Chapter 12. Conclusion

This chapter concludes the thesis and is structured as follows. First, each research question (see Table 1) is addressed and answered. This is then followed by a discussion of threats to validity. Following this, hypotheses as to the applicability of this research to other domains are presented. Finally, the chapter is ended with the key contributions of the thesis and discussion of further research.

12.1 Research Questions

This section consists of two parts. The first part lists the research questions addressed in this thesis, and answers them one by one (see Table 1 for a mapping of research questions to chapters). The second part discusses the mapping of these research questions to the problem statement and problem domain breakdown (see Section 1.2).

12.1.1 Research Question Answers

RQ 1.1: Does continuous integration support the agile testing practices of automated customer acceptance tests and writing unit tests in conjunction with production code?

Answer: Through interviews with industry practitioners, it is shown that there exists a relationship between continuous integration and the agile testing methods of automated customer acceptance tests and writing unit tests in conjunction with new production code. That being said, however, there are unanswered questions raised as to whether continuous integration supports the unit test practice, or if it’s the other way around, or if they support one another. Furthermore, it is difficult to isolate the effect of continuous integration on the practice of automated customer acceptance testing from contextual factors, such as organizational structure, culture and customer availability.

RQ 1.2: Does continuous integration contribute to improved communication both within and between teams?

Answer: Interviews with industry practitioners show that continuous integration is generally perceived as having a positive effect on communication – not just within the team, but in larger projects also between teams. It shall be noted, however, that in the collected data set, interviewee answers differ widely, with interviewees providing diverging accounts of how product build and quality status was communicated in their respective cases. Therefore further investigation into how differences in continuous integration implementations affect these potential benefits is warranted.

RQ 1.3: Does continuous integration contribute to increased developer productivity as an effect of facilitating parallel development in the same source context and reduced compiling and testing locally before checking in?
Answer: Through interviews with industry practitioners, strong support for the hypothesis that continuous integration improves productivity is found, but only for one of the stipulated reasons: while there is consensus that it facilitates parallel development, there is very weak support for any time saving benefits. The latter is another area of conflicting views, however: some interviewees perceive this effect very clearly, whereas others do not see it at all.

This contention between the views of industry practitioners in different cases is highly interesting, particularly in the context of research question RQ 1.8, which considers whether there is any disparity or contention in descriptions of actual practice. Finally, the interview data also supports the notion that productivity increases due to more effective troubleshooting and easier re-basing and merging, although to a lesser degree than the parallel development effect.

RQ 1.4: Does continuous integration improve project predictability as an effect of finding problems earlier?

Answer: Interviews with industry practitioners show that continuous integration is perceived as improving project predictability by findings problems earlier. Additionally, predictability is found to be improved by the fact that continuous integration enables early non-functional system testing outside of the project’s critical path.

RQ 1.5: What is the impact of using agile principles and practices in large-scale, industrial software development?

Answer: Through a study investigating the impact of using agile principles and practices in two large-scale software development projects – one largely plan-based project and one entirely agile project – five effects of these principles and practices were found:

• Balanced use of internal software documentation.
• Facilitated knowledge sharing.
• Increased visibility of the status of other teams the entire project.
• Effective coordination, with less overhead.
• Possibly increased productivity.

It was further found that internal software documentation is important also in agile software development projects, and cannot fully be replaced with face-to-face communication, that it is possible to make a partial implementation of agile principles and practices and still receive a positive impact, and that it is feasible to implement agile principles and practices in large-scale software development projects.

Agile principles and practices were not found to increase negative pressure and stress within the studied projects, although some agile practices were perceived as stressful.

RQ 1.6: Which factors must be taken into account when applying continuous integration to software-intensive embedded systems?

Answer: Based on experiences from two industry cases, the following impediments to continuous integration in the context of software-intensive embedded systems are identified:

• 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.

RQ 1.7: What is the correlation between size of an area of direct change impact and the continuity of continuous integration in industry practice, and how does it affect developer behavior?

