Impediments to Introducing Continuous Integration for Model-Based Development in Industry

Robbert Jongeling, Jan Carlson, Antonio Cicchetti
Department of Innovation, Design and Engineering, Mälardalen University
Västerås, Sweden
Email: {robbert.jongeling, jan.carlson, antonio.cicchetti}@mdh.se

***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.**

***Index Terms*—Model-based development, Continuous integration, Interview study**

I. Introduction

For the design of complex modern software systems, model-based development (MBD) is often leveraged, i.e., models are used as core artifacts for activities like system design, simulation, and code generation [1]. The models are core artifacts in the sense that the eventual code and application match them and the models are not used, for example, just for informal communication. Industrial practice implies collaboration on these models by multiple engineers, possibly from multiple domains. Empirical results show improved productivity of development in industrial settings when that includes using models in this way [2]–[5].

In parallel, Continuous Integration (CI) has also been evaluated in industrial settings and shows an improvement in the productivity of software development [6], [7]. CI proposes a collaboration in which developers frequently (at least daily) integrate their work into a shared repository [8]. Notably, CI does not entail continuously making a release available (as in continuous delivery) or continuously deploying the software on user machines (as in continuous deployment). The respective scopes of these practices are illustrated in Figure 1.

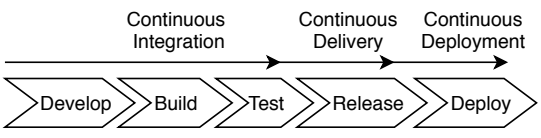

Fig. 1. Steps included in continuous integration, continuous delivery, and continuous deployment.

To optimally make use of their individual benefits, we consider the combination of CI and MBD. This entails multiple developers using models as core artifacts for development and frequently integrating new versions of these models into a shared repository. In our experience, industrial adoption of this combination of practices is low, despite several evaluations showing its potential benefits [9], [10]. More specifically than agile, combining CI and MBD is also identified as a promising practice towards increased development productivity [11], [12]. Understandably, introducing these development methods in industry is not an overnight process and many obstacles are encountered. To identify the most important of these impediments, we have interviewed industry practitioners who share their experience and expectations for the future from different perspectives.

The remainder of this paper is organized as follows. In Section II, we describe the design of the interview study, the findings from which are then presented in Section III and further discussed in Section IV. Relevant related work is considered in Section V and the paper is concluded in Section VI.

II. Research approach

A. Context

We consider the combination of CI and MBD to entail a practice in which models are developed in rapid iterations and developers integrate their work frequently into a shared repository. When models at different levels of abstraction are created, for example for system design and software implementation, they should be synchronized. Changes to any model could incur a build and test run, as part of the CI pipeline. A common current industrial practice is development using the "V-model" [13], so initially this CI can be seen as an enhancement of some steps in that process, as illustrated in Figure 2.

By means of interviews with industry practitioners of MBD, and to some extent also CI, we reflect on the difficulties of adopting this combination of CI and MBD in practice. Specifically, we aim to answer the following research question: *What are the experienced and perceived impediments to introducing continuous integration in model-based development projects?*

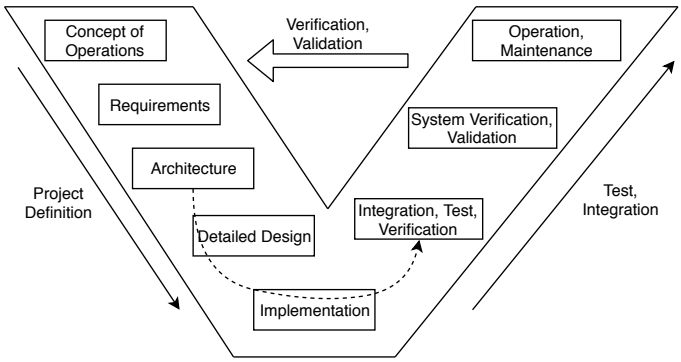

Fig. 2. Continuous Integration concerns the steps in the bottom of the V-model [13]. Modelling artifacts belonging to these blocks may be subject to rapid iterations that may impact modelling artifacts in the other blocks, as indicated by the dashed arrow.

B. Interview Design

In this work, we consider the perspective of three large companies, referred to as company 1, company 2, and company 3. Two of the companies are based in Sweden and one is based in the Netherlands. The companies develop software for embedded systems in the varied domains of avionics, electronics, and vehicular embedded systems.

