Chapter 11

Summary and Conclusion

This chapter concludes the thesis, and is structured as follows. Section 11.1 presents answers to the research questions. This is followed by a summary of threats to validity in Section 11.2 and a discussion of generalization in Section 11.3. Finally, the chapter is ended with a summary of the key contributions of the thesis in Section 11.4 and discussion of further research in Section 11.5.

11.1 Answers to Research Questions

Challenges or limitations related to both large-scale and proximity to hardware are discussed in previous work, but without a holistic perspective which summarizes all challenges that must be taken into account (as described in Section 1.2). Answers to the research questions in this thesis presents solutions which cover the whole continuous integration and delivery pipeline. This will help organizations struggling with challenges related to scale and proximity to hardware to better implementations of continuous integration and continuous delivery.

11.1.1 Answers to Research Question RQ1

Research question RQ1 was phrased as: What are the challenges that must be taken into account when applying continuous integration to large-scale software-intensive embedded systems? The research question is addressed in the studies presented in Chapters 3, 4, 5 and 8. Results from the studies are presented in more detail in Sections 3.5, 4.9, 5.6 and 8.8 of this thesis.

Answers to research question RQ1:

Specific Factors Related to Software-intensive Embedded Systems

Based on experiences from two industry cases, the following factors are identified which must be taken into account when applying continuous integration to software-intensive embedded systems:

- If the developers run tests in a simulated environment, they cannot fully ensure that the same tests will pass for the integration build that runs on real hardware.
- Tightly coupled systems (causing long build- and test-time) imply additional challenges related to frequent deliveries and integration builds several times a day.
- A product with complex user scenarios and/or bespoke hardware (especially a large number of hardware configurations) implies that the rule that “all tests must pass for every build” must be replaced with other testing approaches.
• In a highly regulated environment, “fixing broken builds” must be balanced against other project objectives.
• At the initial phase of development of a new product (before the architectural runway is established) the sub-systems cannot be assembled in order to test the system functionally end-to-end and expose any integration problems.
• It is more difficult to achieve a common understanding of a product with a large number of technology fields or security aspects, which affects tests and reviews.

Correlation Between Scale and Continuity of Continuous Integration
Based on the study of developer behaviors in six industry cases over a two month period, it is found that organizational size clearly correlates with continuity: the larger the organization, the larger are each of the commits by its developers and the lower the number of builds per commit (the developers integrate less continuously). These findings were validated by interviews with ten senior engineers from five companies in different industry segments. The interviews also showed that product size, architecture and modularity are considered important factors in achieving continuous integration.

The collected data also points at several additional phenomena: The higher the percentage of non-developers in the organization, the more infrequently developers tend to commit. Another observation was that in some cases the average size of commits made by external consultants is twice as large as that of internal employees. This difference only manifests in cases where developers work on team or feature branches.

Importance of the Build System
From the comparison of the results from two studies in a large-scale industry project (before and after the introduction of a new build system) it is found that build system capacity is one of several factors which, if not considered, could result in that (according to the developers) the delivery process is time-consuming.

The three factors which affect continuous integration behaviors (the factors that make developers deliver less frequently) are:

• The delivery process is time-consuming
• It’s too complicated to deliver
• No evident value to deliver often to the mainline

Build system capacity should be considered as an important factor – but other factors should be seen as at least as important.

Continuous Integration Impediments
Based on interview results from developers in two companies, twelve factors are identified that affect how often developers deliver software to the mainline – the main factors that can enable more frequent integration of software. The twelve factors (grouped in four themes) are:
• Activity planning and execution:
  — Work breakdown
  — Teams and responsibilities
  — Activity sequencing
• System thinking:
  — Modular and loosely coupled architecture
  — Developers must think about the complete system
• Speed:
  — Tools and processes that are fast and simple
  — Availability of test environments
  — Test selection
  — Fast feedback from the integration pipeline
• Confidence through test activities:
  — Test before commit
  — Regression tests on the mainline
  — Reliability of test environments

11.1.2 Answers to Research Question RQ2

Research question RQ2 was phrased as: How should continuous integration and continuous delivery be defined for development of large-scale software systems? The research question is addressed in the studies presented in Chapters 6 and 7. Results from the studies are presented in more detail in Sections 6.5 and 7.7 of this thesis.

