Licentiate Thesis Proposal

Controlling consistency for continuous model-based development

Robbert Jongeling
School of Innovation, Design and Engineering (IDT)
Mälardalen University
robbert.jongeling@mdh.se
December 2019

Main Supervisor: **Jan Carlson**
Co-supervisors: **Antonio Cicchetti**
Federico Ciccozzi

Abstract

For the development of complex embedded software systems, two prominent paradigms have gained popularity: Model-Based Development (MBD) and continuous integration (CI). MBD holds the promise of improving the productivity of software development through abstraction, by focusing on the problem domain and capturing it in models, rather than just focusing on the solution domain and capturing that in implementation code. MBD is sometimes seen as conflicting with CI, due to its apparent nature of longer development phases as opposed to the short development cycles as proposed in the agile methodology. In this thesis, we explore what would be needed to nevertheless combine these two practices successfully and get the best from both worlds. In particular, the overall goal is to investigate impediments to introducing CI in MBD, i.e., to allow development to profit both from the benefits of abstraction through models and the frequent feedback to developers through builds and test results. The first two papers present work investigating the current state-of-practice of combining CI and MBD and to narrowing down this overall goal to specific challenges. In the remainder of the thesis, we consider one of those challenges: model synchronization, the management of consistency between disparate development artefacts. Different models may be created to describe the same system at different levels of abstraction or from different viewpoints. There often is some overlap between these models, since they describe the same parts of the system, but in different levels of detail. Then it becomes relevant to know if the different models are consistently describing that system, i.e., that they are not contradicting each other. Therefore, we propose a lightweight approach to notifying developers of introductions of inconsistency between different models. Finally, we focus on a part of that problem which impedes applications of this approach in industrial applications with very large models: automatically creating traceability links.

1 Introduction

Modern software systems are getting increasingly complex. An important amount of innovation for many products is in the software, leading to expressions like "This car runs on code" [4]. Indeed, software development nowadays accounts for a significant portion of the development costs of new systems. Hence, there has been a lot of work on trying to improve the productivity of software development and the quality of the resulting artefacts. We discuss two of the paradigms that have been proposed to improve the productivity of software development: model-based development (MBD) and agile software development [3].

In MBD, models are used to abstract from the implementation code [25]. These models are used for design and specification, in which it is beneficial to abstract the implementation details away in favor of an overview of the overall system structure and functionality. Models can be used in all stages of the development process, from using them just for communication purposes to using them to generate the implementation code. In this work, we use the term MBD to refer to practices in which models are used in a formal way in the development of a software system. That means that the models should be used as development artefacts and that they are expected to be consistent with the resulting implementation code. Not in scope of our use of MBD are practices in which models are created just for communication purposes between engineers or as documentation, and in which further development is not based on them.

Since the publication of the agile manifesto [3], software development has become increasingly focused on shortening development cycles. The aim of agile software development is to allow software development to respond quicker to changes in requirements. An example practice is continuous integration, in which multiple developers collaboratively work on a software project and each of them integrates their work frequently in a shared repository. Support tooling then automatically performs builds and tests, giving the developers an up-to-date overview of the status of a project throughout the development and preventing a difficult integration period of uncertain length at the end of the project.

Separately, both these practices have been shown to be beneficial to the productivity of software development in industry [1, 12, 20, 21, 28]. Yet, the application of both practices in combination in industrial development projects is met with scepticism. We hypothesize that applying agile development practices and MBD in combination improves the productivity of software development. In this work, we study the state-of-practice in several industrial environments with the aim of identifying challenges to this combination. Eventually, we aim to propose methods and means to alleviate some of these impediments.

One challenge to model-based development is keeping different artefacts consistent with each other. In a development setup where all artefacts are code, a build system typically notices inconsistent definitions between different portions of code. For example, the code will not be able to be compiled when a class does not implement all the methods defined in an interface. In MBD, it often occurs that no formal links exist between artefacts and hence, these types of inconsistencies might go unnoticed for a long time. The ultimate consequences of this might be late changes to the implementation or even an incorrect implementation. When moving to agile development, with its shorter development cycles and aims of continuously integrating the development artefacts, keeping artefacts consistent becomes both more challenging and more important. Different artefacts will sooner start to rely on each other and hence, inconsistencies may be propagated faster through the different artefacts and can be more impactful and more difficult to resolve. One way to address this challenge is by frequently checking the consistency of different artefacts, through automated consistency checking mechanisms.

