own illustration)

Figure 35: Flow chart for the Functional test with CI process variation

Code3: Tool for CI process

After the coding process for interview protocols, the coded data for Tool for CI process is presented in Table 40.

Table 40: Coded results for Tool for CI process

Unit of Analysis	**Coded text**
PBK with SAP	CI process for the development in SAP IBP has been built with Jenkins.
Daimler Fleetboard	Only with selenium is not enough. One selenium is for only one master.
Scrum Event (Standard CI)	1. VCR: CVS/Git 2. CI Server. 2.1Automated deployment: Self-built Own deployment process. 2.2Automated build with Docker. 2.3Automated Test: HP Quick Test Professional (QTP), currently known as HPE Unified Functional Testing (UFT). 2.4 Virtualization Platform with Docker. 3. Feedback: Self-built own build process.
Daimler Test Factory	Deployment team takes care of software/application version controlling with Jenkins. From automation tests perspective test team uses SVN (Subversion) and test management tools, e.g. HP ALM for version controlling of the scripts.
Daimler Test Factory	Depending on the application technology various tools are used to build test automation, for example, Selenium for Web application, SAP TAO for SAP application, HPE UFT for other applications are in the current set of tools for test automation
Daimler Test Factory	We have well-defined automation frameworks that will generate detailed results/ reports that can be used to measure application quality. These reports can be customized as per our customer needs.

Based on the information provided by different unit of analysis, the tool for CI process can be summarized as follows:

- CVS/Git, SVN and HP ALM are often used version control repository;

- Jenkins , Selenium, and Docker are often used as CI server;
- SAP TAO, HPE UFT(formerly known as QTP) are often used as automated test tool;
- Most agile teams are using self-developed feedback framework;

Code4: Purpose of CI process

After the coding process for interview protocols, the coded data for Purpose of CI process is presented in the Table 41.

Table 41: Coded results for Purpose of CI process

Unit of Analysis	**Coded text**
Daimler Test Factory	The process of automated test is functional test aligns to sprint methodology.
Daimler Test Factory	From the most part yes. For us quality assurance team, the most important is that, the software works as the customer wished from Business aspect. So the functional test is the key point for our CI process.
NTT Data	Focus on until unit test, in the following tests, e.g. functional test, system test are not the focus.
Daimler Fleetboard	Unit test for methods or class, module test for different web modules, and component test with different interface collections. Functions will also be tested.
Daimler Fleetboard	Automatically integration and regression test for Server/Client component and communication component will be run.
Daimler Fleetboard	Acceptance test is running for 3 times: (1) Develop Desk Truck; (2) Release candidate; (3) Actual release version (end release) end of one sprint.
Daimler Fleetboard	For acceptance test. Test with end-to-end device in vehicle.
PBK with SAP	Unit test, Component test and System test can be run through this CI process.
PBK with SAP	Currently, configuration test is being implemented. After every configuration, to test if the software works as expected. Automated regression test is being implemented to test the configurations.
PBK with SAP	Yes, CI process for the configuration is also possible.

The Statistical results for purpose of the CI process are presented in Table 42.

Table 42: Statistical results for Purpose of CI process

Purpose the CI process	**Number**
Functional test	3
Unit test	3
Automated regression test	2
Acceptance test	2
Configuration test	2

The purpose of the CI process is equally distributed. Functional test and Unit test are slightly more popular than other purposes.

Code5: Frequency of CI process

After the coding process for interview protocols, the coded data for Frequency of CI process is presented in Table 43.

Table 43: Code results for Frequency of CI process

Unit of Analysis	**Coded text**
PBK with SAP	After every commit change
PBK with SAP	a manual trigger, this depends on the decision of each consultant. Or choosing a fixed time trigger, e.g. once a day in the evening.
Daimler Test Factory	The frequency to take automated regression test depends on the frequency of finished deployment and build. Inside a sprint, which usually takes 3 weeks, several user stories or functions are developed. During developing one user story or function, Jenkins is used for the deployment and build. If the deployment or build are failed, the developer has to correct the defected part. If the deployment and build are successfully passed, an automated regression test is triggered. Therefore automated regression test can be taken once or several times in from one up to 3 weeks.
Scrum Event (Standard CI)	Only a trigger is needed to run the tests. CI process is triggered twice a day. One time is between 12:00 and 14:00. Another time is between 22:00 and 24:00.
Daimler Fleetboard	Can be once in a week/ month
Daimler Fleetboard	Stage1: after every commit

Daimler Fleetboard	Stage2: Everyday
Daimler Fleetboard	Stage3: Every 4 weeks
Daimler Fleetboard	It depends on the sprint cycle. It will be tested at the end of one sprint.

The frequency to run CI is totally various from agile team to agile team. 2 agile teams runs CI after commit change as described from the CI book. But most agile team runs CI dependent on their specific needs.

Code6: Time to run CI process

After the coding process for interview protocols, the coded data for Time to run CI process is presented in Table 44.

Table 44: Coded results for Time to run CI process

Unit of Analysis	**Coded text**
Scrum Event (Standard CI)	So the CI process usually takes 2 hours.
Daimler Fleetboard	Time for CI in Stage1: 1-20min
Daimler Fleetboard	Time for CI in Stage2: 10 -14hours.
Daimler Fleetboard	CI for Stage3 takes at least one day, maximum until 4 weeks.

Time to run CI is also totally different within different team, ranging from 1-20 min to as long as a sprint.

6.2.2 RQ2: What are the challenges to build CI in agile development process?

The purpose of RQ2 is to research the usage situation of CI process in agile development process. Table 45 presents the opinion from interviewee about CI process in the development.

Table 45: Coded results for the opinions to CI process

Unit of Analysis	**Coded text**
Scrum Event (Standard CI)	CI should be implemented in all agile developing process
PBK with SAP	Will CI be implemented in the new developing? Yes.

PBK with SAP	Yes, continuous software development should be realized.
Scrum Event (Standard CI)	CI should be built in all software developing process.
PBK with SAP	It is very important and necessary to project PBK.

In Table 45, all the interviewees agreed on that, CI should be built in the development process. But in the reality, CI was not built in every agile project. In fact, all CI variations reflect from the side that, they cannot build the standard CI process.

The difficulties to build CI process are listed in the Table 46.

Table 46: Coded results for difficulty to build CI process

Unit of Analysis	**Coded text**
PBK with SAP	Release plan is a difficulty to build CI in configuration test. Release plan influences the developing process, when considering both IBP on cloud and Daimler on premise system. If doing test with integrated 3 system parts from the cloud and on premise, the release cycle is every 3 months from SAP to Daimler on premise. Test can be done in SAP SolMan with logical components. If test errors happened, with this release Plan, developer has to wait for 3 months, until the next test. If considering only test for IBP, which the whole development is only on the cloud. There is no limit for a release cycle.
PBK with SAP	It is hard to test all components together, with the release cycles. For the particular system test, the standard CI server can be directly used. And it is very expensive and demanded to build an own CI server. SAP developing field consists of developing, consolidation system, and productive system. With this SAP developing field, no databank integration is needed. If an own CI server should be developed, it is very difficult to realize databank integration because of huge data volume.
Scrum Event (Standard CI)	But for the configuration in PBK is not ideal. Because it is not to build a new function. So it is difficult to define the concrete test cases and the expected test results. This is a complex and multiple dimensional planning software. The expected data comes from special customer request. When the data is defined, it will be difficult to change all of them to suit the configuration. It is

	not like in the normal software developing. In the normal software developing, it is easier to adjudge, if the test went through the test or not. If not, only the part with error should be corrected.
Scrum Event (Standard CI)	Software development with developing new function is suitable to use CI. But with the configuration, or only parts of the software should be developed, it will be difficult to realize CI. In these cases, the developing team can only use the idea from CI to design the testing process. Like the development team has been doing in the project PBK, to build automated regression test with CI.

From the coding results, the difficulty to build CI lies in the release cycle, non-new development process, and configuration-developing mixed development process. These kinds of development process are currently not suitable to build standard CI process. The relative agile team has either built CI process with variation or planned to build proper CI process for the agile development.

Project PBK is the cooperation partner for this thesis. At the beginning of the research, project PBK was the focus of the case study. A lot of information has been gathered to analyze the difficulty to build CI process in project PBK, which are summarized in the following:

Firstly, for the whole software development process, the primary task in the project is requirement analysis. But in the requirement engineering, it is difficult to build standard CI. To face this problem, CI process with variation1 for requirement engineering is planned for project PBK.

Secondly, during the research period, the software development of project PBK focuses on the configuration of standard software. In the configuration process, it is difficult to build standard CI process. The reason has been clearly explained within the coding Table 39 from Scrum Event (Standard CI). For example, "The expected data comes from special customer request. When the data is defined, it will be difficult to change all of them to suit the configuration." Therefore CI process with variation3 for configuration test is planned to solve this problem.

Thirdly, the acceptance test within CI process is difficult to be built for configuration-developing mixed development of SAP IBP, which is developed in cross system. The standard CI server can't be directly used for SAP landscape. A self-developed CI server is very expensive to be built and with extremely high requirements on technical background. On the other side, it is very difficult to realize databank integration on a self-

developed CI server because of huge data volume. Therefore, a CI process with variation3 for Acceptance test was planned to be built by the Daimler side. But this goal cannot be realized during the study period. The connection between SAP IBP and Solution Manager is not possible yet because of the technical problem.