Answers to research question RQ2:

Continuous Integration Behaviors in Large-scale Projects
Based on the interviews with developers in two companies, it is found that the interviewees commit to a branch on a frequency spanning from multiple times a day to every second day, and commits directly to the mainline span from multiple times a day to once a week. Nine of the 20 interviewees prefer to commit to a branch, and the others prefer to commit directly to the mainline.

The analysis comes to the conclusion that continuous integration should not be simplified to discussing how the developers commit to the same mainline, but also includes integration of a subsystem on a branch or integration of binaries built from several different mainlines. In other words, one might be practicing continuous integration at one level, but not at another, and this may have different pros and cons depending on the context.

Definitions of Continuous Integration and Continuous Delivery
Based on a systematic mapping study it is found that there is indeed a high degree of confusion and contradictions (often internally in the same source) regarding the meanings attached to these terms. Definitions are presented (applicable to development of large-scale software systems) that closely corresponds to what is considered to be a mainstream interpretation:
• **Continuous integration** is a developer practice where developers integrate their work frequently, usually each person integrates at least daily, leading to multiple integrations per day.

• **Continuous delivery** is a development practice where every change is treated as a potential release candidate to be frequently and rapidly evaluated through one’s continuous delivery pipeline, and that one is always able to deploy and/or release the latest working version, but may decide not to, e.g. for business reasons.

This means that a large-scale organization may practice continuous delivery (implementing a continuous delivery pipeline) but may at same time fail to convince its developers to adopt continuous integration (integrating their software frequently).

In addition to the definitions of continuous integration and continuous delivery, definitions for continuous deployment and continuous release (both considered to be closely related to continuous integration and continuous delivery) are presented:

• **Continuous deployment** is an operations practice where release candidates evaluated in continuous delivery are frequently and rapidly placed in a production environment, the nature of which may differ depending on technological context.

• **Continuous release** is a business practice where release candidates evaluated in continuous delivery are frequently and rapidly made generally available to users/customers.

11.1.3 Answers to Research Question RQ3

Research question RQ3 was phrased as: *How can a model be defined that shows what companies should prioritize to improve their implementations of continuous integration and continuous delivery?* The research question is addressed in a study which is presented in Chapter 8. Results from the studies are presented in more detail in Section 8.8 of this thesis.

Answers to research question RQ3:

**Visualization of Continuous Integration Impediments**

The EMFIS model provides a framework that shows companies which areas they should prioritize in order to improve their implementations of continuous integration and continuous delivery.

The model is used to perform an assessment of the twelve factors, where the participants rate to which degree (on a Likert scale from 1 to 5) the description representing each factor mirrors the situation in their organization:

• **Work breakdown**: A way of working that supports work breakdown into small pieces that can be delivered to the software mainline.

• **Teams and responsibilities**: A project organization that supports both working with functional changes and at the same time takes responsibility for the architecture.

• **Activity sequencing**: Synchronization between the development teams in order to optimize the flow of activities when functions and systems are implemented.
• Modular and loosely coupled architecture: Modular architecture with small components, and loosely coupled architecture with as few dependencies as possible between the components.
• Developers must think about the complete system: Developers understand the functions and design of the whole system, and not just their own sub-system.
• Tools and processes that are fast and simple: Developers consider all tools and processes related to integration of the software fast and simple to use.
• Availability of test environments: Developers can get access to sufficient test resources for their test activities before committing to the mainline.
• Test selection: The suite of regression tests are constantly updated as new tests are added and old tests are removed (preferably automated selection in real time).
• Fast feedback from the integration pipeline: The developer gets fast feedback when software is committed to the mainline, signaling any deviations or problems.
• Test before commit: The test activities that are performed by the developers before committing software to the mainline are appropriate and conducive.
• Regression tests on the mainline: The regression tests on the mainline include a mix of test activities of varying scope and test coverage that protects mainline quality and stability.
• Reliability of test environments: The test environments have idempotent behavior and do fully represent the production environment.

In other words, the participants compare their current situation to an ideal situation without impediments, and thereby identify what the organization should focus on in order to enable more frequent integration of software. The EMFIS model has been validated in workshops and interviews in five companies, and was generally well received.

11.1.4 Answers to Research Question RQ4

Research question RQ4 was phrased as: How should the continuous integration and delivery pipeline be designed for large-scale software-intensive embedded systems? The research question is addressed in the studies presented in Chapters 9 and 10. Results from the studies are presented in more detail in Sections 9.9 and 10.7 of this thesis.