We have interviewed eleven engineers, five at company 1, two at company 2, and four at company 3. The interviewees have various roles in the companies, such as system architect, system designer, software engineer, or software integrator and have varied levels of experience, from an interviewee being involved in modelling only since its company introduction in the last 2 years to an interviewee who has been involved with modelling for more than 25 years.

At two companies, the interviews were conducted by two interviewers, at the other company the interviews were conducted one-on-one. The interviews at one of the companies were in person, the others remote, through Skype and by phone. Audio of eight interviews and detailed notes of all interviews were recorded. The notes were summarized and sent to the interviewees for confirmation and discussed extensively between the interviewers.

In the design of the interviews, we have included some measures to alleviate threats to external, construct, and internal validity as well as threats to their reliability, using the categorization of these threats by Runeson and Höst [14].

C. Threats to validity

A threat to the external validity of this work is unjustly drawing generalizing conclusions based on a too narrow sample. We alleviate this threat by including companies who implement MBD to different degrees and thus have the required different perspectives. Albeit a small sample size, we synthesize our results such that they are nevertheless relevant for companies that have adopted agile and MBD to similar degrees as the interviewed companies.

To alleviate threats to construct validity, in particular the threat of the interviewers and the interviewees having a different idea in mind when talking about agile MBD, we presented our views on these concepts before starting each interview. In this brief introduction we presented definitions of MBD and CI, as well as what we mean by their combination, similar to the introduction of this paper. Furthermore, during each interview, the interviewers have taken detailed notes that were, as mentioned, summarized and sent for review to the interviewees, such that they could confirm that we reflected their statements correctly or to allow them to correct misunderstandings.

The internal validity of our study can be impacted by interviewers not having a complete picture of the context in which the interviewee is working. Consequently, the interviewers could unjustly attribute certain impediments to certain practices. To alleviate this threat, we started the interviews with questions to determine the current state of practice at the companies and used this information after the interviews to discuss the results between the interviewers to understand from what viewpoint the different comments originate.

Towards the reliability of any interview study, there is a trade-off between a strict set of completely reproducible closed questions and an open conversation highly influenced by the personal input of the interviewers. We have balanced these interests by creating semi-structured interviews following the pyramid model [14]. Early questions are closed and specifically related to the current state of practice at companies and the professional background of the interviewees. In later stages of the interview, the questions are open and only the themes decided beforehand. During that stage, we ask the interviewees to identify impediments to combining CI and MBD, which allowed us to then talk more in-depth about these identified impediments by asking follow-up questions to explore them further.

III. Findings

In this section, we present the results from the semi-structured interviews. First, we briefly describe the current state of practice at the industrial partners. Then, we categorize perceived and experienced impediments as identified by interviewees.

A. State of practice

All three companies are, to different extents, model-based development practitioners and also have adopted agile development practices to different extents. Figure 3 sketches the relative positions of these companies on both spectra, where the horizontal axis denotes adoption of models in development and the vertical axis denotes adopting agile practices with respect to the development of those models.

Company 1 utilizes models to create system designs at a high abstraction level. At this level, the design is mostly concerned with subdividing the system into clusters of software components and specifying their interfaces. After this stage, the models are handed over to software teams, who implement the designed components in code. There are some agile processes in place in the software development part,

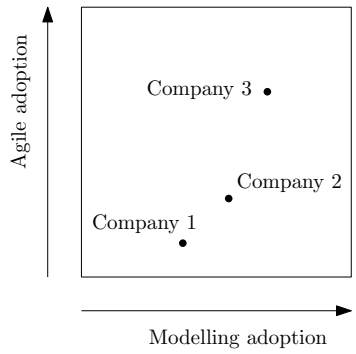

Fig. 3. Relative positioning of involved companies based on their adoption of agile and MBD practices.

but not involving the development of the system models. Iterations of the models are spread out over a long time and are usually made far in advance of the software development, making system models and the software implementation quite decoupled. Changes to the system design are communicated by system designers to software engineering teams after each new iteration by means of a manual handover, in which the teams agree on the changes and their implications.