Answer: Through 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 the changes committed by its developers and the lower the number of builds per commit. These findings were corroborated by interviews with ten senior engineers from five companies in separate segments of the industry – interviews which also show that product size, architecture and modularity are considered important factors in achieving continuous integration.

Apart from these findings, the collected data hints at several additional phenomena worthy of further research:

• The higher the percentage of non-developers in the organization, the more infrequently developers tend to commit.

• 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, however.

RQ 1.8: Is there disparity or contention evident in the descriptions of various aspects of the software development practice of continuous integration found in literature?

Answer: Through a systematic literature review, sixteen points of divergence in continuous integration practice are discovered, ranging from build duration and pre-integration procedure to test scope and fault handling. This shows that continuous integration, as implemented in industry, is not a single, homogeneous and well understood practice, but arguably more of an umbrella term for related and similar methods and techniques.

RQ 2.1: To what extent can differences in perceived benefits of continuous integration be explained by differences in practice, as documented by a descriptive model?
Answer: To answer this question, five continuous integration cases in industry were studied and modeled using the ASIF modeling technique. These models were then analyzed and compared to the reported experiences of interviewed engineers in the respective cases. Through this analysis, differences in experiences were connected to differences in practice, resulting in the formulation of six guidelines for the design of continuous integration systems:

- **Comprehensive Activities:** Construct the automated activities such that their scope affords a sufficient level of confidence in the artifacts processed by them.

- **Effective Communication:** Ensure that the integration flow itself and its output are easily accessible and understandable by all its stakeholders.

- **Immediacy:** Make the integration flow easily and quickly accessible for the project members.

- **Appropriate Dimensioning:** Adjust the capacity of the automated integration flow according to the traffic it must handle.

- **Accuracy:** For all activities, know well what the input is, its history and consequently what level of confidence may be assigned to it.

- **Lucidity:** Keep the flow of changes through the integration system clear and unambiguous.

RQ 2.2: How may continuous integration modeling be applied to benefit industry practitioners in their day-to-day work?

Answer: Through the study of four industry cases it is found that modeling of continuous integration systems can be beneficial to industry professionals by improving understanding and communication of complicated continuous integration system, as well as for identifying and planning improvements to those system and, to a lesser extent, as direct input and support to the subsequent technical work to implement those improvements.

RQ 2.3: How can the continuous integration specific modeling techniques of ASIF and CIViT be effectively applied in tandem to complement each other?

Answer: Through application of the ASIF and CIViT modeling techniques to four industry cases it is found that the two continuous integration modeling techniques can favorably be used to complement each other by leveraging their focus on low and high levels of abstraction, respectively. However, it is also found that there is room for improvement both in their alignment – allowing models of one technique to serve as input to the other more effectively – and in the design of each individual technique itself, as well as in improved tool support.

RQ 2.4: In what way can the paradigm of architecture frameworks favorably be applied to facilitate the design and description of continuous integration and delivery systems?

Answer: In answering this question, Cinders, an architecture framework specifically designed to address the needs of describing continuous integration and delivery pipelines while at the same time combining and improving upon the modeling techniques of ASIF and CIViT, is designed and presented. This work represents a significant step in the iterative design process which began with the proposal of a modeling technique addressing established points of divergence in continuous integration practice (see Chapter 6) and then evolved through repeated industry applications (see Chapters 7 and 8).
It is shown that this architecture framework addresses the requirements collected from industry application of the modeling techniques it is designed to replace, such as allowing users to change level of abstraction, offering multiple views of the same description and representing both manual and automated activities and that it constitutes an improvement over the two separate modeling techniques. Consequently, it is shown that the paradigm of architecture frameworks can favorably be applied to facilitate the design and description of continuous integration and delivery systems.

**RQ 3.1:** How can traceability needs of large scale software development in industry be effectively addressed in a continuous integration and delivery context?