Moreover, the difficulty to build CI process for these special development process from RQ2 and the planned CI process with variations to solve the related problem from RQ1, are the basis of research in RQ4. The research of RQ4 will be presented later.

6.2.3 RQ3: What effects have CI process?

RQ3 is dived into 4 sub-research questions:

RQ3.1 What effects has CI process to Software Quality

RQ3.2 What effects has CI process to Communication

RQ3.3 What effects has CI process to Development process

RQ3.4 What effects has CI process to Risk management

For RQ3.1 What effects has CI process to Software Quality. The software quality is researched from 8 sub aspects based on the quality model from Section 2.4.1, which correspond to the following 8 codes in the coding table for RQ3:

(1) Functional suitability;
(2) Performance efficiency;
(3) Compatibility;
(4) Usability;
(5) Reliability;
(6) Security;
(7) Maintainability;
(8) Portability.

A code for Software quality in general has also been taken in consideration. This code is designed for the evaluation from a unit of analysis, which can't be dived to the sub aspects of software quality.

RQ3.1 what effects has CI process to Software Quality

Code1: Functional suitability

Table 47: Coded results for Functional suitability

Unit of analysis	**Coded text**
PBK with SAP	Functions can often be checked with both SAP and Daimler sides
Scrum Event (Marco)	High quality of code.
Scrum Event (Standard CI)	In CI, different test cases for different functions will be implemented. So that functionality can be assured through CI.
Daimler Test Factory	Requirement stability will be affected through CI.
Daimler Test Factory	Test coverage is improved through CI.
Scrum Event (Marco)	The customer can give the feedback, if the right functions have been developed and if the requirements have been covered. Therefore, the software quality can be assured.
Daimler Test Factory	Defect leakage (UAT) will be reduced greatly through CI.
Daimler Test Factory	Defect identification will be improved through CI.
Daimler Fleetboard	Functionality: through regression test, functional test. Functionality can be assured through automated build, automated test and fast feedback.

9 evaluations for the Functional suitability have been collected and coded. All of them are positive evaluations. This result means that CI process can improve the Functional suitability of the software quality very well.

In the coding process for test factory, a detailed analysis has been undertaken about how functional suitability has been improved. The relative analysis can be used here. In the table for Coded results for Functional suitability, this point has also been explained by other agile teams.

Table 48 is created to present how functional suitability has been improved within different agile teams.

Table 48: CI process improves functional suitability

Agile organization	**How functional suitability has been improved?**
PBK with SAP	Feedback
Scrum Event (Marco)	High quality of code

	Feedback
Scrum Event (Standard CI)	Various test cases
Daimler Fleetboard	Automated build, automated test and fast feedback.
Daimler Test Factory	Requirement stability(feedback, communication with stakeholder) Test coverage(feedback, communication with stakeholder) Defect leakage(UAT)(feedback, communication with stakeholder) Defect identification(Unit test, regression test, various test cases)

Feedback has been mentioned by almost all agile organizations from interviews. Besides, automated build and test are also very important. Without these two processes, various test cases cannot be reached in a good test quality.

Code2: Performance efficiency

Table 49: Coded results for Performance efficiency

Unit of analysis	**Coded text**
Daimler Fleetboard	Efficiency: through fast feedback. Not only with acceptance test, inside one sprint, will the feedback also be given to stakeholders, so that, the stakeholder can judge, if the software is developed in the right direction.

This evaluation is about efficiency. But after the comparison with the concept of Performance efficiency from ISO/IEC 25010:2011, the research has determined that, the efficiency mentioned by Daimler Fleetboard is not the Performance efficiency of software quality, but the efficiency of getting the decision from stakeholder.

So it can be concluded that, the CI process cannot improve the Performance efficiency of the software quality.

Code3: Compatibility

Table 50: Coded results for Compatibility

Unit of analysis	Coded text
PBK with Daimler	With this CI process with variation, the IBP on cloud can be tested, if it can work with the software On-premise together. So that, the compatibility of the software can be improved.
PBK with SAP	Functions can be often checked with both SAP and Daimler sides

2 evaluations about Compatibility have been collected and coded. From the coded text from PBK with Daimler, a positive evaluation has been given to compatibility. The planned CI process with variation2 for the acceptance test by Daimler side will improve the compatibility. From the coding content from PBK with SAP, "Functions can be often checked with both SAP and Daimler sides" is not an absolute positive evaluation about compatibility. But during the check with both sides, the compatibility can also be assured.

Therefore it can be deduced that, CI process can improve compatibility through and automated acceptance test and fast feedback.

Code4: Usability

Table 51: Coded results for Usability

Unit of analysis	Coded text
Scrum Event (Marco)	reducing complexity
Daimler Test Factory	Requirement stability will be improved through CI.
Daimler Fleetboard	Usability: difficult to say. Because CI is a systematic process without person.
Daimler Test Factory	After integration, the finished parts can be showed to customer in necessary. Then the customer can always check in side one sprint, if the development is going in the right direction. So after the final release, the software will act as exactly as possible with the expectation from customer.

4 evaluations about compatibility have been collected and coded. 3 evaluations are positive, which states that CI process has improved Usability. 1 evaluation is neither positive nor negative. From the statement of this interviewee: “Because CI is a systematic process without person.” It can be judged that, the interviewee holds a negative view that, CI process cannot improve Usability. As a mechanical system, CI process cannot judge, whether Usability has been assured or not.

“Requirement stability will be improved through CI.” In the coding process, it has been analyzed in details that, the high Requirement stability indicates the improvement of appropriateness recognizability and operability of Usability.

According to the analysis from the researcher, through continuously fast feedback of CI process and thus frequent communication with stakeholder, the Usability of the software quality has been improved.

Code5: Reliability

Table 52: Coded results for Reliability

Unit of analysis	**Coded text**
Scrum Event (Marco)	High quality of code.
Scrum Event (Standard CI)	In CI, the frequently implemented automated process has checked the reliability, namely automated build, automated deployment, and automated test. So that, the reliability can be assured.
Daimler Test Factory	Test execution productivity will be greatly improved through CI, because of the high automation process in CI.
Daimler Fleetboard	Reliability: through regression test. When the same code is run through automated test from CI for over 10 times, the reliability can be assured.
Daimler Test Factory	Defect identification will be improved through CI.
Daimler Test Factory	Percent of automation is great improved through CI, because of the automated build /test process in CI.
Daimler Test Factory	Test coverage is improved through CI.

7 evaluations about Reliability have been collected and coded. All of the evaluations are positive. This result indicates that, CI process can improve the Reliability of the software quality very well.

In the coding process for test factory, a detailed analysis has been undertaken about how Reliability has been improved. The relative analysis can be used here. In the table Coded results for Reliability, the improvement of Reliability has also been explained by other agile teams.

Table 53 presents how Reliability has been improved within different agile teams.

Table 53: CI process improves Reliability

Agile organization	**How has Reliability been improved?**
Scrum Event (Marco)	High quality of code.
Scrum Event (Standard CI)	frequently implemented automated process
Daimler Fleetboard	Frequent automated regression test
Daimler Test Factory	Test execution productivity(with the automated build /test process in CI) Defect identification(automated Unit test, Component Test, Regression test and Functional test) Percent of automation is greatly improved through CI(with the automated build /test process in CI) Test coverage(feedback, communication with stakeholder)

The automated process from CI process, e.g. automated build and various kinds of automated test have been mentioned as reason for high reliability by almost all agile organizations from interview. Besides, feedback and communication with stakeholder are also very important to improve the reliability of the software quality.

Code6: Security

No coded evaluation for security has been collected. This result indicates that, CI process cannot affect the security of the software quality.

Code7: Maintainability

Table 54: Coded results for Maintainability

Unit of analysis	Coded text
Daimler Test Factory	Defect Leakage (UAT) will be reduced greatly through CI.
Daimler Test Factory	Test execution productivity will be greatly improved through CI, because of the high automation process in CI.
Daimler Test Factory	Requirement stability will be affected through CI.
Daimler Test Factory	Defect identification will be improved through CI.
Daimler Test Factory	Test coverage is improved through CI.
Daimler Fleetboard	Maintainability has been improved through working with SonarQube.
Daimler Test Factory	Percent of automation is greatly improved through CI, because of the automated build /test process in CI.
Scrum Event (Marco)	High quality of code.

8 evaluations about Maintainability have been collected and coded. All of the evaluations are positive. This result indicates that, CI process can improve the Maintainability of the software quality very well.

6 evaluations about Maintainability are based on the Quality Metric and KPIs from Daimler Test Factory. In the coding process for test factory, a detailed analysis has been undertaken about how Reliability has been improved. The relative analysis can be used here. The other 2 evaluations are from Daimler Test Factory and Scrum Event (Marco).

Table 55 is created to present how Reliability has been improved within different agile teams.

Table 55: CI process improves maintainability

Agile organization	How has Maintainability been improved?
Scrum Event (Marco)	High quality of code.
Daimler Fleetboard	Maintainability has been improved through working with SonarQube.