Company 2 works in a similar way; a system design is created using models at a high level of abstraction, which are not directly linked to the software implementation. Additionally, a portion of the software is implemented in models from which code is generated. CI is used in the development of these software models, it is done in rapid iterations and they are integrated very frequently, up to multiple times per day.

Company 3 has a continuous integration pipeline in place for software models, all code is generated from them. For the design of the system, models are used in a less formal way, embedded in documents and only meant for communication of the design ideas. Since they are used in this way, the models do not necessarily strictly conform to their language syntax and no explicit links exist between these design models and the software models.

It should be noted that we considered only companies who already employ models for the development of their software systems. So, we are investigating only how to adopt CI once modelling is to some degree in place, rather than considering also the symmetrical case, where CI is in place but models are not used at all. Given the known challenges in adopting modelling, it is more in line with industrial practice to consider the introduction of CI in MBD, rather than supposing that modelling is introduced in an already agile, but code-centered software development process.

B. Conservative views

Early on in our research, we encountered MBD setups in which there is no need nor desire for more rapid development iterations, contradicting our initial assumptions by saying for example that there is no need for CI and that the current practices are good enough. An example of such a case involves a clear distinction between the design and implementation phases. In the design phase, system models are used to describe the system at a high level of abstraction. In the implementation phase, models or code are used to design the software, but there are no formal or automated links between the system models and the implementation models or code. Therefore, in both of these cases, the synchronization between design and implementation is done manually, either by communication between system design teams and implementation teams, or by having the same engineers work on both parts. Rapid iterations are then undesirable because they would only increase the communication overhead between teams, or impractical, when both phases are performed by the same engineers but the implementation phase starting after the design phase is completed.

To better align our questions with these practices, we have first asked interviewees about their views on potential benefits to introducing more models in the software development, we then asked them to think about impediments they see towards that goal. Having established first the introduction of models throughout development, we moved on to questions regarding the benefits of applying CI in modelling contexts and perceived impediments towards doing so. The questions thus follow the expected adoption process of agile modelling, i.e., first introducing MBD, then going more agile. This order of the questions was chosen to separate concerns interviewees might have regarding the introduction of modelling from the impediments they encountered or expect to encounter when introducing CI.

Using more models in development can refer to extending modelling practice from system design to software models. Alternatively, it can refer to using models more formally in all development stages, rather than for example only for communication of system design concepts. The opinions are divided on the formality required in the created models, where more strictness is seen as a must for introducing more automation, but on the other hand, some engineers would like to have more freedom in a modelling tool, for example by writing free text in models, to enable easier communication of design ideas.

When code is written manually, interviewees view the potential productivity gains of introducing CI in MBD as minimal. But at the same time, modelling for code generation encounters some apprehensive views. Firstly, enabling code generation requires modelling to a low level of abstraction, which is a big step when the current practice includes only modelling on system level. This causes some reserved reactions from engineers stating that modelling is more difficult than coding for certain concepts, such as parallelism: *"Things are sometimes very difficult to describe in models, perhaps software on a higher control level can be described, but parallel processes and what is described in VHDL on our FPGAs, ..., complete parallelism is impossible to describe, at least in SysML. All those continuous flows, I don't know how to describe those."* Similarly, some doubts are shed to the applicability of modelling in their domain, or for their specific products: *"code generation is good if you want to make very*

simple things." These types of comments illustrate some of the conservative viewpoints in industry, but more experienced modellers have expressed their wonder at the resources spent on writing code while it could be generated from models: *"I don't see why we have to write code anymore."*

More rapid iterations in the system design might also not be considered relevant because the design is often made far in advance of the software implementation and is not so flexible but only updated for very relevant changes. Another viewpoint is that it would hinder the freedom of developers, as said by one interviewee about introducing more formal modelling at system design level (instead of using models only for communication): *"I think in some situations this might help a little, but mostly it would be experienced as a limitation if those high-level diagrams have to be correct."*

CI for models at the implementation level is in place in two of the companies in this study, although those models are not explicitly linked to the system design models. It is noteworthy that both companies use, for this stage, a single modelling tool, simplifying the implementation of a CI process. More rapid iterations involving both the system design and the software implementation are not considered useful due to the previously mentioned factors. The system design is typically to a large extent completed before the implementation starts and is often created for the purpose of communication, so the models are not a precise description of the design. Furthermore, as discussed before, the process of synchronizing between the phases often involves communication between people, which scales badly to faster iterations. Rather, a more explicit coupling between the models at the different abstraction levels is desired, such that the impact of changes in the design are clearer to the system designers and the required changes in the implementation are easier communicated to the implementation teams. As reflected by the following statement about more rapid iterations between models on all levels, *"If the models would be more connected, then that would work much better"*.