Many approaches have been presented to deal with inconsistency between different models. The limitation of much of this work is that the presented approaches require a lot of developer interaction to set up and maintain. Our goal is to create a pragmatic

approach that instead has a lightweight nature, requiring a minimal amount of developer interaction. We study industrial practice to find out what concessions can be made to achieve this goal. This leads us to propose an approach that is less expressive and can not automatically resolve inconsistencies, but is simple in use.

The proposed thesis consists first of two papers investigating the state-of-practice of model-based development and agile development. The last two papers then look specifically into the consistency challenge. The first of these papers presents a pragmatic approach to consistency management. The second paper continues on this path and aims to reduce the required amount of human interaction when establishing correspondence links between model elements. We study the application of a consistency checking approach in an industrial scenario and apply existing naming conventions to automatically establish correspondence links and create consistency checks.

The remainder of this proposal contains a discussion of related work on the covered topics (Section 2), an overview of the followed research methodology (Section 3), an overview of the contributions and the papers in which they are made (Section 4), and a plan for the coming months until the licentiate defense (Section 5). Appendix A contains an excerpt of the individual study plan (ISP) to illustrate that the required activities towards the licentiate degree have been completed or are planned to be completed before the defense.

2 Background and related work

2.1 Combining agile and model-based development

Under the term agile model-driven development (AMDD), several authors have presented work towards introducing agile practices while using models as main development artefacts. Zhang et al. [29] have presented benefits of combining the two paradigms from experiences at Motorola. They present the way in which their development processes were set up to allow for shorter development cycles, continuous integration, and frequent testing. Other case studies also find the potential benefits of applying agile practices in MBD [13]. Lano et al. [18] also advice a process to follow when combining agile and modeling practices, among the tips are to do regular integration and testing. Another case study suggests the use of plant models as an enabler of combining agile and MBD [8]. Given these experiences, there seems to be support for our hypothesis that MBD and agile in combination can boast development productivity.

Of special interest is then continuous integration (CI), one of the key practices in agile development. In paper A, we define CI as: "Continuous Integration is a collaborative development practice where software engineers frequently, at least daily, integrate their work into a shared repository. After each integration, an automatic build is performed. On successful build, automated tests are performed" [15]. This is different from continuous delivery, in which not only the code is frequently integrated, but additionally, a release is made available after each integration. It is also different from continuous deployment, in which not only the code is frequently integrated and delivered, but moreover, every release is automatically deployed on customer devices. We limit our focus on continuous integration, not delivery or deployment.

Some recent work has been published towards methods and processes enabling the combination of CI and MBD. The most important of those works consider more involved modeling practices, in which models are the only development artefacts and code is generated from them. Hence, the problems they identify are closely related to that way of working. Gatcía-Díaz et al. identify model versioning and incremental artefact generation as two problems to combining modeling and CI [10]. Considering a similar level of involvement of models, Garcia and Cabot propose to utilize the continuous delivery pipeline to deal with co-evolution of models, metamodels, and model transformations. In essence,

they chain existing activities and tools using the automation capabbilities of Jenkins[1] In our work, we consider modeling practices in which code is typically hand-written, although it should conform to the created models.

2.2 Inconsistency management

The detection and resolution of consistencies within or between different diagrams of the same model (intra-model consistency checking) or between different models (inter-model consistency checking) has been studied extensively. In this research, we focus on inter-model consistency checking. In particular, we consider consistency between different views of a system, captured in different models that are potentially created using different modeling languages and in different tools.

The importance of consistency in the development process is not disputed [14] but despite the considerable amount of work on model synchronization, it is still considered a challenge to industrial adoption of MBD [24]. Industrial evaluations of multi-view modeling and its consistency problems are lacking [5], perhaps because of the complexity and scale of those environments. Selic identifies the scale of industrial applications as one of the main challenges to overcome for a model synchronization approach to be applicable. In particular, the amount of consistency links can be huge and then the effort to maintain them will be very high and thus at constant risk of being neglected in favor of more pressing issues [26]. Another often identified challenge is a lack of tool interoperability in MBD [19], which naturally complicates the type of consistency management we are interested in.