**Answer:** Through study of three very-large-scale, geographically dispersed software development projects and interviews with engineers in those projects, it is shown that traceability is a serious concern to software professionals of multiple roles, not least in the context of continuous integration and delivery, where it is in fact regarded as a prerequisite for successful large scale implementation of these practices. Particularly, based on these findings, it is concluded that the lack of adequate traceability solutions poses a threat to several important software engineering abilities:

- Troubleshooting faults discovered in continuous integration and delivery.
- Discovering problems such as congestion or resource scarcity in the continuous integration delivery system.
- Determining the content of system revisions.
- Monitoring the progress of features, requirement implementations, bug fixes et cetera.
- Providing developers with relevant feedback with regards to the source code changes they commit.

Addressing these concerns, the industry developed open source continuous integration and delivery framework Eiffel is presented and validated, using multiple methods. It is shown to address the identified lacks and facilitate the above software engineering abilities.

**RQ 3.2:** How may traceability data automatically generated in real time throughout the continuous integration and delivery pipeline be used to improve software testing practices?

**Answer:** The opportunities of comprehensive real time traceability data generated in continuous integration and delivery pipelines are discussed, and it is hypothesized that the data provided by the Eiffel framework in particular can support automated real time dynamic selection of the most cost effective test cases to execute at any given time.

**12.1.2 Research Question Mapping to Problem Domain**

Revisiting the problem domain breakdown (see Figure 1), one finds that the answers to the research questions also shed light on and can be mapped to one or more of the leaf nodes of that graph, as shown in Figure 50, thereby contributing to answers to the root question derived from the original problem statement (see Section 1.2). It would be preposterous as well as presumptuous to suggest that these answers are exhaustive. As with any field of research, our understanding of the phenomenon of continuous integration and delivery can be likened to the expanding ripples on the surface of a pond: the more we learn, the more our horizon expands and the more we realize that we in fact do not yet know, allowing us to pose further questions.
In this spirit, this thesis marks one small movement of the ripples in that pond, while recognizing that the question of how continuous integration and delivery practices can be effectively, efficiently and reliably adopted is likely to occupy researchers and to yield new answers for many years. A small selection of topics for further research are presented in Section 12.5.

12.2 Threats to Validity

In this section the threats to validity of the results are discussed. Since each chapter of the thesis is independent in terms of methodology and results, the threats are discussed for each chapter separately.
12.2.1 Chapter 2

A threat to the external validity of Chapter 2 is that only cases within a single company were studied. This is partly mitigated, however, by the fact that four independent cases – developing separate products within a very large company – were investigated. The cases were also selected to represent cases with long and short experience of continuous integration, respectively, as well as lesser and greater focus on integration of binary components, respectively. The findings are further supported by subsequent chapters, which document divergence in interpretation, implementation and experience of continuous integration and related practices, leaving the author confident that the results are not restricted to the one company.

With regards to internal validity, selection is a threat because the interviewees were not randomly sampled: managers in each respective case were asked to provide available interviewees, without interference from the authors. Had the set of interviewees been smaller, this would have been problematic, but with a minimum of four interviewees per case, representing different engineering roles, this threat is reduced.

12.2.2 Chapter 3

The results of Chapter 3 are largely based on a survey conducted in the two studied cases. The validity of such a survey is always subject to its response rate, but even though participation in the survey was voluntary it yielded an acceptable response rate, particularly in the agile project. The respondents also represent different functional roles and levels of experience in a satisfactory manner.

Another threat is that the survey questions do not capture the real perceptions of the respondents. To mitigate this risk, people with domain knowledge – including the project management of one of the cases – were asked to review the questions and the methodology as well as to pilot the survey beforehand.

Finally, the study results were presented to both the studied projects for validation, where they were given the opportunity to present their views on both the collected data and the subsequent analysis.

12.2.3 Chapter 4

Chapter 4 presents experiences of impediments to effective implementation of continuous integration, gathered from two study companies. This type of experience report is always subject to threats to both internal and external validity: it is possible that the experiences reported are misconstrued or that they are limited to the cases reported form, particularly when based on a single case and/or the experiences of a single individual. In this case, however, these threats are largely mitigated by the fact that the presented results represent the synthesis of experiences from two highly experienced engineers with many years of collective experience from continuous integration of multiple products in their respective companies, which operate in independent industry segments.