C. Impediments

The conservative views of some interviewees were not shared by all interviewees in companies 1 and 2. Some engineers did see benefits in moving towards CI even with current modelling practices. *"Yes obviously, we would like to have that, because it's working quite well in the software area, but it's hard to get to a point where it is convenient, especially in the modelling world."* In general, the benefits of a short turnaround loop between software and system design are appreciated, but thinking about doing CI in MBD may be a bridge too far. Many other problems need to be solved before getting to a stage where these practices are useful enough to be increasing productivity. It seems that the views of the engineers with conservative views to introducing CI in MBD are influenced by the impediments they foresee towards its implementation.

We have collected impediments from different perspectives, by asking engineers in companies 1 and 2 what they perceive as impediments now, and expect as impediments in the future, towards implementing CI in MBD. Engineers in company 3 were asked to reflect on their implementation of CI and the biggest obstacles encountered, as well as looking ahead to expected impediments when further streamlining their existing development processes. We categorized the obtained impediments as those having causes based on the desired functionality of the combination of CI and MBD, causes related to non-functional elements, such as the development process, human causes and business causes. A summary of those findings is provided in Table I. We now elaborate each category and each identified impediment.

TABLE I
SUMMARY OF IDENTIFIED IMPEDIMENTS TO INTRODUCING CI IN MBD IN THIS PAPER, IN DIFFERENT CATEGORIES OF CAUSES.

Functional	Lack of tool interoperability Lack of synchronization between models Requires model validation Lack of model merge support Lack of configuration management
Non-functional	Too long time needed for builds and tests Lack of impact analysis Tooling frustrations
Human	Lack of modelling expertise Lack of willingness to model Difference in modelling styles
Business	Difficult to enthuse management and colleagues Lack of time/knowledge to set up tool-chain Domain-induced complications

1) Functional: A CI practice in which models at multiple levels of abstraction are included requires a tight coupling between all models, such that automatic builds and tests can provide insight into the state of the integration. Since models describing the system at different levels of abstraction and from different disciplines are typically expressed in different languages and created in different tools, this requires multiple tools that work well together. While engineers endorse this need: *"An integration would be good to have, between different tools, different categories of system designs"*, in practice, getting tools to cooperate is very challenging. *"I don't really believe strongly in having several tools if they are not really tightly integrated, but then it's the same tool."* As another interviewee stated, on the impediments to more frequent integrations of models on all levels: *"one reason is the different tool vendors, which are not integrated."* Companies have dealt with these tool interoperability challenges mostly by avoiding it. The models from which code is generated are all created in the same tool. Other models, for example system models, are created in different tools, but are connected to each other only informally, so there are no automatic checks between them or formal definitions evaluated to check that these models express the same design.

Indeed, this lack of tool interoperability consequently contributes to development in which the synchronization between models is a manual task, increasing overhead and decreasing the ability of continuously integrating new model changes. As

discussed earlier, a more explicit coupling of these models is desired. As one interviewee underscored with the following statement about system and software models: *"We have a need to integrate it really."* Or, particularly about generation of code and the creation of a feedback loop between models at different levels: *"It would be good to have a nice turnaround from design to code and from code to design. I think we will always need the possibility to change code. Because of performance reasons or maintenance reasons.* In a CI context, the state of the integration should be known after each build, but this is greatly complicated if there is no automatic support to synchronize models or check consistency between them. *"Using more models would be hard if there are no consistency checks, because if they are standalone they will never be the same."* Furthermore, mistakes due to model inconsistencies can be propagated more rapidly in a CI context.