Several attempts have been made to provide a definition of consistency. Some attempts were made to mathematically define [24] or create an ontology of possible inconsistencies [17] [11]. We consider views that express overlapping concerns to be inconsistent when they contradict each other, as per [23].

It has long been established that complete consistency is infeasible and undesirable, because forcing this would inhibit the software development process. Therefore, the idea of "tolerating inconsistency" was proposed [2]. That paradigm prioritizes devising methods to identify and keep track of inconsistency rather than providing means to automatically synchronize models. In a similar philosophy, Finkelstein et al. have emphasized the possible averse effects of strict consistency enforcement on the development process and have proposed a more lightweight which focuses on consistency of relevant parts of a system design [9]. In a related effort, Easterbrook et al. [7] proposed an approach that tolerates inconsistencies as long as possible during the development process, until they must be resolved. In general, these approaches were found to lack a process-view, i.e., a consideration for the participants and the different phases involved in a software development process [27]. Later work further emphasizes the need for flexible approaches suitable for software development practice, especially when systems grow to incorporate many different artefacts at different times during its evolution [22].

Some recent approaches have incorporated these ideas. For example, Dávid proposes to allow inconsistencies to a certain extent, until their impact on the overall design becomes too big [6]. As a way of reducing the workload of defining consistency rules, Kattner et al. propose to build up a database of correspondences by monitoring normal development activities [16]. Our work focuses on both of these aspects, we propose easy to define and maintain consistency checks that notify developers of inconsistency but do not force consistency.

[1] `https://jenkins.io/`

3 Research methodology

3.1 Research Process

The research was performed in close collaboration with industrial partners. Through our research project in Software Center[2], we have collaborated with three industrial partners: Saab, Volvo CE, and Bosch. We have followed the 6-month sprint structure of the Software Center projects. Where each sprint was started with a research proposal agreed on with the participating companies and concluded with a reporting workshop, in which the results for that sprint were presented.

The result of that iterative process is that early results inspired research questions that were then investigated in later sprints. Figure 1 shows the resulting research process in terms of the order of the contributions and the creation of new research goals. As shown in the figure, the results of papers A and B have been used as input to defining research goal 2 and the results of papers B and C have been used as input to defining research goal 3.

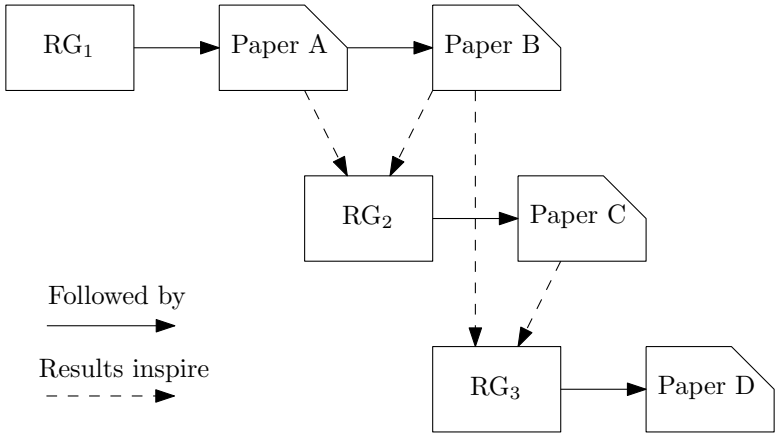

Figure 1: Overview of conducted research process

3.2 Research Goals

We have defined our research goals in close collaboration with our industrial partners and incrementally. Our first goal, was simply to identify the state-of-practice of model-based development and the obstacles towards developing in shorter cycles. After the first results towards this research goal, we started to focus on one of the identified obstacles that was of interest to our industry partners: model synchronization. We proposed a lightweight approach aimed at minimizing the effort in defining and maintaining consistency checks. Finally, we considered a specific instance of such a model synchronization scenario in an industrial context. This required some changes to assumptions we made previously. Most notably, in this industrial scenario we aimed for automated discovery of instance-level links between different artefacts. In summary, our research goals are:

- RG_1: To identify obstacles towards the adoption of continuous integration in model-based development.
- RG_2: To improve the management of inter-model inconsistencies in industrial practice.
- RG_3: To facilitate the creation and maintenance of instance-level traceability links between elements of different development artefacts.