12.2.4 Chapter 5

There are several threats to internal validity in Chapter 5. Selection is a threat, because both the studied cases and the interviewees were purposively sampled, rather than randomly selected. Considering the rationale of these samplings and the fact that the resulting data sets are mutually
supportive, however, the author considers this threat to be mitigated. Ambiguity about causal direction is also a threat, as the causality of size and continuity is not established. For this reason, the author is very careful about making any statements about causality, preferring to instead discuss correlation. Finally, compensatory rivalry is always relevant when performing interviews and comparing scores or performance. To mitigate this threat, the questions (see Table 16) were deliberately chosen to be value neutral by assessing correlation, rather than judging capability.

With regards to external validity, the quantitative data was gathered from six cases within the same company. To mitigate this, the findings were triangulated using interviews with ten engineers in five separate companies.

12.2.5 Chapter 6

Chapter 6 clusters discrete statements extracted from a systematic literature review into 22 concrete aspects of continuous integration practice, and finds divergence and/or contention in all but six of them. Based on this finding, a descriptive modeling technique designed to address all these points of divergence is presented.

There is a clear threat to external validity in that this modeling technique is only applied in a single illustrative case study: it is entirely conceivable that such a modeling technique is not applicable outside of that particular case. There are two mitigating factors, however. First, the model designed is derived not from a single case but from 46 descriptions of the practice in published literature. Second, the modeling technique is evolved and further validated in subsequent studies (see Chapters 7, 8 and 9).

12.2.6 Chapters 7 and 8

The methodology and results in these chapters are similar, and consequently discussed together.

Researcher bias is a concern, as one or both of the authors, respectively, have previously been involved in designing the modeling techniques studied in these chapters. However, the techniques were not chosen because of this involvement, but rather despite it, as there is a demonstrable lack of alternatives (see Section 8.5). In addition, a certain degree of protection is offered by the research design, particularly in the case of Chapter 8, which is based on the direct and presumably unbiased feedback from software professionals in multiple companies and settings as the primary data source.

Similar to previous studies (see Section 12.2.1) the participants in the workshops where the studied techniques were applied to industry cases were not randomly selected, but rather appointed by the author's contact person in each respective case. This method was chosen since random sampling was not considered conducive to the study – not everybody involved in a development project will necessarily possess the knowledge to contribute to an accurate model of the continuous integration and delivery pipeline. Furthermore, the reason for letting the contact persons appoint the participants was to protect from research bias, but also for the pragmatic reason that the researchers did not possess the required insight into the competence of potential participants.

While it is true that this method caused differences in participant composition from case to case, which may have affected the outcomes of the workshops and interviews (see Section 8.4), the author does not consider this effect significant enough to pose a threat to the conclusions.
12.2.7 Chapter 9

In this chapter the architecture framework Cinders is presented along with the work leading up to its design. Data triangulation was relied upon to ensure construct validity: validating data was collected from multiple separate and independent sources, using different methods.

With regards to internal validity, relevant threats include the selection of participants in the workshops and interview sessions used to evaluate the architecture framework. Similarly to previous studies, the selection of workshop participants was performed by the company representative for each studied case, with the only requirement to include multiple roles, while the interviewed specialists were selected based on availability.

External validity is addressed by the fact that the architecture framework has been successfully applied to three independent industrial contexts in very different business segments (military aerospace, food packaging and visual surveillance and security systems, respectively). This was then followed up by interviews with senior engineers in the telecommunications industry. As to the question of whether the results of the study are generalizable, there are two separate but related questions to consider. One is the extent to which the systematic approach to continuous integration and delivery system descriptions represented by the Cinders framework and its predecessors is applicable to the software industry in general. The author would argue that this has been clearly shown: apart from the successful applications described in Chapter 9, the ASIF and CIViT techniques, with which Cinders shares many similarities, have in Chapters 6, 7 and 8 been successfully applied to a wide array of industry cases.