Daimler Test Factory	Defect leakage(UAT)(few defects have been found in the acceptance test) Test execution productivity(with the automated build /test process in CI) Requirement stability(CI process has helped the test team indirectly to establish test criteria) Defect identification(In CI process, the test phases have been dived different stages) Test coverage(all test cases have been transferred into script being running in CI process) Percent of automation (with the automation build /test process , all established test criteria can be checked efficiently)

The analysis indicates that, automated build/ test process and various test categories in stages in CI process have improved the Maintainability of the software quality a lot.

Daimler Fleetboard mentioned SonarQube, which is a statistical code analysis tool. This statistical code analysis process has improved Maintainability a lot. Statistical code analysis process is a kind of inspection process of CI process. More analysis about statistical code analysis will be undertaken in the part of RQ4 and RQ5 as an optimization for CI process.

Code8: Portability

Table 56: Coded results for Portability

Unit of analysis	**Coded text**
Scrum Event (Marco)	reducing complexity

For Portability only one evaluation has been collected from Scrum Event (Marco). CI process can reduce complexity, so that the Portability of software quality can be improved. The analysis from the researcher is that, CI process builds and tests software in stages, namely from unit test to acceptance test. When the hardware, software, operational, and usage environments have been changed, it is easier to find out and correct the relative code parts to adapt the new requirements within staged tests.

Code9: Software Quality in general

Table 57: Coded results for Software Quality in general

Unit of analysis	Coded text
Daimler Fleetboard	Quality to market is increased.
Scrum Event (Standard CI)	CI has improved the software quality. The theory of CI is to implement automated build, automated deployment and automated test. So that the software can reach a specific quality level.
Scrum Event (Marco)	CI helps the development process in development and quality assurance.
Scrum Event (Standard CI)	Because agile emphasis to reach fast delivery and quality assurance in the software developing. With CI this goal can be reached. Because the purpose of CI and CD is to realize permanent delivery and good quality.

4 evaluations have been coded and collected about software quality in general. These evaluations have not stated the characters from software quality like in the other tables for coded results. One evaluation has mentioned the automated build, automated deployment and automated test as the reason for improving software quality.

RQ3.2: What effects has CI process to Communication

For RQ3.2, the Communication is researched from 2 sub aspects based on additional effects of CI process from Section 2.4.2, which correspond to 2 codes in the coding table for RQ3.2:

(1) Communication with stakeholders;

(2) Communication in development team.

Code1: Communication with stakeholders

Table 58: Coded results for Communication with stakeholders

Unit of analysis	Coded text
Scrum Event (Marco)	Cooperation with stakeholders. Strengthen Communication:

	causes team members to consider each other's work and context
Scrum Event (Standard CI)	The communication between development team, manager, and customer can be improved because of fast feedback. In fact, all the automated process in CI is the basis to give a fast feedback.
Daimler Fleetboard	The communication with customer can be improved. Nowadays software can be released frequently in 1-2 weeks, fewer error will occur during the use in the customer. The customer will be satisfied with the software quality. So the cooperation will be kept at a good level. In the past, 1-2 weeks after release is always very busy to solve the error occurred in customer. Now it is very locker. The reasons are (1) The release has been dived into frequent and small release cycle in CI. In compared with releasing huge whole software without CI. (2) With frequent feedback, most errors have been correctted during the development process, few errors will occur by the side of customer.
Scrum Event (Standard CI)	From the aspects of cooperation: When the development is a cooperation case from twice-side, the cooperation between supplier and customer will be improved through CI. And it is also because of fast feedback.
Daimler Fleetboard	With CI, the software can be presented to the stakeholders, so that, they can give the feedback on the development, if it is god or not. So the development can be assured in a right direction. Stakeholder can give the feedback, if the development has been done good or not. With automated test, the test coverage is higher than manual test
PBK with SAP	Communications can be taken very often, to lead to the right decision.
PBK with SAP	And the stakeholder can also get fast feedback about the progress of configuration, which is very meaningful for the decision on the further development.
Scrum Event (Marco)	In CI, after integration, the software can be delivered to the customer. Trough frequently acceptance test from the cus-

	tomer, the development of software stays as possible as the customer wish.
Daimler Test Factory	Requirement stability will be affected through CI.

8 evaluations about Communication with stakeholders have been collected and coded. All 8 evaluations are positive. Based on the evaluations, how Communication with stakeholders has been improved has been listed in Table 59.

Table 59: CI process improves Communication with stakeholders

Agile organization	**How has Communication with stakeholders been improved?**
Scrum Event (Marco)	Communication on each other's work and context.
Scrum Event (Standard CI)	fast feedback.
Daimler Fleetboard	The release has been dived into frequent and small release cycle in CI. With frequent feedback.
PBK with SAP	Communications can be taken very often. fast feedback.
Daimler Test Factory	Requirement stability(fast feedback and frequent communication)

Fast feedback has been mentioned by most agile organizations as the reason for improving Communication with stakeholders.

Code2: Communication in development team

Table 60: Coded results for Communication in development team

Unit of analysis	**Coded text**
Scrum Event (Standard CI)	The communication between development team, manager, and customer can be improved because of fast feedback. In fact, all the automated process in CI is the basis to give a fast

	feedback.
PBK with SAP	Communications can be taken very often, to lead to the right decision.
Scrum Event (Marco)	Besides, from the fast feedback, the development team can evaluate if the development direction is right. To fix the bug, the developer may exchange the experience within the team, so that the knowledge databank can be built in the development team.
Daimler Fleetboard	CI is also helpful with knowledge transfer. For example, the new college enters a team, with the test results the new college can be trained to solve the problem from test. It is usually the first task.
Daimler Fleetboard	The effects on communication can be improved through fast feedback. With the fast feedback, when the test is red, the developer who has made the error can know very soon that, he has made the error and he should fix it. Problem can be solved soon after it occurs. Daily meeting, at 17 o'clock, all developer teams will meet each other, they can communicate with each other about the cross problems.
PBK with SAP	With CI process, because of the fast feedback, the consultant who is responsible for the configuration can know in the early time, if the configuration has been done correctly or not.

6 evaluations about Communication in development team have been collected and coded. All 6 evaluations are positive. Based on the evaluation, how Communication in development team has been improved has been listed in Table 61.

Table 61: CI process improves Communication in development team

Agile organization	**How has Communication in development team been improved?**
Scrum Event (Marco)	fast feedback, knowledge transfer
Scrum Event (Standard CI)	fast feedback

Daimler Fleetboard	Knowledge transfer, fast feedback.
PBK with SAP	Communications can be taken very often fast feedback

Similar to the analysis in Communication with stakeholder, fast feedback has also been mentioned by most agile organizations as the reason for improving Communication in development team. Besides, fast feedback also leads to fast knowledge transfer and build of knowledge data bank in the team, which promotes good communication in the development team.

RQ3.3 what effects has CI process to development process

For RQ3.3, the Development process is researched from 3 sub aspects based on the additional effects of CI process from Section 2.4.1, which correspond to 3 codes in the coding table for RQ3.3: (1) Time to run build/test; (2) Time for development; (3) Cost for development; a code for development process in general has also been taken in consideration. This code is designed for the evaluation from unit of analysis, which can't be dived to the sub aspects of development process.

Code1: Time to run build/test

Table 62: Coded results for Time to run build/test

Unit of analysis	**Coded text**
Daimler Fleetboard	Before: time to test is 15 persons in 4 weeks Now: 0 person in one evening
Daimler Fleetboard	Before: feedback of test can be given in 4 weeks Now: right after CI process
Daimler Test Factory	Test execution productivity will be greatly improved through CI, because of the high automation process in CI.
Daimler Test Factory	Percent of automation is great improved through CI, because of the automated build /test process in CI.

4 evaluations about Time to run build/test have been collected and coded. All 4 evaluations are positive. The evaluation from Daimler Fleetboard indicates obviously that,

time to build and test is greatly reduced with CI, which leads to fast feedback. From the evaluation from Daimler Test Factory, the reason of improving time to run build/test is described. “Because of the high automation process in CI”, the time to run build/test is reduced.

Code2: Time for development

Table 63: Coded results for Time for development in Development process

Unit of analysis	**Coded text**
Scrum Event (Marco)	fast error correction and bug fixing, developer has more time for developing
Scrum Event (Marco)	In CI, fast feedback can be given; the development team can react as fast as possible on the error. A research has shown that, when an error is fixed on the same day, it will be 240% faster fixed than in 2 weeks later.
Daimler Fleetboard	Time to market has also been reduced.

3 evaluations about Time for development have been collected and coded. All 3 evaluations are positive. Because of fast feedback, the development team can react as fast as possible on the error. The error or bug can be corrected faster. So that, the developer has more time for developing, and time to market can be reduced.

Code3: Cost for development

Table 64: Coded results for Cost for development in Development process

Unit of analysis	**Coded text**
Scrum Event (Marco)	In CI, fast feedback can be given; the development team can react as fast as possible on the error. A research has shown that, when an error is fixed on the same day, it will be 240% faster fixed than in 2 weeks later.
Daimler Test Factory	Percent of automation is great improved through CI, because of the automated build /test process in CI.
Daimler Test Factory	Test execution productivity will be greatly improved through CI, because of the high automation process in CI.