[2]https://www.software-center.se/

4 Results

4.1 Contributions

- C_1: An overview of challenges towards combining CI and MBD.
- C_2: A set of requirements for consistency management approaches to be applicable in industrial practice.
- C_3: A lightweight approach to consistency management.
- C_4: An approach to visualizing inter-model inconsistency in industrial practice.
- C_5: An industrial evaluation of an approach to automated discovery of traceability links between a system model and implementation code.

4.1.1 Mapping to goals

A mapping of the identified contributions to the posed research goals is shown in Table 1.

Table 1: Contributions C_1 through C_5 address research goals RG_1 through RG_3 in this way.

	RG_1	RG_2	RG_3
C_1	X		
C_2		X	
C_3		X	
C_4		X	
C_5		X	X

4.2 Included papers

4.2.1 Paper A

Title: Continuous integration support in modeling tools
Authors: Robbert Jongeling, Jan Carlson, Antonio Cicchetti, Federico Ciccozzi
Status: Published at COMMitMDE workshop at MODELS 2018
Abstract: Continuous Integration (CI) and Model-Based Development (MBD) have both been hailed as practices that improve the productivity of software development. Their combination has the potential to boost productivity even more. The goal of our research is to identify impediments to realizing this combination in industrial collaborative modeling practices. In this paper, we examine certain specific features of modeling tools that, due to their immaturity, may represent impediments to combining CI and MBD. To this end, we identify features of modeling tools that are relevant to enabling CI practices in MBD processes and we review modeling tools with respect to their level of support for each of these features.

Paper contributions: Although there are no major surprises in the results, the work does contribute to the body of knowledge on impediments towards adopting CI in MDE. Further, it strengthens some conclusions made previously by others that have indicated impediments such as tool interoperability and model versioning.

My role: I was the main driver of the work under supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. I performed the tool evaluations and wrote the draft of the paper. The co-authors have reviewed the paper after which I have improved it.

4.2.2 Paper B

Title: Impediments to Introducing Continuous Integration for Model-Based Development in Industry
Authors: Robbert Jongeling, Jan Carlson, Antonio Cicchetti
Status: Published in proceedings of Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2019)
Abstract: Model-based development and continuous integration each separately are methods to improve the productivity of development of complex modern software systems. We investigate industrial adoption of these two phenomena in combination, i.e., applying continuous integration practices in model-based development projects. Through semi-structured interviews, eleven engineers at three companies with different modelling practices share their views on perceived and experienced impediments to this adoption. We find some cases in which this introduction is undesired and expected to not be beneficial. For other cases, we find and categorize several impediments and discuss how they are dealt with in industrial practice. Model synchronization and tool interoperability are found the most challenging to overcome and the ways in which they are circumvented in practice are detrimental for introducing continuous integration.

Paper contributions: The main contribution of this work is the finding that, sometimes, current practices actively prohibit companies from going towards shorter development cycles. We identify those practices and discuss how they are impeding the adoption of CI.

My role: I was the main driver of the work under supervision of the co-authors. The plan for the paper was formed in joint discussions with the co-authors. The work contains interviews at three industrial partners, which me and Jan conducted together at 2 companies. At the third company, I arranged and conducted the interviews by myself. I was the main contributor to the paper writing, the co-authors have given feedback after which I have improved the paper.

4.2.3 Paper C

Title: Lightweight Consistency Checking for Agile Model-Based Development in Practice.
Authors: Robbert Jongeling, Federico Ciccozzi, Antonio Cicchetti, Jan Carlson
Status: Presented at European Conference on Modeling Foundations and Applications (ECMFA 2019); Published in Journal of Object Technology Volume 18, No. 2 (July 2019).
Abstract: In model-based development projects, models at different abstraction levels capture different aspects of a software system, e.g., specification or design. Inconsistencies between these models can cause inefficient and incorrect development. A tool-based framework to assist developers creating and maintaining models conforming to different languages (i.e. *heterogeneous models*) and consistency between them is not only important but also much needed in practice. In this work, we focus on assisting developers bringing about multi-view consistency in the context of agile model-based development, through frequent, lightweight consistency checks across views and between heterogeneous models. The checks are lightweight in the sense that they are easy to create, edit, use and maintain, and since they find inconsistencies but do not attempt to automatically resolve them. With respect to ease of use, we explicitly separate the two main concerns in defining consistency checks, being (i) which modelling elements across heterogeneous models should be consistent with each other and (ii) what constitutes consistency between them. We assess the feasibility and illustrate the potential usefulness of our consistency checking approach, from an industrial agile model-based development point-of-view, through a proof-of-concept implementation on a sample project leveraging models expressed in SysML and Simulink. A continuous integration pipeline hosts the initial definition and subsequent execution of consistency checks, it is also the place where the user can view

results of consistency checks and reconfigure them.