Answers to research question RQ4:

Designing the Continuous Integration and Delivery Pipeline

From interviews in four case study companies the following five stakeholder interests are identified:

• Check quality and correctness of software changes (e.g. software developers in a development team or a function team)
• Secure stability and integrity in the system during development (e.g. test managers or QA department)
• **Measure project progress** (e.g. project managers or line managers)
• **Verify compliance with requirements or user scenarios** (e.g. product owner or technical managers)
• **Optimize test activities and performance in the test facilities** (e.g. test specialists or test managers)

The Test Activity Stakeholders (TAS) model shows how the continuous integration and delivery pipeline can be designed to include test activities that support each of the stakeholder interests:

- **Early in the pipeline**: Unit/component tests followed by system tests to check the developers’ software changes. System tests of vital functions to secure stability and integrity in the system (monitored by e.g. a test manager or QA department).
- **Later in the pipeline**: A wider range of system tests, which the developer uses to check software changes. The same tests also secure stability (supporting e.g. a QA department) and can be used to measure project progress (for e.g. a project manager or a line manager).
- **Release pipeline**: When the organization is getting close to a release, test cases corresponding to each requirement should be executed on real hardware to support e.g. a technical manager to verify compliance. Progress in these test activities can also be followed by e.g. a project manager to measure project progress.

The fifth stakeholder interest (“Optimize test activities and performance in the test facilities”) optimizes the complete flow of test activities, and does not focus on particular test activities as the other stakeholder interests.

The TAS model shows how the test activities flow from primarily simulated test environments early in the pipeline to primarily real hardware late in the pipeline. Automated testing best supports test activities that are repeated, which to a large extent are test activities to secure stability and integrity in the system. Automated testing is complemented by manual testing, and especially exploratory testing which provides different types of feedback than automated testing.

The model can then serve as a starting point for companies when designing their continuous integration and delivery pipeline, and is valuable for both researchers and practitioners as it provides a systematic approach rather than making changes blindly and hoping for the best.

**Exploratory Testing in the Continuous Integration and Delivery Pipeline**

A test method for exploratory testing of large-scale systems is presented with the following characteristics:

- **Exploratory testing as an activity in the continuous integration and delivery pipeline**: Testing is conducted with an exploratory approach where the testers simultaneously learn about the system’s characteristics and behavior. Testing is done regularly on the latest system build, which has passed the test activity in the preceding step in the continuous integration and delivery pipeline.
• **Session-based testing in teams with experienced engineers representing different subsystems:** Testing is conducted in time-boxed sessions by teams of hand-picked experienced engineers, representing the different subsystems of the product. If the size or complexity of the system under test cannot be covered by a single team, the test scope can be split between several teams.

• **Scenario-based testing with an end-user representative as part of the test team:**
  Testing is conducted in scenarios, which represent how the product will be used by the end-user. An end-user representative is participating in both planning and test execution, securing that the scenarios are reflecting appropriate conditions.

The validation of the test method showed that the presented test method succeeds in incorporating exploratory testing in the continuous integration and delivery pipeline and is an efficient test method for large-scale and complex software products. Automated testing and exploratory testing could be complementing one another, each mitigating the weaknesses of the other by addressing unique concerns.

Whereas automated test activities in the pipeline are able to rapidly provide feedback to developers and to verify requirements, exploratory testing can provide more in-depth insights as it utilizes experienced engineers to identify complex integration problems. Exploratory testing should be used in a continuous integration and delivery pipeline, preferably to test new functions and systems in a large-scale system.

### 11.2 Threats to Validity

This section discusses threats to construct validity, threats to internal validity and threats to external validity in the studies, structured according to the four research questions.

#### 11.2.1 Threats to Validity Relevant in the Studies for Research Question RQ1

As described in Section 1.4.2, research question RQ1 (“What are the challenges that must be taken into account when applying continuous integration to large-scale software-intensive embedded systems?”) is addressed by four studies presented in this thesis. The studies are based on both quantitative and qualitative data (as described in Section 2.2.1)

In order to handle threats against **construct validity**, the interview guides used in the studies have been designed with open questions. The presentation of the studies also include information about the background for both the interviewees and the case study companies in order to provide as much information as possible about the context. As described in Section 2.1.2, observational bias has been mitigated with observer triangulation in all studies: researchers from different organizations have been involved in all studies and have participated in research design and analysis of collected data.