Towards the same goals of creating an automated pipeline for building and testing, model validation is required, but typically minimally in place. This refers to syntax checking within models, since automation requires more formality from the included models, as well as consistency checking between models. The former is often in place inside modelling tools, but are not always used: *"Continuous validation of the model would be very convenient, the first step would be to get help from the tool in validating. Just get rid of all these stupid errors that you might introduce. For example, scoping, some of these errors have really serious consequences."* When models are used only for communicating designs, even less strict validation is desired by some engineers, for example the ability to write free text in some models. Consistency checking between models is hampered by the previously mentioned impediment of tool interoperability, but not felt so strongly since the models are not explicitly coupled. So, inconsistencies have no direct effects in terms of failing builds or tests, but rather may subtly affect the subsequent development, possibly incurring late and costly changes if they are noticed late. The validation of system design models is completely manual, since they are not connected to the implementation models. These manual actions do not scale to larger models or more rapid iterations and are thus hampering the move towards CI.

A common impediment to introducing MBD in industrial settings is the lack of good version control systems for models, one part of the broader challenge of collaborating effectively on models. As one interviewee said: *"for us system engineers, this is one of the hardest parts, to share the model and not interfere with each other more than we have to."* Different to code, line-based diffs of XML representations of models are not helpful in indicating the differences between models and merging them. Particularly, the graphical representations of models are difficult to version, which is problematic especially when models grow large (and they typically do). Engineers mention a lack of model differencing in current practice: "*We have no good ways to merge models. It's even worse, we have no good ways of comparing models."* As well as too many manual steps required to obtain differences. *"There is no really good way to get a delta out of the model. To get that delta the system engineer manually has to go through and mark what is changed. To make a diff on the model, we don't have any good tools to do that."* In practice, the lack of faith in merging has two consequences. First, merging is circumvented by locking models for changes, thus avoiding the need to merge. Second, the system design is divided and tasks are assigned in such a way that the need for concurrent changes to the same model are avoided as much as possible. Locking models is a good enough solution for small teams, but already involved *"a lot of legwork"*, where intense communication was required between engineers to allow synchronous collaboration. So, locking does not scale well to larger and possibly distributed teams. It also impedes the introduction of more rapid iterations on the models. The strict division of the model is similarly impeding the agility of development. Rather, each contributor should be able to make changes at any place in the system. This lack of support for merging drives both these sub-optimal and non-scaling development practices. Notably, configuring the CI pipeline for automatic merging was also named as the biggest overcome challenge in company 3 and something that still can be improved to allow for more parallel work.

Version control is one of the components of configuration management, which manages among other things the change history and deployment of specific versions of the software on specific platforms. In general, in software engineering, configuration management is challenging, especially when software needs to be supported for a long time (possible decades) after it is first produced, thus requiring the possibilities to make changes and test them in old configurations. This is also a challenge in MBD, and a problem when trying to do MBD in more rapid iterations, since the tooling ecosystems are typically fragile. *"I think configuration management and better tooling are what we need."*

2) Non-functional: In this category, we include those impediments that are not directly related to current tooling or other technical problems, but are rather related to non-functional elements such as current practices and development processes. A first example of a current practice hindering the introduction of more frequently integrating is the duration of builds and regression tests. In a similar vein, a large amount of computation power is required for extensive simulations including all models. Initially, such problems can be (and have been, by company 3) avoided by allocating more computing power to these tasks. But this is of course addressing the symptom and not the cause, and will eventually also be insufficient. This is not an insurmountable problem, but it is one of the practical hinders that are encountered when introducing agile MBD processes.

Another such practical problem is partly caused by the current division of development teams between system modellers and software designers. Changes in a system model impact lower level models, but to the system modellers, it is not always clear how. This also works the other way around; the software designers are not aware of the exact changes in the system model and the entailed required changes in software models. Both effects are strengthened by the

size and complexity of the models: *"the (system) model is complex organized, the developers don't know where to look for information."* Consequently, communication, sometimes through documentation, is needed to align the activities of different modellers. This also does not scale well to larger settings and more rapid iterations. The alternative, more formal and tool supported impact analysis, requires a more formal usage of the models. If the system models are only used to communicate design and do not strictly adhere to some syntax, or do not capture the exact semantics, then trying to automatically assess impact of changes is hopeless.