Paper contributions: Many approaches for checking inter-model consistency already exist. The contribution of this work is an approach to checking inter-model consistency that is explicitly lightweight, i.e., easy to use and deploy in industrial settings. Furthermore, the approach is aimed to be general, it can be applied to any modeling language with a hierarchical structure that can be mapped onto a tree structure. It should be seen as a first step towards creating a lightweight consistency checking approach that supports more types of structural consistency and can deal with the evolution of the involved models.

My role: I am the main driver of the work. The plan for the paper was formed in joint discussions with the co-authors. I have implemented the prototype tool and written the first draft of the paper, after which me and Federico have rewritten the paper to its final version.

4.2.4 Paper D

Title: Detecting and visualizing consistency between a system model and the implementation – an industrial case study.
Authors: Robbert Jongeling, Johan Fredriksson[3], Federico Ciccozzi, Antonio Cicchetti, Jan Carlson
Status: To be submitted to: ECMFA (deadline February 21 2020, notification April 24[4])
Abstract: During the development of complex systems, system models can be created as a guide to the final implementation. Creating such a model might even be said to only be useful when the implementation conforms to it. To judge whether an implementation fulfills the model, is consistent with it, some links are required stating which part of the model is implemented by which part of the code. Identifying links between different artefacts and maintaining them throughout the system development is a labor intensive task. In this paper, we explore the possibilities of automatically detecting traceability links between model and code artefacts in an industrial use case. SysML model elements are linked to their C++ counterparts using rules based on naming conventions in use at the company. Given these links, we then visualize the amount of consistency between the code and the model. We report on our experiences of supporting model-driven development using these automated consistency checks in large-scale industry projects.

Paper contributions: The main contribution of this work is an industrial case study of how to create a consistency checking approach given a naming convention between the artefacts on different levels of abstraction. The paper contains experiences on applying consistency checking approaches in large-scale modeling projects. The main contribution is a number of recommendations for implementing consistency checking in MBSE projects, taking into account industrial needs such as scalability and ease-of-use.

My role: I am the main driver of the work. The plan for the paper was formed in joint discussions with the co-authors. I will work in close collaboration with the industrial partner to perform the case study.

4.2.5 Mapping to contributions

A mapping of the papers to the contributions they contain is shown in Table 2.

4.3 Other papers

The following papers will not be included in the licentiate thesis:

[3]An industrial co-author is required for the applications track at ECMFA. The exact author list is TBD.
[4]Alternatively, submit to MODELS applications track, deadline May 8, Notification July 6 (after licentiate seminar).

Table 2: Papers P_A through P_D address contain contributions towards C_1 through C_5 in this way.

	C_1	C_2	C_3	C_4	C_5
P_A	X				
P_B	X				
P_C		X	X		
P_D				X	X

- Robbert Jongeling. "**How to live with inconsistencies in industrial model-based development practice**" Doctoral Symposium at the IEEE / ACM 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS 2019)
- Robbert Jongeling. "**Considerations About Consistency Management for Industrial Model-Based Development**" 12th Seminar on Advanced Techniques & Tools for Software Evolution (SATToSE 2019)

5 Planning

5.1 Time plan

The time plan for the remaining activities of writing paper D and the thesis is shown in Figure 2. Due to the limited time between notification from the prospective venue and submitting the thesis for printing, we include two plans for paper D, where plan B would entail submitting the paper as a technical report.