Of the 12 threats to **internal validity** listed by Cook, Campbell and Day (1979), Selection, Ambiguity about causal direction and Compensatory rivalry have been considered to be relevant in the studies. The interviewees in all studies were purposively sampled to represent a mix of roles, parts of an organization and/or industry segments.
In the same way the cases which provided quantitative data for the study analyzing correlation between scale and continuity of continuous integration (presented in Chapter 4) were purposively sampled: they were selected to create a population of accessible cases as similar as possible in all respects but their size.

The studies discuss correlation, but are very careful about making statements regarding causation. Statements that include cause and effect are collated from the interview results, and not introduced in the interpretation of the data. When performing interviews and comparing scores or performance, the threat of compensatory rivalry must always be considered. Therefore, the questions were deliberately designed to be as value neutral as possible, rather than judging the interviewee’s performance or skills.

Mortality was also considered to be relevant in the study with the objective to analyze the importance of the build system (presented in Chapter 5), as only part of the interview group used in the first part of the study was reused in the interviews in the second part of the study. As the study focused on the roles of the interviewees and not on the specific individuals, this was not considered to be an actual threat to validity.

Threats to external validity were considered in all studies. It is conceivable that the findings from the studies are only valid for the case study companies included in each study, or for companies in the same industry segments. Statements on generalizability of the results are therefore always supported by comparison with literature.

11.2.2 Threats to Validity Relevant in the Studies for Research Question RQ2

As described in Section 1.4.3, research question RQ2 (“How should continuous integration and continuous delivery be defined for development of large-scale software systems?”) is addressed by two studies presented in this thesis. The studies are based on qualitative data and a systematic mapping study of published literature (as described in Sections 2.2.1 and 2.2.2).

In order to handle threats to construct validity, the study which was to investigate continuous integration behaviors in large-scale projects (presented in Chapter 6) was basing the interview guide on the terminology which was considered to be most commonly used (see Section 6.2.2). Open questions were purposely designed to fit both developers who commit to a branch and those who commit directly to the mainline. Observational bias has been mitigated with observer triangulation.

Of the 12 threats to internal validity listed by Cook, Campbell and Day (1979), Selection, Ambiguity about causal direction and Compensatory rivalry have been considered to be relevant for the interviews in the study, and was handled in the same was as in the studies addressing RQ1 (described in Section 11.2.1).

The study was based on interviews from two case study companies. It is conceivable that the findings from this study are only valid for these companies. However, related work also describes that large-scale continuous integration is challenging, which shows that the problems identified in the study is evidently not isolated to the two case study companies (external validity).

The study with the objective to propose definitions of continuous integration and continuous delivery investigated published literature in a systematic mapping study. One may object that there is a substantially larger body of literature which is relevant.
for the study. While not arguing the point, based on the researchers own experience the publications included in the search were considered to be largely representative to general usage of the terms (continuous integration and continuous delivery) in the industry.

Another relevant concern is whether the algorithm used to classify publications is sound. The algorithm was validated in a process where the results were compared to manual searches, including the same search strings as used by the algorithm for specific themes.

### 11.2.3 Threats to Validity Relevant in the Studies for Research Question RQ3

As described in Section 1.4.4, research question RQ3 ("How can a model be defined that shows what companies should prioritize to improve their implementations of continuous integration and continuous delivery?") is addressed by one study presented in this thesis. The study was developing a new model (the EMFIS model) based on the studies which was conducted for RQ1, a systematic literature review, and qualitative data (as described in Sections 1.4.4, 2.2.2, 2.2.3 and 2.2.4).

Threats to **construct validity** was considered in the study: The EMFIS model is based on the twelve factors that have been identified the study with the objective to identity continuous integration impediments (presented in Chapter 8). That is, during an EMFIS assessment, the participants are rating the factors that (according to developers) are the factors that affect how often developers commit software to the mainline. Due to this, the EMFIS model is considered to have construct validity.

Of the 12 threats to **internal validity** listed by Cook, Campbell and Day (1979), Selection, Ambiguity about causal direction and Compensatory rivalry have been considered to be relevant for the interviews in the study, and was handled in the same was as in the studies addressing RQ1 (described in Section 11.2.1).

The validation of the EMFIS model was based on workshops and interviews with participants from five case study companies. It is conceivable that the findings from this study are only valid for these companies, or only for companies that operate similar industry segments. However, all the twelve factors used in the EMFIS model are discussed as continuous integration limitations or challenges by different sources in related work. This supports the generalizability of the results of this study (external validity) and that the EMFIS model is valid for all types of large-scale and complex software systems.