PBK with SAP	The automatically test is very helpful for the project. The configuration test involves with large amounts of data. With automatically test, a lot of manual expenses can be saved.
Daimler Fleetboard	Because the error is found earlier. One error is found after 2 day is cheaper to solve than at end of the development.
Daimler Fleetboard	Before: time to test is 15 persons in 4 weeks Now*:* 0 person in one evening

6 evaluations about Cost for development have been collected and coded. All 6 evaluations are positive. Because of the automation process, on one side, a lot of manual expenses can be saved. On the other side, since time to run build/test can be reduced, the error can be found earlier, which is much cheaper to be fixed than found later.

Code3: Development process in general

Table 65: Coded results for Development process in general

Unit of analysis	**Coded text**
Scrum Event (Marco)	CI helps the development process in development and quality assurance.
Daimler Fleetboard	Improve developing process: through automated build/test/deployment and Fast feedback

2 evaluations about Development process in general have been collected and coded. Both 2 evaluations are positive. Because of the automation process and fast feedback, the developing process can be improved.

RQ3.4 what effects has CI process to Risk management

For RQ3.4, the Risk management is researched without sub aspects. Therefore only one code is designed for the evaluation from unit of analysis on Risk management.

Code: Risk management

Table 66: Coded results for Risk management

Unit of analysis	Coded text
Scrum Event (Standard CI)	From the aspects of risk: The risk of project and product can be reduced through the assured functionality and reliability from CI.
Daimler Test Factory	Defect identification also indicates that; risk of development has been reduced.
Scrum Event (Marco)	Through CI, the possibility of refactoring is reduced. Refactoring happens, when the cost of not refactoring is higher than refactoring. It means, when the development has so many bugs that, the development cannot be further developed or the requirements from the customer cannot be covered. With CI a long term, slight change and extension can be assured. The possibility of refactoring can be therefore reduced. So that, the risk can be reduced.
Daimler Fleetboard	Reducing risk through fast feedback and automated test When the error has occurred 2 days after developing, but been found at the end of the sprint/developed, namely in one month, it harms the performance of the software. It can happen that the error is so harmful that, the whole development has to be refactored. With CI, I can always know what the error is and the error can be fixed as soon as possible. So the rate of refactoring is greatly reduced. And the performance of software is kept in a high level.
Scrum Event (Marco)	low possibility of refactoring
Daimler Test Factory	Effort variance, schedule variance, and cost variance will also be improved through CI.
PBK with SAP	With CI process, each configuration version can be tested in time. So that, risk of wrong configuration or configuration with error can be greatly reduced.

7 evaluations about Risk management have been collected and coded. All 7 evaluations are positive.

How Risk management has been improved has been listed in Table 67.

Table 67: CI process improves Risk management

Agile organization	**How has Communication in development team been improved?**
Scrum Event (Marco)	Through CI, the possibility of refactoring is reduced.
Scrum Event (Standard CI)	assured functionality and reliability from CI
Daimler Fleetboard	Reducing risk through fast feedback and automated test
Daimler Test Factory	Effort variance, schedule variance, and cost variance(stable development process)
PBK with SAP	Risk of wrong configuration or configuration with error can be greatly reduced

The continuous integration build, automated test and fast feedback from CI have reduced the possibility of refactoring, assured the functionality and reliability of the software, and kept the development process in stable. Therefore the risk management in the development process has been greatly improved.

6.2.4 RQ4: How to optimize CI process?

The goal of RQ4 is to research: how to optimize CI process. To answer this research question, literature review about optimization of CI process, interests from agile organization on optimization of CI process, and interviews about CI process in the practice in the previous research from this study have been taken into consideration.

Literature review for RQ4 demonstrates that, automated inspection is an important part for software development. To implement automated inspection, statistical code analysis is a good solution. By implementing this approach in CI process, the reliability, maintainability of the developed software can be greatly improved and the risk of the software development can be accordingly reduced.

Different agile organizations have been asked during the interviews, from all the researched effects of CI process, especially from software quality aspects, what effects are

the most in need to be improved. The interview result shows that, for the most agile organization, functionality and reliability are of the most interest.

During the analysis for RQ2, the difficulty to build CI in agile development process has been analyzed:

(1) The configuration test with the focus on testing configured functions and configured database cannot be efficiently executed with the current standard CI process. CI process can be optimized to reach the goal of configuration test.

(2) Since the acceptance test has high demands on the configuration of testing environment and integration of different database. In the practice, the acceptance test can't be executed within CI process under specific condition. One CI process with variation has been planned to realize the acceptance test.

The difficulties to build CI have been researched as the basis for further optimization.

Based on the above analysis, 2 aspects to optimization CI process have been researched. RQ4 is dived into 2 sub research questions:

RQ4.1: How to optimize CI process with statistical code analysis process?

Daimler Fleetboard has implemented statistical code analysis with SonarQube, a second interview is organized to research, how has SonarQube been implemented in their CI process.

CI process with statistical code analysis is described in the following:

Step1: Developer checked in the Code part A or upload new version of Code A to VCR. This Code part A belongs to Module A.

Step2: Jenkins takes out this Code part A, calling Maven to build Code part A with the other existed parts. When the build is successful, Code part A is successful integrated to Module A.

Step3: In the developing system, the developer operates configurations manually. Module A is also manually deployed. Maven (for build and compile) calls up Unit Test for the whole Module A. The connection between Module A and other modules in Component A will also be tested in the Component test.

Step4: Afterward, Maven calls up Sonar to take the statistical code analysis for this whole Component A.

Step5: The feedback of Sonar can be given through 3 methods:

(1) Per mail or Jawber (Jenkins has record of the ID of person, who has compiled the code).

(2) Superficial screen, e.g. with a web browser.
(3) Connected with Eclipse, to show the warnings directly inside the code parts.

Step6: If the results from Sonar are critical, this Component A with the piece of code A will not be deployed for the following acceptance test and regression test. If the results from Sonar are ok, then this Component A will be shifted for automated deployment for acceptance test and regression test.

Step7: The automated deployment begins every noon, from about 12:00 to 15:00. To prepare for the following acceptance test and regression test. The configuration management will be undertaken with Python script, for example, the database will be freshly set up. There are also some special configurations, for example, Configuration for web4. The configuration lasts usually until 15:00 to 17:00.

Step8: Then the acceptance test and regression test are usually running for the whole evening, until morning on the next day.

The statistical analysis serves as the control for code quality before acceptance test and regression test. It is of significant benefits to be integrated into CI process.

RQ4.2: How to optimize CI process to improve Functional suitability and Reliability from software quality?

From the analysis of functional suitability and reliability in RQ3, it can be concluded that, automated build, automated test and fast feedback in CI process can improve the functional suitability and reliability. The previous research from this study demonstrates that, to improve functional suitability and Reliability a step further, more automated regression test, functional test and acceptance test should be implemented in CI process.

How to implement automated regression test, functional test and acceptance test in CI process? 3 optimization methods have been proposed.

Optimization 1: To implement automated regression test in CI process

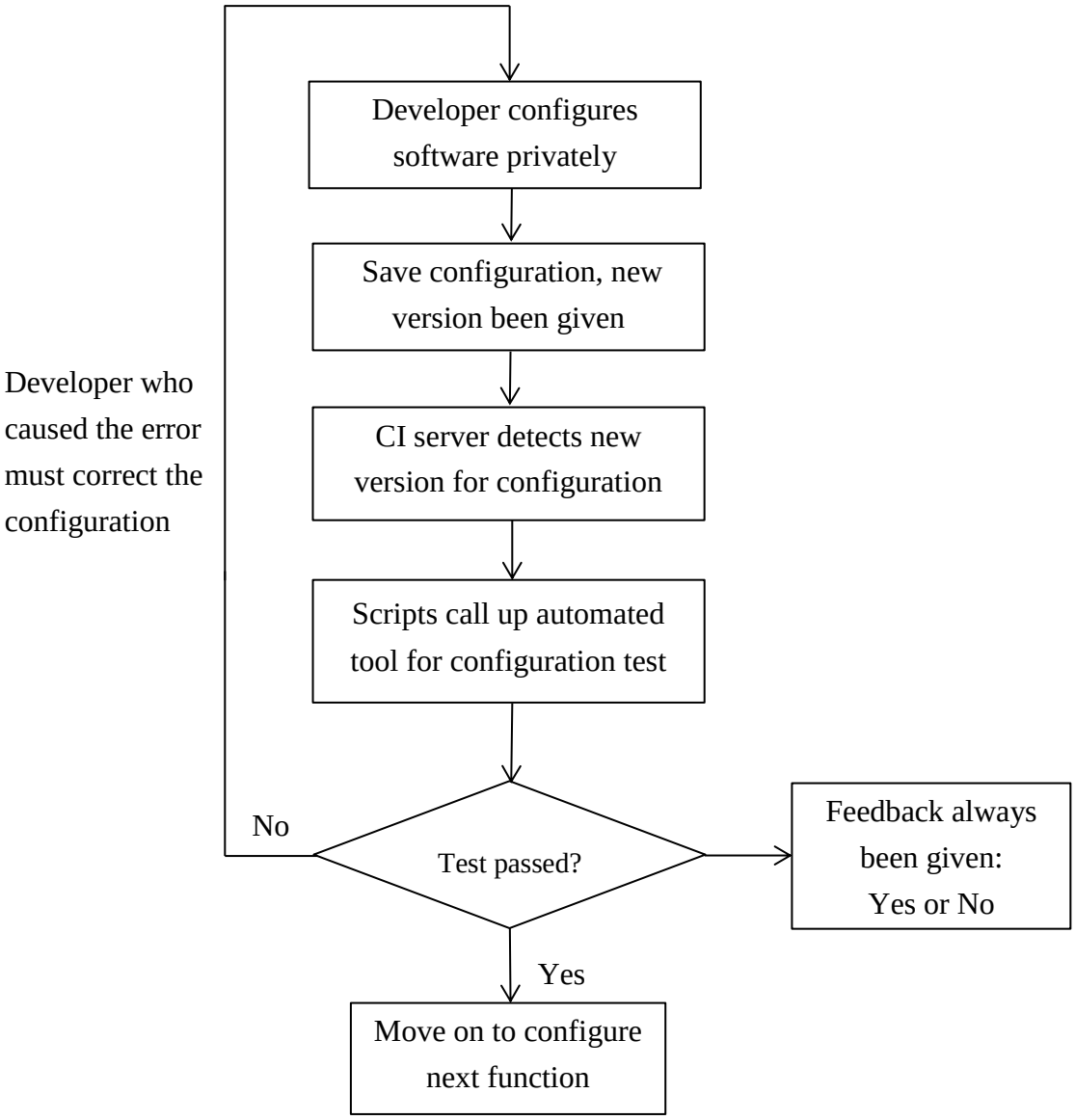
With the interview with “PBK with SAP (Ralf)”, a CI process for automated regression test for configuration of SAP IBP for project PBK has been built, which is presented in Table 68.