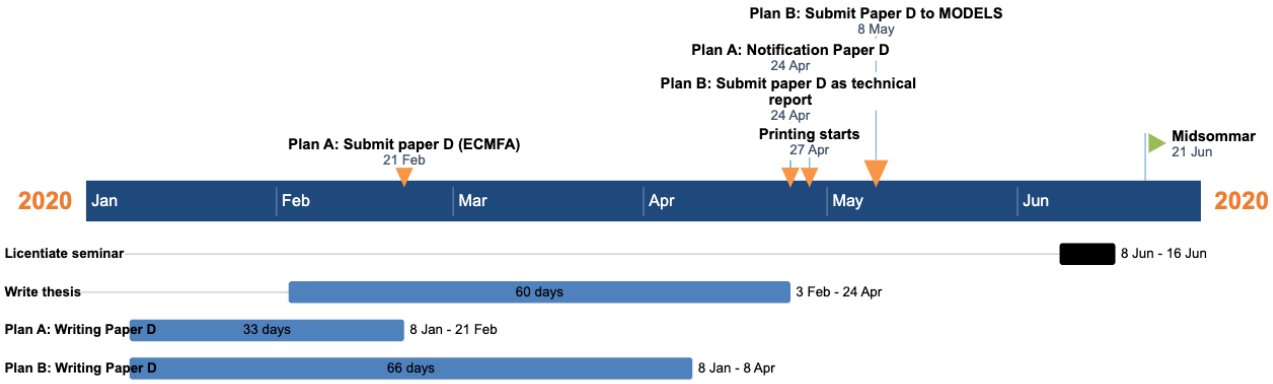

Figure 2: Time plan until licentiate seminar, which in this plan occurs somewhere between and 16 June 2020.

5.2 Thesis outline

The format of the licentiate thesis will be a collection of papers. The following sections are planned to be included in the thesis:

Part I

1. Introduction and motivation
2. Research methods and goals
3. Background and related work

4. Included papers and thesis contributions
5. Conclusions and future work

Part II

1. Paper A
2. Paper B
3. Paper C
4. Paper D

5.3 Progress

5.3.1 Publications

- Paper A: Published
- Paper B: Published
- Paper C: Published
- Paper D: To be written

5.3.2 Courses

The required amount of credits towards a licentiate degree is 45. Of those, 23.5 ECTS have been completed, 11.5 are ongoing and are to be completed, and 10 are planned as seen in Table 3.

Table 3: Overview of courses towards licentiate degree.
*: If unacceptable, could be replaced by a reading course on model-based systems engineering.
**: The last lecture is on June 2nd, so it is likely not in time registered in LADOK before the defense. Will that be a problem?

Course name	ECTS	Status
Model-driven engineering	7.5	Completed
Swedish for employees*	2	Completed
Summer school on cyber-physical systems	2	Completed
Summer school on Software engineering	2	Completed
System of systems engineering	2.5	Completed
Research methods in computer science	7.5	Completed
Quality assurance - Model based testing in practice	2.5	To be completed (Fall 2019)
Introduction to graduate education	4.5	To be completed (Spring 2020)
Research Planning	4.5	To be completed (Spring 2020)
Supervisorship	2.5	Planned (Spring 2020)
Time-sensitive networking**	7.5	Planned (Spring 2020)
Total	45	

References

[1] P. Baker, S. Loh, and F. Weil. Model-Driven Engineering in a Large Industrial Context—Motorola Case Study. In *LNCS 3713*, pages 476–491. Springer, 2005.

[2] R. Balzer. Tolerating inconsistency. In *Proceedings of the 13th international conference on Software engineering*, pages 158–165. IEEE Computer Society Press, 1991.

[3] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. Manifesto for Agile Software Development, 2001.

[4] R. N. Charette. This car runs on code. *IEEE spectrum*, 46(3):3, 2009.

[5] A. Cicchetti, F. Ciccozzi, and A. Pierantonio. Multi-view approaches for software and system modelling: a systematic literature review. *Software & Systems Modeling*, pages 1–27, 2019.

[6] I. Dávid. *A foundation for inconsistency management in model-based systems engineering*. PhD thesis, University of Antwerp, 2019.

[7] S. Easterbrook, A. Finkelstein, J. Kramer, and B. Nuseibeh. Coordinating distributed viewpoints: the anatomy of a consistency check. *Concurrent Engineering: Research and Applications - Concurrent Engineering: R&A*, 2:209–222, 09 1994.

[8] U. Eliasson, R. Heldal, J. Lantz, and C. Berger. Agile model-driven engineering in mechatronic systems-an industrial case study. In *International Conference on Model Driven Engineering Languages and Systems*, pages 433–449. Springer, 2014.

[9] A. C. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multiperspective specifications. *IEEE Transactions on Software Engineering*, 20(8):569–578, 1994.