### 11.2.4 Threats to Validity Relevant in the Studies for Research Question RQ4

As described in Section 1.4.5, research question RQ4 ("How should the continuous integration and delivery pipeline be designed for large-scale software-intensive embedded systems?") is addressed by two studies presented in this thesis. The studies were developing a model (the Test Activity Stakeholders model) and a method (test method for exploratory testing of large-scale systems) based on a systematic literature reviews, quantitative data and qualitative data (as described in Sections 2.2.2, 2.2.3 and 2.2.4).
Threats to construct validity has been considered in both studies. The interview guides used in both studies were designed with open questions in the same way as in other studies included in this thesis. Observational bias has been mitigated with observer triangulation. The results from the two series of interviews in the study with the objective to describe how the continuous integration and delivery pipeline should be designed (presented in Chapter 9) were presented to the case study companies at a workshop and at separate meetings. This two-step validation supports construct validity of the Test Activity Stakeholders model.

In the study with the objective to incorporate exploratory testing in the continuous integration and delivery pipeline (presented in Chapter 10) the test rig was considered to be a scarce and valued resource. Therefore, the number of problem reports (defects found) per unit of rig time was measured in order to discuss the efficiency of the test method. The study does not claim to discuss efficiency on more general terms, such as comparing the importance of the problem reports from different types of test activities (which would be much harder to measure or quantify).

Of the 12 threats to internal validity listed by Cook, Campbell and Day (1979), Selection, Ambiguity about causal direction and Compensatory rivalry have been considered to be relevant for the interviews in the studies, and was handled in the same way as in the studies addressing RQ1 (described in Section 11.2.1).

The Test Activity Stakeholders model are based on interview results from four case study companies. It is conceivable that the findings from this study are only valid for these companies, or only for companies that operate in the same industry segments. However, the characteristics of the Test Activity Stakeholders (TAS) model are supported by publications that were covered by the systematic literature review, which supports the generalizability of the results of this study. Furthermore, the TAS model was validated with a series of interviews with twelve individuals from three case study companies, which also supports the generalizability of the model (external validity).

The validation of the test method for exploratory testing of large-scale systems was based on interviews and quantitative data from a single company. It is conceivable that the findings from this study are only valid for this company, for companies that operate in the same industry segment, or for similar products in different types of industry segments (e.g. other types of vehicles). The characteristics of the test method are in different ways described in related work, which supports the generalizability of the results of this study (external validity).

11.3 Generalization

The results presented in the articles included in this thesis are based on quantitative data (e.g. number of developer commits or registered problem reports) and qualitative data (results from interviews and workshops) coming from one or several case study companies. It is conceivable that the findings from the studies are only valid for the case study companies in the study, or only for companies that operate in the same industry segments.

However, the conclusions in the studies were generally drawn from several case study companies and are often based on a large number of participants, which supports
the generalizability of the results reported in this thesis (as described in each of the articles). In the same way, the quantitative data for the study analyzing correlation between scale and continuity of continuous integration (presented in Chapter 4) was collected from six industry cases (altogether close to 2,000 engineers), purposively sampled to create a population of accessible cases as similar as possible in all respects but their size.

The results from the nine research studies presented in this thesis are based on data from eight case study companies. Table 23 summarizes the number of case study companies and number of participants for each of the research studies. The study with the objective to identify specific factors related to software-intensive embedded systems (presented in Chapter 3) was based on the researchers own experiences. The study that was to investigate continuous integration behaviors in large-scale projects (presented in Chapter 6) was based on a systematic mapping study of published literature, and did therefore not include any case study companies.

<table>
<thead>
<tr>
<th>Study Description</th>
<th>Number of case study companies</th>
<th>Participants in the study (interviews and workshops)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify specific factors related to software-intensive embedded systems (Chapter 3)</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>Analyze correlation between scale and continuity of continuous integration (Chapter 4)</td>
<td>1 + 5</td>
<td>10</td>
</tr>
<tr>
<td>Analyze the importance of the build system (Chapter 5)</td>
<td>1</td>
<td>15 + 10</td>
</tr>
<tr>
<td>Identify continuous integration impediments (Chapter 8)</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Investigate continuous integration behaviors in large-scale projects (Chapter 6)</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>Propose definitions of continuous integration and continuous delivery (Chapter 7)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Visualize continuous integration impediments (Chapter 8)</td>
<td>5</td>
<td>46</td>
</tr>
<tr>
<td>Describe how the continuous integration and delivery pipeline should be designed (Chapter 9)</td>
<td>4 + 3</td>
<td>7 + 25 + 12</td>
</tr>
<tr>
<td>Incorporate exploratory testing in the continuous integration and delivery pipeline (Chapter 10)</td>
<td>1</td>
<td>28</td>
</tr>
</tbody>
</table>