Table 68: Optimization 1 of CI process to implement automated regression test

Unit of analysis	Optimization for CI process
PBK with SAP(Ralf)	(1) Each consultant configures SAP IBP privately. After each configuration, a version will be automatically given. From the configuration history, the person who has taken configurations can also be retrieved. (2) A VCR tool is not necessary. The version control can be done manually by each consultant after private configuration. (3) Jenkins can be used as CI Server. As the test for configuration is functional test. The test goal cannot be reached only with Jenkins. But a script can be written, so that, Jenkins can call up this test tool to run the automated test for configurations. A trigger can be set to call up the test tool. For example, a manual trigger, this depends on the decision of each consultant. Or choosing a fixed time trigger, e.g. once a day in the evening. (4) Feedback can be given, either real time on the test screen or by Jenkins per mail after finishing the test.

The most important part of this CI process is that, a script has been written, in order that Jenkins can call up the self-defined automated test tool to run the automated test for configurations.

Based on this automated regression test for configuration test from the interview, and the analysis of the researcher, a flow chart in Figure 36 for the optimization of CI process has been built.



(Source: Own illustration)

Figure 36: Optimization 1 of CI process to implement automated regression test

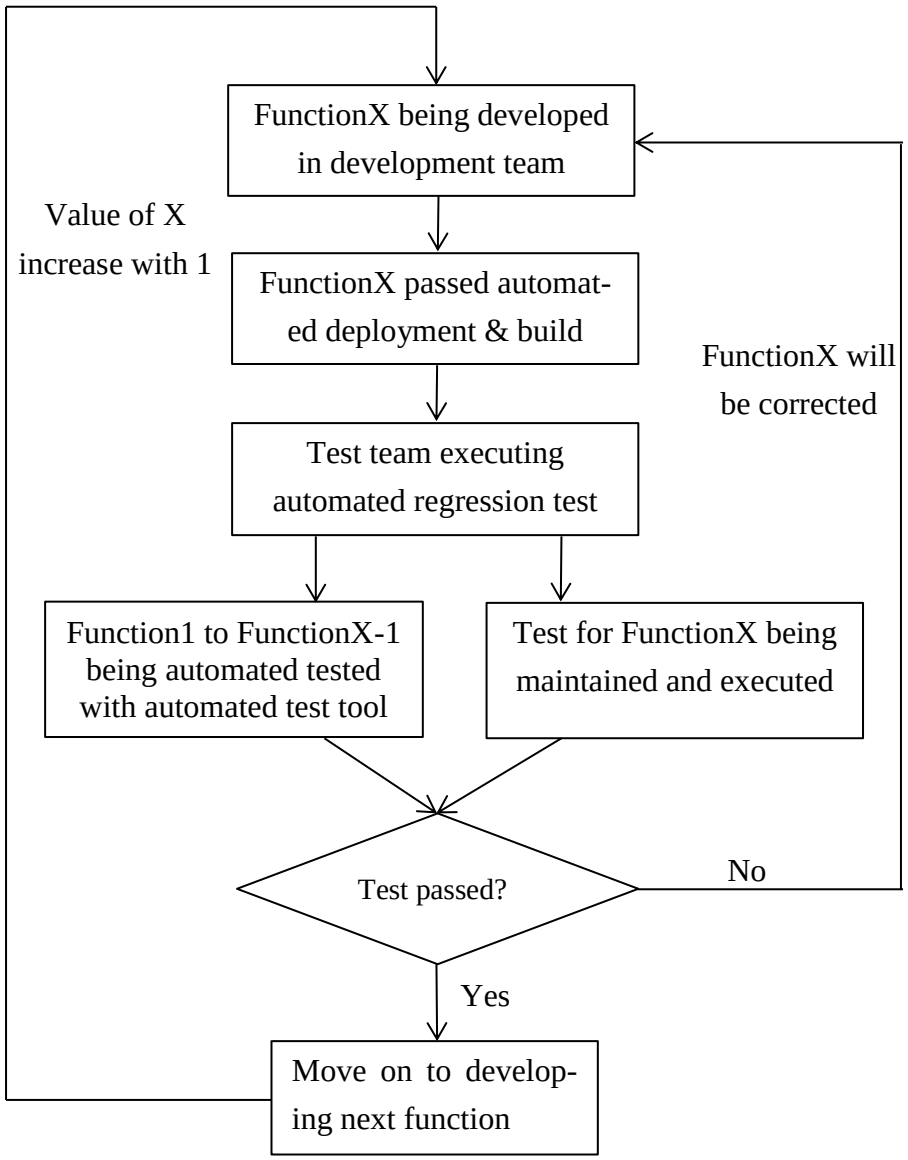
Optimization 2: To implement functional test in CI process

With the interview with "Daimler Test factory", a CI process for functional test has been built, which is described in Table 69.

Table 69: Optimization 2 for CI process to implement functional test

Unit of analysis	**Optimization for CI process**
Daimler Test Factory	Code is automatically built with Jenkins. As in the project no access to Jenkins is available, a manual trigger is needed. But the test team is trying to build the automated trigger. After the manual trigger, Functional test is executed with CI in variation.
Daimler Test Factory	The Function X is new developed and has passed the automated deployment and build. Now the Function X should be tested, whether it can work together with all the existed functions from Function1 to FunctionX-1. During the regression test, Function1 to FunctionX-1 will be automated tested with automated test tool. In parallel the test for new Function X will be manually step by step maintained and executed. If an error happens during the test, the feedback will be given to the development team to correct the relative defected parts. If the tests have been successfully passed, the development moves on. When the Function X+1 has passed the automated deployment and build, the Function1 to Function X will be automated tested, and in parallel, the FunctionX+1 will be manually tested. After test, the relative feedback will be given.

This process has been presented with the flow chart in Figure 37.



(Source: Own illustration)

Figure 37: Optimization 2 for CI process to implement functional test

Optimization 3: To implement acceptance test in CI process

Since acceptance test happens usually by the custom side, which usually has the particular development and productive environment. To execute the acceptance test on a standard CI server for the custom side is not enough sometimes. In the Acceptance test for PBK, the standard CI server such as Jenkins is not used, but the own integrated test tool SAP Solution Manager is being used. To adjust to this specific "CI server", the standard version control like SVN is also not suitable. The Change Request management and transport management have been used as Version control. CI process should be optimized to fulfill the requirements of acceptance test. Through implementing more acceptance test in CI process, the reliability will be greatly improved and risk can be reduced.

6.2.5 RQ5: To validate the optimization of CI process

In RQ5, the optimization of CI process from RQ4 has been validated, whether the optimization will improve the relative aspects.

RQ4 is divided into 2 sub research questions:

RQ4.1: How to optimize CI process with statistical code analysis process?

RQ4.2: How to optimize CI process to improve Functional suitability and Reliability from software quality?

The validation of the optimization is correspondingly divided into 2 parts:

RQ5.1: Validation of the optimization with statistical code analysis process

During the interview with Daimler Fleetboard, a comparison has been taken between the current CI process with and without SonarQube in the past. The differences between these two CI processes are described in the following.

(1) If the results from Sonar are critical, the component A will not be deployed. All developers, who have checked the code, will get a feedback. The parts with error have to be corrected. This control before regression test and acceptance test is very helpful for the development process. Without this control of SonarQube, a large amount of errors would appear in the regression test and acceptance test, which results in an extremely long test phase sometimes. Therefore the statistical code analysis with SonarQube improves the development process.