The second question is whether Cinders constitutes an improvement over its predecessors in the general case, and not just in the studied cases. Here the number of studied cases and consequently the body of evidence is smaller. That being said, great care has been taken to choose cases in independent companies in separate business segments to maximize diversity.

12.2.8 Chapter 10

In Chapter 10 the traceability challenges of continuous integration and delivery are discussed, whereupon the Eiffel framework is presented and evaluated through the study of three industry cases in two companies. The threat to external validity is partly mitigated by this, but also by the fact that comparable solutions have not been found in literature.

Threats to internal validity include selection and compensatory rivalry. Selection, because interviewees were not randomly selected. Instead, purposive sampling was used – in case A in order to cover as many roles and perspectives as possible, and in cases B and C to find perspectives correlating to those having the most articulate functional needs in case A. Compensatory rivalry, because it is conceivable that interviewees feel compelled to put their project in the best light. To protect against this threat, care was taken to ensure that there were no incentives in the study to provide positive answers and, in the author's view, most interviewees tended more towards self-criticism than self-praise.

Similarly to Chapter 9, researcher bias is a concern due to the involvement of two of the researchers in the development of the studied framework. To defend against this, apart from participant observation, three additional methods of validation were employed to maximize triangulation (see Section 10.5).
12.2.9 Chapter 11

Chapter 11 does not present any research results on its own, but rather looks forward and identifies opportunities for improved testing practices based on other work, particularly that presented in Chapter 10.

12.3 Generalization

The focus of this thesis is continuous integration and delivery in large scale contexts. That does not necessarily mean that its results are inapplicable to smaller contexts. To exemplify, systematic design and description of one’s continuous integration and delivery pipeline is arguably valuable even if that pipeline is a small one, although it is reasonable to expect that value to increase the larger the system is, and the greater the number of people involved. Conversely, at some point the effort involved in creating that architecture description presumably exceeds its benefits. The same applies to the Eiffel framework: in a project of one or two people, it is far from certain that the effort of adopting the framework will pay off.

On a more general note, one can argue that the need for and/or difficulties involved in continuous integration and delivery scale in a similar fashion. In a one-person project, the entire notion of continuous integration becomes contrived. In a five-person project, on the other hand, there is a clear rationale for continuously integrating, but its adoption is arguably fairly straight-forward. In a thousand-person context, on the other hand, the need for and challenges involved in systematic approaches to rapidly and frequently integrating everybody’s changes are equally great, and hence the gains from any improvements that can be made.

It is also the case that much of the research presented in this thesis has been conducted in or focused on the domain of embedded systems, particularly within the realm of the telecommunications industry, the automotive industry and the defense industry; indeed, particular impediments to continuous integration have been identified within this domain. That is not to say that the research results of the thesis at large are limited to embedded systems, however.

Continuous practices are prevalent in the software industry, and not limited to any specific segment. Similarly, the majority of the research in this thesis is aimed at these practices in general and presumed to be applicable to a multitude of contexts. Consequently, while it is not shown that every research result is applicable to every segment of the industry, there is no reason to believe that they are not – at least to large parts of it – unless otherwise stated. On the contrary, application of the modeling techniques and frameworks presented in this thesis to new cases has repeatedly shown their applicability to yet more contexts: there is yet to be an industry case where inapplicability is demonstrated for these methods and frameworks.

That being said, it shall be noted that different industry segments operate under very different conditions in the space of continuous practices. This is particularly the case with regards to continuous deployment, which is largely outside the scope of this thesis, and to a lesser extent continuous delivery. Such differences include:

- **Deployment method:** Does one deploy a web service, release user-installed software, or deliver embedded systems operated by the customer?

- **Customers:** Is the customer a single, large, powerful and opinionated organization, or tens of millions of individuals?
• **Regulatory space**: Is the software subject to regulations requiring evidence of compliance to trade laws, safety regulations et cetera, or is it unburdened by such constrains?