Table 23: The number of case study companies and number of participants in the studies included in the thesis.
Table 6 in Section 2.3 presents the applicability of each study (depending of the research question). The table shows that the results from the studies reported in Chapters 4-6 are applicable for all types of large-scale software system, i.e. not only large-scale software-intensive embedded systems. Chapter 7 has, in the same way, general applicability (all types of software systems). The results reported in Chapters 8-10 are primarily applicable to large-scale software-intensive embedded systems. However, in both Chapter 8 and Chapter 9 the areas are isolated that are especially related to the characteristics of software-intensive embedded systems, and with that taken into account the results can be generalized to all types of development of large-scale software system.

11.4 Key Contributions

Continuous integration and continuous delivery were introduced by XP and other agile methodologies to mitigate problems with a long and unpredictable integration process (as described in Section 1.1). As shown in Section 1.2, publications in related work discuss a wide range of “difficulties” or “challenges” related to continuous integration combined with large-scale systems or embedded systems. Sources in published literature propose ideas or solutions that could improve implementations of continuous integration and delivery, but are focusing on one particular topic (e.g. prioritization of unit tests). A more holistic approach is needed which covers all relevant aspects, and covers the whole continuous integration and delivery pipeline.

The key contributions of the research presented in this thesis are two models and a method, which will help organizations struggling with challenges related to scale and proximity to hardware to better implementations of continuous integration and continuous delivery. These contributions are valuable for both researchers and practitioners, as they provide a systematic approach rather than making changes blindly and hoping for the best. The key contributions of this thesis are:

- **The EMFIS model**, which allows companies to explicate a representation of the organization’s current situation regarding twelve continuous integration impediments, and visualizes what the organization must focus on in order to enable more frequent integration of software. The model has been validated in workshops and interviews, which in total included 46 individuals from five case study companies. The EMFIS model was generally well received during the validation, and was appreciated for its simplicity and its ability to show differences in rating between developers and enablers.

- **The TAS model**, which shows how the continuous integration and delivery pipeline can be designed to include test activities that support four stakeholder interests: “Check changes”, “Secure stability”, “Measure progress”, and “Verify compliance”. The model is developed to show how each of the stakeholder interests are best supported by unit/component tests or system tests, by automated testing or manual testing and by tests executed in simulated environments or on real hardware. The
model was well received during the validation by all of the four case study companies.

- A test method for exploratory testing of large-scale systems, which incorporates exploratory testing as a test activity in the continuous integration and delivery pipeline. The test method is designed to complement automated testing in the continuous integration and delivery pipeline, and to provide different feedback and insights than the results from an automated test case. Results from the validation show that each of the characteristics of the test method were considered valuable by the interviewed 18 engineers and 7 flight test pilots, and that they consider the test method to be an efficient way of testing large-scale and complex software products.

11.5 Further Research

Suggestions for further work are included as a part of all articles in this thesis. This section summarizes the most significant areas of further research, which could build on the work that is presented in this thesis.

Chapter 8 presents a list of twelve continuous integration impediments in large-scale industry projects – the factors which affect how often developers commit software to the mainline. This thesis focuses on solutions for how to handle the impediments related to the selection of test activities for the continuous integration and delivery pipeline. A vast area of further research is to continue the work with solutions for how to handle the continuous integration impediments, e.g. proposing new approaches for better architecture, faster tools and processes, reliable test environments et cetera.

The initial validation of the Test Activity Stakeholders model (presented in Chapter 9) showed very promising results. Further research could continue to investigate which types of test activities that best support the stakeholder interests, in order to provide more detailed recommendations for the design of the continuous integration and delivery pipeline. Further research could also expand the Test Activity Stakeholders model to other ways of working than using branches for development of features or subsystems. This could be combined with further validation of the model, encouraging companies in other industry segments to use the model to design their continuous integration and delivery pipeline.

As the validation of the test method which is presented in Chapter 10 is based on a single case study, this calls for validation from other case study companies using the same test method. Further research could also further investigate how exploratory testing could be used in the continuous integration and delivery pipeline. A suggested approach is to combine this with the perspectives from the TAS model (presented in Chapter 9) and investigate how exploratory testing could be used for different purposes in different phases of the continuous integration and delivery pipeline.