(2) For 100% test coverage, it would be same between with and without Sonar. But it is too expensive to reach the goal of 100% test coverage. To achieve this goal all the criteria should be contained in one combination, which is very expensive. From the econom-

ic point of view, it is unworthy to reach 100% test coverage in the practice. Since there is no 100% test coverage in the practice, it is very meaningful to have Sonar. The maintainability of software will be substantially increased.

Without statistical code analysis, only errors which occur in the test can be detected, which report incorrect functionality. After integrating statistical code analysis to CI process, not only incorrect functionality but also design errors, e.g. dependence on the other code parts, wrong grouping, anti-pattern error, security, e.g. SQL injections, potential error, e.g. variable denies can also be detected.

Besides, the criteria of the statistical code analysis can be set up in the profile according to the specific requirements in the practice. For example, what kind of error should be detected, under what condition the results of analysis are critical and so on.

In a word, through statistical code analysis, e.g. SonarQube, the code quality has been greatly improved. The improved code quality leads to faster error fixing and better function of the software. Therefore, the maintainability and reliability of the software have been improved to a large extent.

RQ5.2: Validation of the optimization to improve Functional suitability and Reliability from software quality

Validation 1: Implementing automated regression test in CI process

Positive side

With the script from "PBK with SAP (Ralf)", CI process can be not only used for Unit test and component test to check the program errors, but also the errors in the functional level. With this script, the development team has more freedom in choosing the method to validate the results of the test. In "PBK with SAP (Ralf)", the developed product IBP is integrated as a function in excel. With the script, the development team can compare the "Is-data" and "Should-data" in excel, to check if the configured function is the one in need.

This CI process with variation is also suitable for configuration test. As configuration test is not like unit test, there is no large amount of coding change, but more changes of functional combinations. Therefore there is no need to upload the code to the version control repository and test the code on the standard CI server. For configuration test, the most important is to check, if the newly configured function still works with the already validated functions. So that, to test the configuration on the particular self-defined environment with the specific validation method is very important.

With a script to connect the standard CI server with particular test purpose is a perfect solution for configuration test. On one side, the continuous integration build has been realized, which improves the reliability of the software greatly. On the other side, the specific test goal for function configuration and database configuration can also be reached, which facilitates the functional suitability highly.

Negative side

But without standard version control repository, when more developers works in the same time on the configuration. By "save" collaboration may happen. For IBP, as it is a cloud product. All configured functions have been saved automated real time on the cloud. But for a non-cloud product, with a manual version control, collaboration and loss of configuration may happen. In this respect, the risk of the development is being raised.

Instead of uploading the whole code to be tested in the standard CI server, using a script embedded in CI server to call up automated test tool contains more uncertainty. When the test results are not accepted, without further inspection, it cannot be concluded that, the regression test for the configured function has failed. Since the unexpected results may locate in the failed invocation of the test tool. From this perspective, the difficulty to detect error has been increased; time and cost for development have been increased.

Since the test process is defined step by step in the automated tool. Along with the development, the test process needs to be adjusted. Without automated build, test and deployment from a standard server, there exists great expense to adjust the test tool.

Validation 2: Implementing functional test in CI process

Positive side

CI process with variation in "Daimler test factory" is similar to CI process in "PBK with SAP (Ralf)", an extra trigger is needed to execute functional test. As in the argument above, it is a flexible method for automated regression test and functional test. The difference is, to test function regressively; "Daimler test factory" has implemented specific method for functional test. This method is described in details from the coding table and the flow chart. This method assures a very high test coverage, in which each newly developed function is tested with the already validated function from the previous test cycles. Consequently, the Functional suitability and Reliability can be assured.

During a test cycle for a new function, the new function is tested manually step by step. When an error happens, the test can be stopped soon and the part with error can be fixed in time. Therefore the risk of system crash because of error can be avoided.

Negative side

After the successful automated build on the standard CI server (Jenkins), a manual trigger is needed to start the functional test. Manual expenses and time cost are higher than a complete automation process. “Test Factory” planned to build the connection between Jenkins and the functional test, in order to realize a complete automated build and test process.

The new function is tested manually step by step. This method reduces the risk of system crash. But manual test causes much more manual expenses and time cost than automated test. A more serious problem is that, manual error cannot be avoided, which cause higher risk for the development process.

Validation 3: Implementing acceptance test in CI process

The optimization of CI process with acceptance test, which has been described in Optimization 3 in RQ4, has not been built during the research phase for this thesis. The reasons are: on one hand, the current CI process accepted in the public is built based on a standard CI server. Such a CI Server within specific system environment has rarely been researched. The theory and the guidelines to build such CI process are not mature. On the other hand, because of the technical reasons and project plan, “Daimler with PBK” couldn’t implement this CI process in the practice during the research phase for the master thesis.

Thence, this optimization cannot be validated yet.

7 Discussions

This section discusses the research results presented in section 6 and relates them back to earlier research literature at first. The next, it explains how far the results support the hypothesis and/or answer research questions. At the end, it demonstrates the understanding of the researcher about the limitations of the research.

7.1 Discussions on Research Questions

The discussion on research questions has been undertaken from 3 points:

Point 1: Build up CI process in the practice

In RQ1, Quantitative data analysis indicated that, almost all interviewees have implemented CI process with version control, automated build and automated test in their development process. Similar results occurred in the qualitative analysis for Standard CI process. But in the qualitative analysis, not only standard CI process but also CI process with variation has been researched. Why CI process with variation exists has been explained in RQ2. Since some agile projects face difficulties to build standard CI process:

(1) It is difficult to implement configuration test and acceptance test. This result has also been proved and explained in the study for "Purpose of CI process" in RQ1.

Unit test is the most popular purpose of CI process. A half CI process has the purpose to realize system test, functional test and component test. CI process has rarely been implemented to realize acceptance test. To realize functional test, not only code part should be taken into account like in unit test, but also the specific test case should be checked, whether the specific function is fulfilled. So it is much more difficult to realize Functional test than Unit test in CI process. It is also the reason for few purposes for configuration test and acceptance test.

Therefore, to implement Functional test, configuration test and acceptance test, CI process with variation is needed.

(2) CI process cannot cover the requirement analysis in software development process. A project with high demand on requirement analysis like project PBK can only imple-

ment a CI process with variation, in order to bring the advantages of CI process into the requirement analysis process.

Based on the challenges to build CI process in agile projects, RQ4 has been researched further to find out optimizations of CI process to improve software quality. CI process with variation in RQ1 has been used as foundations to build the optimizations of CI process.

In conclusion, to be more widely used in the practice and to bring more value to the software development process, CI process should be designed more flexibly to adjust to different kinds of agile projects.

Point 2: Usage of CI process in the practice

In the research for "Frequency of CI process" in RQ1, only a half CI process has been executed after every commit change, which has been actually defined in the concept of CI. A lot of CI processes have been triggered depending on the actual needs. This situation reflects a non-positive use of CI in the practice. In fact, CI process should be executed after commit change as in the definition. Only in this way, the benefits of CI process can be fully performed. This situation also indicates the difficulty to implement CI by side. In the agile organization where CI is not executed after every commit change, maybe they don't need such frequent build integration at all, or such frequent build integration is too expensive for them to implement.

This situation can be further explained through the comparison of the background of interviewees and respondents to questionnaire. Among interviewees, Daimler Fleetboard and Daimler Test factory have implemented professional CI process and also provided excellent evaluation about CI process from all 4 aspects. In Comparison, Project PBK in Daimler faces difficulties to build standard CI process. Most of the respondents to the questionnaire are software developers. 14 from 16 respondents have implemented CI process in the development.

During the interviews, a lot of interviewees regard automated test as CI or vice versa. Through the research result for "Relationship between CI process and effects" in RQ3, this phenomenon can be explained. Research result indicates that, apart from CI server, automated test has promoted vast improvement on the Software quality, Development process and Risk management. Therefore, from all components of CI process, automated test has the most important function. Therefore, Daimler Fleetboard which has enormous amount test tasks and Test factory which provide test service for over 20 agile projects have obtained great success through CI process. In contrast, project PBK in

Daimler is focusing on the configuration of standard software, which has much less amount of test cases which can be tested automatically.

In conclusion, CI process is with more benefits to be implemented in agile project or organization, which has more demand on frequent tests, development with most coding tasks.

Point 3: Effect of CI process on software quality

The analysis for sub aspects of software quality indicates that, CI process has no effect on Performance efficiency and Security, While CI process has very good effects on Functional suitability, Reliability and Maintainability. The results can be judged superficially from the number of evaluations, for efficiency and Security no evaluation has been collected. While for Functional suitability, Reliability and Maintainability over 7 positive evaluations have been collected for each sub aspect.

Among of all CI effects, CI process can be optimized to improve Functional suitability, Reliability and Maintainability a step further, which has been explained in RQ4. For example, in Daimler Fleetboard, through implementing statistical code analysis with the tool SonarQube, the Maintainability can be greatly improved. SonarQube can be related to automated code inspections, which has been introduced as in the Section 2.3.3 Processes in CI Server. In the most of the standard CI processes studied in this study, no statistical code analysis has been implemented. If the agile project has enough technical knowledge about CI process, statistical code analysis can be implemented in CI process. Even better, the development team can set up their own rules of statistical code analysis, for example, if the results of statistical code analysis are critical, no further development and testing can be continued. Only if the relative error or error warning have been removed, the development can move on. In this way, the Reliability and Maintainability of the software can be improved remarkably.