[10] V. García-Díaz, J. Pascual Espada, E. R. Núnez-Valdéz, G. Pelayo, B. C. Bustelo, and J. M. Cueva Lovelle. Combining the continuous integration practice and the model-driven engineering approach. *Computing and Informatics*, 35(2):299–337, 2016.

[11] S. Herzig, A. Qamar, A. Reichwein, and C. J. Paredis. A conceptual framework for consistency management in model-based systems engineering. In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE 2011; Washington, DC, United States, 28-31 August, 2011*, pages 1329–1339. ASME, 2011.

[12] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical Assessment of MDE in Industry. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pages 471–480. IEEE, 2011.

[13] S. Ilieva, I. Krasteva, G. Benguria, and B. Elvesæter. Enhance your model-driven modernization process with agile practices. In *Proceedings of the 1st International Workshop in Software Evolution and Modernization–SEM 2013*, pages 95–102. Angers Loire Valley, France, 2013.

[14] ISO/IEC/IEEE. ISO/IEC/IEEE 42010:2011(E) Systems and software engineering – Architecture description. Technical report, Dec 2011.

[15] R. Jongeling, J. Carlson, A. Cicchetti, and F. Ciccozzi. Continuous integration support in modeling tools. In *2018 MODELS Workshops*, volume 2245, pages 268–276. CEUR-WS, 2018.

[16] N. Kattner, H. Bauer, M. R. Basirati, M. Zou, F. Brandl, B. Vogel-Heuser, M. Böhm, H. Krcmar, G. Reinhart, and U. Lindemann. Inconsistency management in heterogeneous models-an approach for the identification of model dependencies and potential inconsistencies. In *Proceedings of the Design Society: International Conference on Engineering Design*, volume 1, pages 3661–3670. Cambridge University Press, 2019.

[17] D. Kolovos, R. Paige, and F. Polack. Detecting and Repairing Inconsistencies Across Heterogeneous Models. In *2008 1st International Conference on Software Testing, Verification, and Validation*, pages 356–364. IEEE, 2008.

[18] K. Lano, H. Alfraihi, S. Yassipour-Tehrani, and H. Haughton. Improving the application of agile model-based development: Experiences from case studies. In *The Tenth International Conference on Software Engineering Advances*, pages 213–219, 2015.

[19] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson. Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Software & Systems Modeling*, 17(1):91–113, 2018.

[20] A. Miller. A hundred days of continuous integration. In *Agile*, pages 289–293. IEEE, 2008.

[21] P. Mohagheghi, W. Gilani, A. Stefanescu, and M. A. Fernandez. An empirical study of the state of the practice and acceptance of model-driven engineering in four industrial cases. *Empirical Software Engineering*, 18(1):89–116, Feb 2013.

[22] C. Nentwich, W. Emmerich, A. Finkelsteiin, and E. Ellmer. Flexible consistency checking. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(1):28–63, 2003.

[23] R. Paige, P. Brooke, and J. Ostroff. Metamodel-Based Model Conformance and Multi-view Consistency Checking. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 16(3):11, 2007.

[24] J. Reineke, C. Stergiou, and S. Tripakis. Basic problems in multi-view modeling. *Software & Systems Modeling*, pages 1–35, 2017.

[25] D. C. Schmidt. Model-driven engineering. *IEEE Computer*, 39(2):25, 2006.

[26] B. Selic. What will it take? a view on adoption of model-based methods in practice. *Software & Systems Modeling*, 11(4):513–526, 2012.

[27] G. Spanoudakis and A. Zisman. Inconsistency management in software engineering: Survey and open research issues. *Handbook of Software Engineering and Knowledge Engineering*, 11 2000.

[28] D. Ståhl and J. Bosch. Experienced benefits of continuous integration in industry software product development: A case study. In *The 12th IASTED International Conference on Software Engineering (Innsbruck, Austria, 2013)*, pages 736–743, 2013.

[29] Y. Zhang and S. Patel. Agile model-driven development in practice. *IEEE software*, 28(2):84–91, 2011.