Such considerations have significant impact on the way one must shape one’s continuous integration and delivery pipeline: which tests must be executed, what level of traceability is required, and how far does it reach into the production environment? Consequently, they will also impact the way and the extent to which the research results of this thesis can be applied to the individual case.

### 12.4 Key Contributions

This thesis demonstrates that the adoption of continuous integration and delivery in large scale contexts is anything but trivial, and presents two frameworks to mitigate these difficulties: Cinders and Eiffel, respectively.

The key contributions of the research presented in this thesis can be split into four categories: overview and classification of the many points of divergence in continuous integration practice, analysis and discussion of impediments to effective implementation of continuous integration, the architectural framework Cinders, and the open source continuous integration and delivery framework Eiffel.

The specific contributions of this thesis with regards to the overview and classification of divergence in continuous integration practice are:

- Positive effects of agile practices, including continuous integration, in large scale software engineering are found to include the facilitation of knowledge sharing, increased visibility of the status of other teams and the entire project, effective coordination and possibly increased productivity.
- It is concluded that while there exists not one, but several potential benefits to continuous integration, and the practice is generally perceived as having beneficial effects on several aspects of software engineering, industry practitioners experience very different benefits from their respective continuous integration implementations.
- Sixteen concrete points of divergence in continuous integration practice are documented, based on a systematic literature review, showing that continuous integration is not a homogeneous and universally understood practice, but that a high degree of diversity in interpretation and implementation exists.

The specific contributions of this thesis with regards to analysis and discussion of impediments to effective implementation of continuous integration are:

- The development of embedded software poses particular challenges to continuous integration practice. Seven impediments to successful continuous integration implementation are presented and mapped to cornerstones of the practice proposed in literature.
- Difficulties in achieving continuous integration are found to correlate with organizational size: the larger the organization developing the software is, the less continuously that software tends to be integrated.
Evidence indicating that not only the size of the organization affects the ability to integrate continuously, but also its composition: in studied cases, the larger the percentage of non-developers, the less frequently developers commit changes.

It is found that software architecture and modularity play a significant role in enabling large scale continuous integration.

Based on multiple case studies, guidelines for continuous integration and delivery pipeline design are presented.

The specific contributions of this thesis with regards to the architectural framework Cinders are:

- Based on the documented points of divergence in continuous integration practice, the Automated Software Integration Flows (ASIF) modeling technique, designed to address these points of divergence, is presented.
- The ASIF modeling technique is validated and evolved through application to a total of nine industry cases.
- The ASIF modeling technique is combined with the Continuous Integration Visualization Technique (CIViT), which it is shown to complement. This type of modeling of continuous integration and delivery pipelines is shown to serve multiple purposes, aiding communication and alignment between multiple stakeholders as well as supporting technical work in designing and planning changes to those pipelines.
- The ASIF and CIViT techniques are evolved into a coherent architectural framework, Cinders, which is shown to be an improvement over the two independent techniques.

The specific contributions of this thesis with regards to the continuous integration and delivery framework Eiffel are:

- Eiffel, an industry developed open source framework for continuous integration, is presented and evaluated. Its ability to satisfy multiple types of traceability requirements for multiple stakeholder roles is documented, not only in the context of the unique challenges posed by continuous software engineering practices, but also generic challenges such as scale, technology diversity, geographic dispersion and process diversity.
- Not only is Eiffel shown to play a crucial part in satisfying traceability requirements in a continuous integration and delivery context, but new opportunities for increased software engineering efficacy afforded by that traceability are discussed, particularly with regards to dynamic real time selection of automated test cases in the continuous integration and delivery pipeline.

12.5 Further Research

While interest from the researcher community in continuous integration and delivery has increased substantially over the last few years, there are many areas left unexplored and questions left unanswered.
12.5.1 Clear Definitions

As demonstrated in this thesis and discussed in Section 1.2, there is a considerable degree of divergence in terms of interpretation and implementation of continuous integration. Unfortunately, this divergence is not limited to continuous integration, but arguably even worse in the case of continuous delivery. Neither is it limited to the community of practitioners, but conspicuous in the researcher community as well.