Naturally, an implementation of CI will require several tools, current manual practices are not good enough to just be done more frequently. *"There are too many manual steps, too little automation."* It is therefore not a good sign that already, often ventilated frustrations have to do with tooling. *"It is slowing us down."* Or, as indicated by another quote regarding using modelling in more phases of development: *"The modelling tool is not so stable, it crashes and it freezes and everything goes slow. You want to have quicker tools, if it was quicker and easy to understand then you could use it more."* While tool instability is not a hinder only to introducing CI, since it also hindering current MBD practices, it is still relevant to mention here, since the potential benefit of introduction of CI in MBD depends heavily on tool support. These are comments about single modelling tools and contribute to a skeptical view about involving more automation in the development process. *"People don't want to use 5 to 10 different tools."* One interviewee described current CI practices, for non-MBD projects, as involving *"a lot of small steps and something is always broken."* Given earlier comments on tool interoperability and tool instability, the interviewees seem not to expect that this is getting any easier when setting up a CI pipeline for MBD.

3) Humans: When discussing these functional and non-functional impediments, we cannot overlook the human aspects, which remain present even if perfect tools are created and used in the perfect process. Most of these impediments have to do with the inherent complexity of modelling, *"not enough people know SysML."* Furthermore, a steep learning curve needs to be overcome to start contributing to models, *"it is hard to learn how to model, it takes time to be a good software modeller and it is even harder to be a good system engineer."* Besides this general modelling knowledge, the complexity of the product sometimes just requires a lot of experience. *"Some parts require so much domain knowledge that you probably need to work here for ten years before you can contribute."* This contributes to a lack of willingness to learn modelling, it seems that modelling has an image problem, people are scared off and do not want to model, despite the views of some interviewees: *"it's fun to model."*

Another human factor is a lack of alignment between modelling practices of different developers. Different styles of modelling can lead to different subdivisions of models, and difficulties in understanding large models, if parts are created in different ways. This is a problem similar to traditional, code-based software development, in which the use and enforcement of coding standards is well established. For this aspect, the main difference between the code and model-based industrial practice is tool support. Several interviewees remarked that the tooling lacks, e.g., checking of conformance of models to design guidelines.

4) Business: In addition to these human factors, impediments were mentioned that are related to the business perspective. Any change in process and tooling requires an investment, be it time, money, or both. The advantage of introducing these processes is often not easily quantified, making it difficult to gather support for them.

A difficulty in achieving a complete modelling pipeline can also be to get all involved disciplines in a company on board. A coupling between models from different domains is desired, but a lack of willingness and resources to do so hinders this synchronization between parts of a company. It might be that engineers estimate the amount of required resources as high, due to other functional or process impediments they see.

Further, engineers mentioned a lack of time and knowledge to invest in setting up a toolchain. Especially considering the need for customization of such toolchains, since almost no two companies are working with the same sets of tools. Furthermore, this customization depends on the type of product developed. *"It depends a lot on the domain what the generated code should exactly look like."* The domain furthermore impacts the required lifetime of products, complicating, as mentioned earlier, configuration management. Another related challenge that might apply is the need for the developed code to conform to strict regulations and certifications.

D. Future visions

Some alternative future visions were proposed, which would make this more useful and more possible. One of them describes an extension of existing tooling in to other modelling domains, such that all modelling activities, from system architecture to software implementation, can be performed in a single tool. An alternative vision assumes that engineers from each domain will keep using their preferred tools, but aims rather at better interactions between these tools. Ultimately, both these visions allow for CI involving all models, by resolving one of the main seen impediments by practitioners: a lack of tool interoperability.

IV. Discussion

The identified challenges in Section III paint part of the picture of the impediments towards adoption of CI in industrial MBD practice, by considering the different points of view from the different industrial partners. Still, the sample size is not big and we should be careful to generalize our findings to cases in which CI and MBD are adopted to different degrees than in the interviewed companies.

The division of the impediments in the different categories emphasizes the broadness of the encountered challenges. Indeed, a silver bullet does not exist, rather, it is a long process to introduce modelling and then CI in industry practice. While for

our research, the functional and non-functional aspects are the most interesting to focus on, the human and business aspects should not be disregarded.

Considering the adoption of CI in modelling in the involved companies, it is noteworthy that in company 3, modelling and CI works well, using a single modelling language. Company 2 as well uses a single modelling tool for software models and develops them in rapid iterations. Tool interoperability and model interoperability is in these cases to a large extent avoided. Nevertheless, implementing a CI pipeline in which models are included for system design, detailed design, and implementation requires information sharing between these models and thus communication between the tools in which they are created. Indeed, these two elements are central to introducing CI in MBD and they make resolving the other named impediments more complicated. For example, impact analysis is more complicated when the impact must be assessed across modelling languages.