7.2 Discussions on threats during the research

One of the biggest threats during this case study is that, the data source of the quantitative data cannot be assured. If 2 respondents from the same agile team have filled in the questionnaire, the data has been repeatedly collected, which leads to statistical error. The researcher has tried to avoid this phenomenon by inviting the most participants personally from social network.

In addition, it cannot be assured that, all responds have understood the questionnaire questions well und thus given the appropriate answer. In the questionnaire, the respondents are required to provide an evaluation of the effects of CI process. Since no explana-

tion to the relative evaluation category has been given in advance, no uniform standard for the evaluation has been provided. As a result, the respondents may have filled in the evaluation based on their subjective speculation. To make sure that, the respondent understands the purpose and background of the questionnaire clearly, a brief explanation about the questionnaire has also be presented. Even though, the statistical error cannot be entirely avoided because of misunderstanding of the questionnaire questions.

In the data analysis for software quality the sub aspects are not exactly the same between qualitative data analysis and quantitative data analysis. The sub aspects in qualitative data analysis are (1) Functionality; (2) Reliability; (3) Usability; (4) Maintainability; (5)Efficiency; (6)Portability; (7)Reduce error rate. In quantitative data analysis the sub aspects are (1) Functional suitability; (2) Performance efficiency; (3) Compatibility; (4) Usability; (5) Reliability; (6) Security; (7)Maintainability; (8) Portability.

Security and Compatibility occurred only in qualitative data analysis, which are the new characteristics in the new quality model. Since this study is an inductive study, the researcher has updated the quality model from ISO 9126 to ISO/IEC 25010:2011 during the cases study, after detecting the more appropriate quality model. At that time, the questionnaire has been already published. Therefore no answers about Security and Compatibility have been collected through questionnaires.

Reduce error rate appeared only in quantitative data. When defining this sub aspect, the researcher has treated this sub aspect as the Functional correctness of Functional suitability, which appeared later on in the new quality model. Thence the data for this aspect has been actually collected by both qualitative and quantitative data.

8 Conclusions

This study is aimed to research CI process in the agile practice. Through an embedded case study, 7 semi-structured interviews and 1 questionnaire were conducted to study the case. Both qualitative data analysis and quantitative analysis haven been performed to the answer the 5 RQs from Section 3.2.

For RQ1: how is CI built in the agile development processes, both quantitative data and qualitative have been analysed. Quantitative data analysis focused on displaying the questionnaire results with numbers in bar charts, thus to explain the general phenomenon of building CI process. Qualitative data analysis focused on demonstrating the interview results in the way of coding table, in order to present more details about how CI process built in different agile organizations.

RQ2: the challenges to build CI in agile development processes were researched only with qualitative data analysis. RQ2 was analysed with the interview results and with research results from RQ1. The research result of RQ2 was further the cornerstone of research for RQ4.

For RQ3: what effects has CI to agile development processes, both quantitative data and qualitative have been analysed. Quantitative data analysis indicated the evaluation of CI effects in bar charts. Qualitative data analysis presented the evaluation of CI process in more details and the provided the reasons of the CI effects based on the particularly built CI process from different agile projects.

RQ4: how to optimize CI process, which was researched only with qualitative data analysis. Proposals of optimization have been originated from literature review, previous research results from RQ1, RQ2 and RQ3, as well as from interview results.

RQ5: to validate the optimization of CI process in the practice, the proposal of optimization from RQ4 has been validated with interview results and the analysis of researcher.

In conclusion, CI process has been used widely in agile projects from the case study and has brought good influence to Software quality, Communication, Development process and Risk management. CI process can be optimized to fulfil the specific requirements based on the actual situation from different projects.

8.1 Implications for practitioners

This study presented how CI process is built in the practice, which can be used as checklist by agile projects that are about to implement continuous integration in the software development process. The CI effects which are researched in the study can also give software organizations insight into the benefits of implementing CI process on Software quality, Communication, Development process and Risk management in the software development.

A step further, this study also demonstrated the challenges to implement CI process in different agile projects, and how to optimize CI to fulfil the specific demands from the agile projects. The optimization proposals have also been validated. For the agile projects that face similar challenge and problems, the research results can be referenced, and some of the barriers that may arise can be avoided.

In a word, this study has great significance for practitioners. From the study results, agile projects can obtain knowledge about building their own CI process based on the actual demands. From the experience from researched agile projects, practitioners can try to avoid the problem may arise in implementing CI process and simultaneously learn from the valuable experience to build appropriate CI process.

8.2 Contributions to academic research

This study filled the vacancy to the existing academic knowledge within software engineering. This thesis has not only dealt with the lack of studies on continuous integration as a research field but also the lack of studies related to important indicators during software development, such as software quality, communication, development process and risk management.

As opposed to prior research on continuous integration, this study adds a more focused description of building up CI process in the practice, challenges that agile organizations face when adopting continuous integration, and optimizations of CI process to fulfil the needs of practical applications. This insight has not received much attention in prior research, according to the researcher's best knowledge [25].

Additionally, the effects of the adoption of continuous integration have been investigated. Though previous research, a rough summary has been given that, CI process can reduce integration problems and allow a team to develop cohesive software more rapidly [7]. But this topic has not been studied intensely. Besides, in the prior research, what effects has CI process on software quality has been less studied.

This study has conducted a detailed study on adopting a continuous integration process from these fields. The study results are presented in more details in section 6.

8.3 Future work

First of all, this case study focused on studying CI process from questionnaire and interview results. Most results come from data analysis and answers from respondents. It would be interesting to take the research through taking measurements for some parameters directly from agile projects. For example, take measurements of the time and cost for development before and after adopting CI process separately, in order to make a comparison between with and without CI process. So that, the effect of CI process could be more convincing.

Moreover, further studies on how to optimize CI process to facilitate more acceptance test in CI process should be continued. During the research on this optimization proposal, because of the technical and time limitations, this optimization cannot be researched deeply and the validation cannot be done. To the author's best knowledge, little research has been done in the area of CI process optimization. It would also be beneficial to investigate further how CI process can be optimized to meet the needs of practical applications in industry better.

Additionally, looking at results from this research, it would be interesting to investigate further how to implement requirements engineering in CI. Project PBK from Daimler is an example of agile projects, in which requirements analysis is the primary task of the project work. The current CI process improves software development process significantly on the aspects of Development, Implementation, Testing, and Update planning. If CI process could cover the field of requirements engineering, the vacancy of implementing CI process in the software development process — Requirement Analysis can be filled. Consequently, CI process could be applied in the whole software development lifecycle.

More significantly, as the current feedback mechanisms from CI process focus on providing feedback about coding error to developer. This kind of feedback would help the developer team to fix errors very fast. But the stakeholder cannot get much meaningful information from that to make further decision. After including the field of requirements engineering into CI process, a future research can be taken to optimize CI process, in order to give feedback related to coverage of the requirements. Even better, similar to the analysis process of CI effects, CI process could give a feedback on the software quality of the developed requirement. Thereby, from one side, the time and cost to translate the feedback on coding level from developer team to stakeholders can

be saved. From the other side, the stakeholders can also get significant information directly from CI process, which would expand the scope of CI usage.

In the end, Duvall et al have mentioned an integration button in the book “Continuous Integration Improving Soft-ware Quality and Reducing Risk”. An integration button in CI process can make the build a non-event, which includes many processes to automate the build. By pushing the “integration” button, all the processes of compiling, rebuilding a database with test data, running tests, inspection, deployment, and providing feedback will be executed automated. But the current CI processes implemented in all researched CI development process haven’t achieved the goal of “integration” button. Different CI process has to be executed one by one, even manually. Therefore future research can be undertaken about, how to build “integration” button in the practice of agile development process. Thus, CI process can be easily implemented and applied by the agile projects. Even better, instead of exchange between different software tools in the current CI process, e.g. from version control tool to automated testing tool, with such an “integration” button, manual error and cost during the development process and the expense for different tools can be greatly reduced.

Literature Reference

[1] Olsson, H.H., Alahyari, H. and Bosch, J. Climbing the "stairway to heaven" – a multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. pp. 392-399, Sept 2012.

[2] Duvall, P.M., Matyas, S. and Glover, A. Continuous Integration Improving Software Quality and Reducing Risk. pp. 3-21, June 2007.

[3] Goodman, D. and Elbaz, M. "it's not the pants, it's the people in the pants" learnings from the gap agile transformation what worked, how we did it, and what still puzzles us. In Agile Conference (AGILE), pp. 112-115, Aug 2008.

[4] Miller, A. A hundred days of continuous integration. In Agile Conference (AGILE), pp. 289-293, Aug 2008.

[5] Miller, A. and Carter, E. Agility and the inconceivably large. In Agile Conference (AGILE), pp. 304-308, Aug 2007.

[6] Lacoste, F.J. Killing the gatekeeper: Introducing a continuous integration system. In Agile Conference (AGILE), pp. 387-392, 2009.