A Completed activities towards third cycle outcome

1. Knowledge and Understanding	Completed
1.1a Broad knowledge	**Yes**
I have completed courses related to the research area (see Table 3) I have attended seminars, Licentiate and PhD Proposals as well as Licentiate and PhD seminars including the following: • "A Model-driven Development Approach with Temporal Awareness for Vehicular Embedded Systems" by Alessio Bucaioni. PhD Seminar, 2018. • Software Center reporting workshop 2018-jun-07 • "Models of Timed Systems" by Edward A. Lee. Seminar, 2018. • Software Center reporting workshop 2018-dec-06 I have attended the schools, workshops and conferences, including: • Software Center reporting workshop 2018-jun-07 • Software Center reporting workshop 2018-dec-06 • Summer school on trustworthiness of cyber-physical systems (June 2019) • Summer school of Software Evolution (SATToSE SESchool, July 2019)	
1.1b Deep knowledge	**Yes**
I have conducted a state of the art study I have contributed with research results via the publications as listed in Section 4.2 and Section 4.3. I have participated in the following workshops and conferences: • CoMMITMDE workshop at MoDELS 2018 conference • STAF/ECMFA conference 2019 • Doctoral symposium at MODELS conference 2019	
1.1c Research methods	**Yes**

1. Knowledge and Understanding	Completed

I have taken the course DVA463 on research methods.

I have reviewed the following scientific articles:

- European Conference on Software Architecture (ECSA) 2019 – **Architectural Rule Conformance Checking in the Continuous Integration Pipeline**
- Models and Evolution Workshop (ME19) – co-located with 22nd International Conference on Model Driven Engineering Languages and Systems ACM/IEEE MODELS'19 — **An Integrated Model-based Tool Chain for Managing Variability in Complex System Design**
- Second International Workshop on Modeling in Automotive Software Engineering (MASE19) – co-located with 22nd International Conference on Model Driven Engineering Languages and Systems ACM/IEEE MODELS'19 — **Simulation as a Service for Cooperative Vehicles**
- IET Software (journal, 2019) – **Removing Object Interaction Inconsistencies in Software Designs**

2. Competences and skills	Completed
2.1a Research questions	**Yes**

I have written the licentiate thesis proposal

I have contributed to the following funding applications:

- Software Center Project Proposal in Sprints 15, 16, 17, and 18

2.1b Contributions to development of knowledge	**Yes**

I am currently supervising a bachelor thesis on consistency checking approaches for UML.

I have developed the following course materials:

- Lab lecture slides for DVA332
- Lab assignments for DVA332

I have assisted in teaching in the following ways:

- Teaching lab lectures for DVA332 (1 in ht2018, 3 in ht2019)
- Supervising groupwork in DVA313 (ht2018, ht2019)
- Teaching lab lectures for DVA436 (vt2019)
- Grading (lab) assignments in each of those three courses

2.2 Oral and written presentation and discussion	**Yes**

2. Competences and skills	**Completed**

I have attended, with own contributions, the following seminars, workshops, and conferences:

- Software Center reporting workshop 2018-06-07
- CoMMITMDE workshop at MoDELS — October 2018
- Software Center reporting workshop 2018-12-06
- SATToSE seminar software engineering, July 08 – July 10 2019
- STAF/ECMFA conference July 15 – July 19 2019
- SEAA conference August 28 – August 30 2019
- Doctoral symposium at MODELS — September 2019

I have participated in the following co-production research/projects:

- Software Center project #32 MoDeCI (VCE, Saab)
- Software Center project #35 MaMi (VCE, Saab, Bosch)
- XIVT (Bombardier)

I have disseminated research in society through

- A 2 minute YouTube video introducing the Software Center project #32;
- Presentations of my project at three Software Center reporting workshops (Jun-2018, Dec-2018, Dec-2019)

2.3 Independent researcher	**Yes**

I have as main driver conducted research for and written all papers listed in 1.1b

I have arranged and conducted research interviews at Océ (in the Netherlands).

I have participated in research discussions in meetings at the following external partners:

- Bombardier (Västerås)
- Bosch (Lund)
- Saab (Järfälla)
- Volvo Construction Equipment (Eskilstuna)

3. Judgement and approach	**Completed**
3.1 Ability ethical assessment	**Yes**

I have participated in the planning of my own research.

I have taken the research methods and research planning courses as listed in Table 3

3.2 Demonstrate understanding of science	**Yes**

I have participated in the seminars listed in 1.1a above.

I have participated in the co-production research/projects listed in 2.2 above.

3.3 Identify need of knowledge	**Yes**

I have participated in the planning of my own research: yes
I have taken the courses listed in 1.1a above.
I have written and published the papers listed in 1.1b above.
I have supervised the theses listed in 2.1b above.