Considering these findings in another way, we can say that there are few impediments to introducing CI when modelling for code generation using a single modelling language. Rather, most of the found impediments are encountered when models are also used in other parts of development, such as system design. Then, maintaining sound architectures and consistency between the models is increasingly challenging, particularly in a CI environment. An apparently practiced way to circumvent many of these impediments is by applying CI only to the lowest level models, those used for code generation, while manually managing the correctness of the other models. The downside of this way of working is that it benefits minimally from one of the main promises of CI, i.e., always having an overview of the state of integration, since that state cannot be completely known using only implementation models.

As also found, introducing CI is not always the desired approach, we have seen a current way of working in which introducing more frequent integrations is not useful. In that case, because there is a strict separation between design and implementation and because many manual steps are involved that would not scale up appropriately. It should be noted that the starting point of a company is crucial in how engineers view this aspect, conservative views are natural given a well-functioning process and foreseen serious impediments to changing them. Furthermore, if the benefits of their introduction are not clear, gathering support for CI practices is difficult. Finally, the domains in which the companies work and the traditions that those bring with them seem to impact the willingness to adopt faster development cycles.

V. Related work

Our study has underscored some results that were found earlier, when investigating the adoption of modelling practices in industry. In their experience report from 2005, Baker, Loh, and Weil [2] already note a lack of performance of tools and interoperability between different tools. Other empirical studies also point to tools as impeding more MBD adoption [4]. In addition to these technical issues, Hutchinson, Whittle, and Rouncefield [15], also using semi-structured interviews, find organizational (process and business-caused) factors important to the success or failure of their introduction. Recently, an evaluation of industrial MBD practice shows its potential benefits but also highlights the difficulties in its adoption, particularly tool interoperability and a steep learning curve of the method itself and tools used for it [5].

In a similar industrial context as our study, an interview study has shown the state of practice and impediments to introducing continuous deployment in agile software development projects [16]. While the paper does not consider modelling, it does identify some impediments that are also relevant in our case. Notably, in companies moving towards CI, one of the impediments is the complexity of test automation. This aspect is underexposed in our interviews, possibly because many of interviewees were doing system modelling and did not operate closer to the implementation.

Another interview study involving large industrial partners identifies four categories of impediments to introducing CI, albeit not in modelling projects [17]. The categories identified in that work are related to testing processes, the usability of tools, the splitting of the system into parts and the division of work among engineers. Some overlap can be noticed between those results and our results, particularly the impediments related to the process.

In our earlier work [18], we have reviewed modelling tools and their suitability to be applied in the context of CI and MBD. There, we concluded CI is achievable when using a single modelling tool, albeit possibly challenging to set up, but much more challenging when using a combination of modelling tools. This aligns with our findings in this work, where we have seen one company where a pipeline is implemented, but for a single tool. Further, we have seen in all companies that introducing or streamlining CI practices raises challenges in tool interoperability, and model interoperability, when using different modelling tools and modelling languages.

A systematic literature review in the area of combining agile methods (of which CI is one) and modelling has shown the immaturity of the field [19]. In particular, the authors have argued for more reports on industrial experiences. One such study focused on introducing agile practices in modelling projects in the automotive industry [10], although not including CI yet, the authors do report a successful application of agile methods. Similarly, a case study in the telecommunications domain has shown benefits of agile MBD [9], although it minimally discusses the extent to which this practice includes continuous integration. This paper contributes to this knowledge base by exploring views and current practices of industry practitioners of agile development methods in MBD projects.

Other work has reported on experiences of introducing CI in an MBD project [11]. The authors show the benefits from combining CI and MBD to the development process and discuss hurdles overcome to achieve this. An impediment identified in common with this work is the lack of support for differencing and merging of models.

Other work has considered practical applications of CI

in the automotive domain [20]. While not explicitly about modelling, some identified impediments are closely related to our findings. In particular the many manual steps that are performed to move data in between tools, due to a lack of tool interoperability. Further, the authors identify many organizational issues, besides the tooling, which we have touched upon in the category of business impediments.