[7] Fowler, M. Continuous integration @ONLINE, Available from: http:// martinfowler.com/articles/continuousIntegration.html. [Accessed 26. August 2016]

[9] Runeson, P., Höst, M. and Rainer, A. et al. Case study research in software engineering: guidelines and examples. March 2011

[10]Yin, R.K. Case Study Research: Design and Methods, 3rd edition. SAGE Publications, 2003.

[11]ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Joint Technical Committee ISO/IEC JTC 1, March 2011

[12] Robson, C. Real world research, 2nd edition. Blackwell, pp. 459, 2002.

[13] Kasurinen, J., Taipale, O. and Smolander, K. Test case selection and prioritization: risk-basedor design-based? In Proceedings of the 4th International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 10:1–10:10, 2010

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

[15] Agile software development. Available from:

https://en.wikipedia.org/wiki/Agile_software_development. [Accessed26. August 2016]

[17] Abrahamson, P., Salo, O. and Ronkainen, J. et al. Agile software development methods: Review and analysis. 2002.

[19] Analyzing the Data--How to analyze your survey data like a survey scientist. Available from: https://www.surveymonkey.com/mp/how-to-analyze-survey-data/. [Accessed 26. August 2016]

[20] Qualitative data analysis software. Available from: http://provalisresearch.com/products/qualitative-data-analysis-software/. [Accessed 26. August 2016]

[21] Agile Alliance. (2013). What is Agile Software Development?. Available from: https://www.agilealliance.org/agile101/what-is-agile/. [Accessed 26. August 2016]

[22] Agile Alliance. (2013). Manifesto for Agile Software Development. Available from: https://www.agilealliance.org/agile101/the-agile-manifesto/. [Accessed 26. August 2016]

[23] Agile Alliance. (2013). 12 Principles Behind the Agile Manifesto. Available from:

https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto. [Accessed 26. August 2016]

[24]Extreme programming. Available from: https://en.wikipedia.org/wiki/Extreme_programming. [Accessed 26. August 2016]

[25] Debbiche, A. and Diener, M. Assessing challenges of continuous integration in the context of software requirements breakdown: a case study. Thesis (Master). Chalmers University of Technology, 2014.

[26] Regression testing. Available from:

https://en.wikipedia.org/wiki/Regression_testing. [Accessed26. August 2016]

Appendix A: Documents for Questionnaire and Interview

In the following are the questionnaire questions in word, interview questions and protocols for all interviews.

A.1 Questionnaire questions in word

Questionnaire about Continuous Integration

Lingwei Li ITP/PL, Daimler AG

1. Have you ever used Continuous Integration in the development?

☐	Yes	☐	No

If Yes, please answer the question from 2 to 5.

2. Do you have the following process in your Continuous Integration? Please mention the relative tool for the process (multiple choices).

Process					Tool for the process
1.Version control	☐	Yes	☐	No	
2.CI server	☐	Yes	☐	No	
2.1Automated build	☐	Yes	☐	No	
2.2Automated deployment	☐	Yes	☐	No	
2.3Automated test	☐	Yes	☐	No	
3.Feedback Mechanism	☐	Yes	☐	No	

3. In which test category you have used Continous Integration?

1.Three test phases	☐	Yes	☐	No
1.1Unit test	☐	Yes	☐	No
1.2Component test	☐	Yes	☐	No
1.3System test	☐	Yes	☐	No
2.Regression test	☐	Yes	☐	No
3.Functional test	☐	Yes	☐	No
4.Acceptance test	☐	Yes	☐	No

4. How often is Continuous Integration be run during development?

☐	after every commit change	☐	≥1 time per day
☐	≥1 time per week	☐	it depends

5. Has Continuous Integration improved following aspects during software development? If possible. Please also mention the process from CI that brings this effect.

1.Improve software quality assurance	☐	Agree	☐	Neutral	☐	Disagree	Process
1.1Functionality assurance	☐	Agree	☐	Neutral	☐	Disagree	
1.2Increase Reliability	☐	Agree	☐	Neutral	☐	Disagree	
1.3Increase Usability	☐	Agree	☐	Neutral	☐	Disagree	
1.4Increase maintainability	☐	Agree	☐	Neutral	☐	Disagree	
1.5Increase efficiency	☐	Agree	☐	Neutral	☐	Disagree	
1.6Increase portability	☐	Agree	☐	Neutral	☐	Disagree	
1.7 Stability	☐	Agree	☐	Neutral	☐	Disagree	
2.Improve communication	☐	Agree	☐	Neutral	☐	Disagree	
3.Improve cooperation	☐	Agree	☐	Neutral	☐	Disagree	
4.Improve development process	☐	Agree	☐	Neutral	☐	Disagree	
4.1Reduce time to run build	☐	Agree	☐	Neutral	☐	Disagree	
4.2Reduce time for development	☐	Agree	☐	Neutral	☐	Disagree	
4.3Reduce cost for development	☐	Agree	☐	Neutral	☐	Disagree	
5.Reducing risk	☐	Agree	☐	Neutral	☐	Disagree	
5.1Reduce rate of refactoring	☐	Agree	☐	Neutral	☐	Disagree	

If No, please answer the question from 6 to 9.

6. Have you ever used following process in the development? Please mention the relative tool for the process (multiple choices).

Process					Tool for the process
1.Version control	☐	Yes	☐	No	
2.Automated build	☐	Yes	☐	No	
3.Automated deployment	☐	Yes	☐	No	
4.Automated test	☐	Yes	☐	No	
5.Feedback Mechanism	☐	Yes	☐	No	

7. In which test category you have used Continous Integration?

1.Three test phases	☐	Yes	☐	No
1.1Unit test	☐	Yes	☐	No
1.2Component test	☐	Yes	☐	No
1.3System test	☐	Yes	☐	No
2.Regression test	☐	Yes	☐	No
3.Functional test	☐	Yes	☐	No
4.Acceptance test	☐	Yes	☐	No

8. How often is automated build/deployment/test be run during development?

☐	after every commit change	☐	≥1 time per day
☐	≥1 time per week	☐	it depends

9. Does the above mentioned process effect the development from the following aspects? Please note the specific process which brings this effect.

1.Software quality	☐	Agree	☐	Neutral	☐	Disagree	pro-cess
1.1Functionality assurance	☐	Agree	☐	Neutral	☐	Disagree	
1.2Increase Reliability	☐	Agree	☐	Neutral	☐	Disagree	
1.3Increase Usability	☐	Agree	☐	Neutral	☐	Disagree	
1.4Reduce error rate	☐	Agree	☐	Neutral	☐	Disagree	
2.Improve communication with team and stakeholders	☐	Agree	☐	Neutral	☐	Disagree	
3.Improve development process	☐	Agree	☐	Neutral	☐	Disagree	
4.1Reduce time to run build	☐	Agree	☐	Neutral	☐	Disagree	
4.2Reduce time for development	☐	Agree	☐	Neutral	☐	Disagree	
4.3Reduce cost for development	☐	Agree	☐	Neutral	☐	Disagree	
4.Reducing risk	☐	Agree	☐	Neutral	☐	Disagree	
5.1Reduce rate of refactoring	☐	Agree	☐	Neutral	☐	Disagree	

A.2 Interview questions

Interview in agile development process

Lingwei Li ITP/PL, Daimler AG

1. Has Continuous Integration been used in the development process?

If Continuous Integration has been used

2. How has the Continuous Integration process been built in the process?
    a. What tool is used for Version Control Repository to control the version? For example, Subversion?
    b. What tool is used for automated build and test? For example, Jenkins, Hudson?
    c. How feedback is given?
    d. How often Continuous Integration is being run in the development process? For example, once per day?
3. What kind of effects has Continuous Integration brought to the development process? Positive or negative? When possible, please include why.
    a. From the aspect of software quality, can Continuous Integration bring better software quality? For example, from functionality? Reliability? Usability?
    b. From the aspect of Communication?
    c. From the aspect of development process?
    d. From the aspect of reducing risk to the project? For example, reducing the risk of refactoring?

If Continuous Integration has not been used

4. How is the test process being built in the development process? Has automated test been used?
5. What automated test tool is being used?
6. Beside automated test tool, are version control, automated build and feedback also involved?
7. What kind of effect has automated test brought to the development process? For example, improving software quality, reducing risk?

A.3 Protocols from all interviews

The names of all 7 interview protocols are listed in the following:

1. Protocol for Daimler Fleetboard
2. Protocol from Interview with test factory
3. Interview about a standard CI process
4. Interview about variations in CI process
5. Interview with SAP in project PBK
6. Interview with software developer from SAP IBP
7. CI in NTT

All interview protocols can be found in the attachment of the master thesis.

Appendix B: Coding tables

After coding process with the qualitative data analysis tool, the coding results are demonstrated in the coding table, which has been widely used to analyze RQs in Section 6 Results. There are 4 coding tables in total, which have been listed in the following:

1. General Information.xls
2. RQ1.xls
3. RQ2.xls
4. RQ3.xls

"General Information.xls" contains information about Interviewee, interviewed project and organization and parallel process to CI.

"RQ1.xls" included all codes for RQ1, for example, standard CI process, purpose of CI process and so on.

Similarly, "RQ2.xls" and "RQ3.xls" included the coding results for RQ2 and RQ3 respectively.

Since the coding tables are excel documents with large amount of data, which cannot be fully and properly presented in the appendix. Therefore, all coding tables can be found in the attachment of the master thesis.