As previously noted, [Rodriguez 2016] finds that while some authors separate continuous delivery and deployment, others “use the terms continuous deployment and continuous delivery inter-changeably”. Moreover, continuous deployment is often defined in terms of and delivery or release, and vice versa, e.g. stating that “continuous delivery is a set of practices and principles to release software faster and more frequently” [Krusche 2014], that “The concept of continuous deployment, i.e. the ability to deliver software functionality frequently to customers [...]” [Olsson 2012] or “[...] continuous delivery – that is, to continuously deploy the environment in a test environment that is reasonably similar to the actual production environment” [Kalantar 2014] to mention a few.

This confusion regarding terminology renders the community great harm and impedes progress for practitioners and researchers alike: unless one is clear on what one describes, studies, presents recommendations for or otherwise reports on, it is very difficult for the community to benefit from and build upon those findings. Consequently, while this thesis attempts to be clear on the definitions used, this is not an issue easily addressed by a single thesis or study, but a crucial question for the community as a whole to resolve in order to achieve both conceptual and terminological clarity.

12.5.2 Cinders Tool Support

As found in the evaluation of the Cinders architectural framework, its usefulness would be greatly enhanced by improved tool support. There are two types of tooling needed: tools for collecting quantitative data, and tools for designing and manipulating the architecture descriptions.

The Eiffel framework is a promising candidate, primarily for the first need, but also for the second. The data contained in the events reported in real time by Eiffel is a very close match to that included in a Cinders architectural description, and so further investigation of the compatibility of Eiffel and Cinders in this regard would be very valuable. It is also possible, however, that Eiffel data could be used not only to populate the quantitative data of such a description, but that the activities themselves and their relationships in e.g. the Causality Viewpoint (see Section 9.5.2.1) could be derived from automated interpretation of Eiffel data.

12.5.3 Dynamic Selection of Test Cases

As discussed in Chapter 11, as test scopes grow the need to critically select which test cases to execute at any given time, and for any given change, grows. Several methods of selection of test cases have been proposed in literature, but the traceability data provided by e.g. the Eiffel framework offers new possibilities: by literally connecting the dots not only to many different types of engineering artifacts – agnostic as to programming language and development environments – but also backwards in time, there are great opportunities for finding improved algorithms for automatically determining which test cases are most cost effective to execute at any given time.
12.5.4 Machine Learning Applied to Continuous Integration and Delivery Pipelines

As has been shown in this thesis, it is possible to generate substantial amounts of structured real time data from continuous integration and delivery pipelines: what has changed, where was it changed, who changed it, why was it changed, where was it integrated, in which environment was the product built, which tests were executed, in which configurations and environments were they executed, what were their results, et cetera. It is impossible for a human mind to keep pace with this stream of data, but it is entirely reasonable to assume that a machine learning algorithm could be trained to detect patterns and make predictions.

To exemplify, such an algorithm could be used to recommend which test cases to select for execution (as discussed in Section 12.5.3), predict the in-production performance and likeliness of faults of product revisions, or estimate the likelihood that a certain composition of components is likely to integrate successfully.

This is an extremely exciting area of research, particularly considering the recent rapid advances in machine learning and artificial intelligence technology.

12.5.5 Improved Understanding of Impediments to Continuity

While this thesis has shown that there are impediments to achieving continuous integration, particularly in large organizations, it is still not clear why it is so. An important area of further research is therefore to investigate what precisely is causing developers to not integrate as frequently as one would expect them to in an ostensibly continuously integrating project. While this is an area that invites to speculation and conjecture – not least based on personal experiences and anecdotal evidence – it is necessary to investigate the causes of the phenomenon as diligently as possible: at the most fundamental level, what is keeping the individual developers from committing their changes to the mainline as frequently as they presumably want?