Although not evaluated in industrial practice, some initial works have sketched the possibilities to wrap modelling tools for each step in the process of modelling to validation in a continuous delivery pipeline [21]. This addresses the tool interoperability challenge for one specific set of tools. The authors stress the need for maturity of individual tools and steps in the pipeline and summarize the challenges of creating a CI pipeline for models by stating that all steps in it should be "model-aware."

VI. Conclusion

In this work, we have identified several impediments to introducing continuous integration (CI) in model-based development (MBD) through eleven interviews at three large companies. Furthermore, we have discussed how some of those impediments are circumvented in current practice. The three companies each have a different current way of working and therefore a different starting point when introducing or streamlining existing CI practices. Some starting points imply that introducing CI is not desired by engineers. Other starting points did see an embrace the idea of combining CI and MBD, but several impediments were named towards that goal. We have categorized these impediments as functional, non-functional, human, and business-related.

To broaden the coverage of our findings, in future extensions of this work, we aim to include more companies on different positions in the graph in Figure 3. For instance, by including companies that are doing more modelling than company 2 and less CI than company 3. Moreover, we aim to include companies that have implemented more explicit links between all models.

To answer the research question posed in the introduction: the impediments to applying continuous integration in MBD projects are summarized in Table I and in addition also implicitly include general impediments to introducing MBD. Our two main findings of this interview study are 1) that introducing CI is not always desirable or useful given a current way of working, and 2) workarounds to common problems of tool interoperability and model synchronization are impeding the introduction of an automated CI pipeline for MBD.

Acknowledgment

The authors would like to thank all interviewees for their time and input in fruitful discussions. This work is partially supported by Software Center[1] and by the Knowledge Foundation in Sweden through the MINEStrA project.

[1]www.software-center.se

References

[1] D. C. Schmidt, "Model-Driven Engineering," *IEEE Computer*, vol. 39, no. 2, p. 25, 2006.

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

[3] 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)*. IEEE, 2011, pp. 471–480.

[4] 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*, vol. 18, no. 1, pp. 89–116, Feb 2013.

[5] 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*, vol. 17, no. 1, pp. 91–113, 2018.

[6] 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*, 2013, pp. 736–743.

[7] A. Miller, "A hundred days of continuous integration," in *Agile*. IEEE, 2008, pp. 289–293.

[8] M. Fowler and M. Foemmel, "Continuous integration," *https://www.thoughtworks.com/continuous-integration*, 2006.

[9] Y. Zhang and S. Patel, "Agile model-driven development in practice," *IEEE software*, vol. 28, no. 2, pp. 84–91, 2011.

[10] 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*. Springer, 2014, pp. 433–449.

[11] 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*, vol. 35, no. 2, pp. 299–337, 2016.

[12] H. Alfraihi, K. Lano, S. Kolahdouz-Rahimi, M. Sharbaf, and H. Haughton, "The Impact of Integrating Agile Software Development and Model-Driven Development: A Comparative Case Study," in *International Conference on System Analysis and Modeling*. Springer, 2018, pp. 229–245.

[13] INCOSE, *Systems Engineering Handbook, v3.2.2*, 2011.

[14] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[15] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming*, vol. 89, pp. 144–161, 2014.

[16] H. H. Olsson, H. Alahyari, and J. Bosch, "Climbing the 'Stairway to Heaven'–A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software," in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. IEEE, 2012, pp. 392–399.

[17] T. Mårtensson, D. Ståhl, and J. Bosch, "Continuous integration impediments in large-scale industry projects," in *2017 IEEE International Conference on Software Architecture*. IEEE, 2017, pp. 169–178.

[18] R. Jongeling, J. Carlson, A. Cicchetti, and F. Ciccozzi, "Continuous integration support in modeling tools," in *Proceedings of MODELS 2018 Workshops co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems)*, 2018, pp. 268–276.

[19] H. Alfraihi and K. Lano, "The Integration of Agile Development and Model Driven Development – A Systematic Literature Review." in *MODELSWARD*, 2017, pp. 451–458.

[20] E. Knauss, P. Pelliccione, R. Heldal, M. Ågren, S. Hellman, and D. Maniette, "Continuous integration beyond the team: a tooling perspective on challenges in the automotive industry," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2016, p. 43.

[21] J. Garcia and J. Cabot, "Stepwise Adoption of Continuous Delivery in Model-Driven Engineering," in *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 2018, pp. 19–32.