Simula Research Lab Technical Report 2011-18

Research-Based Innovation: A Tale of Three Projects in Model-Driven Engineering

Lionel Briand, Davide Falessi, Shiva Nejati, Mehrdad Sabetzadeh, Tao Yue
Certus Software V&V Center, Simula Research Laboratory, Oslo, Norway
Email: {briand, falessi, shiva, mehrdad, tao}@simula.no

Date: October 27, 2011

[simula . research laboratory]
- by thinking constantly about it

Abstract

Engineering research needs to be informed by (or based on) industrial needs to have impact, and industrial innovation depends on research to fill the gaps in knowledge and to pave the way for better tools, technologies, and services. In the past few years, we have been exploring ways to foster a closer collaboration between research and industry both to align our research with practical needs, and to further increase awareness about the important role that software engineering research plays as an enabler for innovation. This paper outlines our experiences with three recent and successful research projects conducted in collaboration with the maritime and energy industry. We take a retrospective approach to examine the way we collaborated with our industry partners and elaborate the decisions that we believe contributed to the effectiveness of the collaborations. We report the lessons learned from our experience and illustrate these lessons using examples from the three projects. The lessons focus on the applications of Model-Driven Engineering (MDE), as all the three projects we draw on here were MDE projects. Our goal from structuring and sharing our experience is to contribute to a better understanding of how researchers and practitioners can collaborate more effectively and to gain more value from their collaborations.

1 Introduction

Research and innovation go hand in hand in all engineering disciplines and software engineering is no exception to this rule. Unless engineering research and innovation are done in tandem and synergistically, both will suffer: research may be poorly aligned with the "pain points" of the industry and will consequently have limited impact; and innovation will be hampered if the industry is deprived of an inflow of creative ideas and solutions stemming from research.

Motivated by the above, we have been seeking in the past few years ways to collaborate more closely with industry, both to ensure better alignment between our research and the current industrial needs, and further, to demonstrate to our industry partners the role of software engineering research in boosting innovation. This is what we refer to as *research-based innovation*.

This paper discusses our experiences with three projects that have reached maturity, selected from a larger set of ongoing projects that we are currently conducting in collaboration with industry. We reflect on the way we managed our interactions with our industry partners in these three projects, our observations, and the decisions that we believe contributed to the collaborations being more effective. We discuss a number of important lessons learned that emerged from our collective experience and illustrate these lessons with concrete examples from the three projects.

All three projects use Model-Driven Engineering (MDE) technologies [8]. This adds yet another dimension of complexity: Despite the increasing momentum of MDE, conducting MDE research in an industrial context remains hard, mainly due to the difficulty of securing adequate buy-in from the partner companies and allaying their concerns about the risks of using MDE. One of the main concerns raised is the need for an upfront investment in learning and tailoring of MDE solutions before these solutions start paying off and bringing about cost savings and quality improvements.

Several of the lessons learned that we discuss in the paper are targeted at mitigating these concerns in MDE projects.

Our focus on MDE also makes our work a useful complement to recent initiatives by other researchers, most notably by Mohagheghi and Dehlen [14] and Hutchinson et al. [10, 11], to investigate the success and failure factors for MDE in industrial settings and the perceptions of practitioners about MDE. Our work, however, differs from these initiatives in two ways: First, our goal is to provide insights about how to engage industry in *MDE research*, rather than applying MDE per se. The second difference is the source of information on which we draw our conclusions. Whereas Hutchinson et al. use surveys and interviews, and Mohagheghi and Dehlen use a literature review as their primary means for data collection, we rely on the experience gained through direct engagement with industry in research and development activities.

The rest of the paper is structured as follows: Section 2 introduces the overarching project (ModelME!) within which the three (sub)projects that we focus on in this paper were conducted. The section continues with a brief summary of each of the three projects. Section 3 describes the way we organized our industry collaborations. Section 4 discusses and illustrates the lessons learned from the three projects, organized according to the steps in our collaboration model (Section 3). Section 5 concludes the paper with a summary and highlights important observations.

2 Context: The ModelME! Project

The three projects that this paper is based on were conducted as part of a larger project, called ModelME! (`http://modelme.simula.no`). ModelME! stands for *Model*-Driven Software Engineering for the *M*aritime and *E*nergy Sectors. The main sponsor of the project is Det Norske Veritas (DNV) – a risk management and assessment company, delivering standards, recommended practices, and certification and advisory services for many industry sectors including the Maritime and Energy (M&E) sectors. Broadly, the objective of ModelME! is to identify, tailor, and improve software engineering best practices for software-intensive systems in M&E.

In the remainder of this section, we briefly describe three (sub)projects within ModelME! that have reached maturity and have been validated in realistic settings. These three projects are the source for the lessons learned discussed in Section 4.

2.1 Traceability and Slicing (TS)

This project concerns the problem of requirements to design traceability and slicing of design models to improve design inspections during software safety certification. The industrial partner in the project was a large supplier of programmable marine electronics worldwide. During our preliminary discussions, the engineers in the partner company noted some issues related to the preparation and assessment of software safety certification documents. To better understand the current software safety certification practices, we began our work with an observational investigation. Specifically:

- We attended a number of certification meetings between the company's engineers and a certification body.
- We familiarized ourselves with the design of the systems at the collaborating unit of the partner company. We then chose, as a case study, a medium-sized safety-critical software module that was planned to undergo certification in the subsequent year.

Our observation of the meetings suggested that many issues identified during design inspections in the certification process arise due to a combination of the following:

- **Traceability.** The certifier demanded traceability links at different levels: Between requirements and their sources, between system-level and software-level requirements, between requirements and environmental assumptions, and so on. These links were either missing, or hard to find in the requirements, design, and architectural specifications.
- **Structuring of the specifications.** Requirements decomposition was not explicit; the software workflows were difficult to follow; and interaction interfaces were not clearly delineated. The main cause of these problems was the specifications being text-based (as opposed to being model-based).

Following the above observations, we set our research goals to be: providing a traceability information model characterizing the traceability links required for safety certification, a model-based methodology for system design and establishing traceability according to our traceability information model, and a mechanism for extracting minimized and relevant slices of the design for a given safety requirement. We grounded our work on the Systems Modeling Language (SysML) [20]. To date, in the course of this project, we have been able to:

- Develop a tool-supported traceability framework. Our tool automatically extracts the safety-related slices of SysML design models [6]. The main enabler for our slicing technique is the traceability between the safety requirements and the design, established by following our proposed methodology [15].
- Apply our framework to the software module that we had selected for a case study and develop guidelines tailored to the partner company for using our framework [19].
- Perform a controlled experiment with student subjects to validate the effectiveness of our framework [4].

The work was well-received by the industry partner. They plan to use the developed guidelines in the design of future modules. The models built for the module in our case study are planned to be used in the upcoming certification of that module.

2.2 Configuration and Derivation of Subsea Control Systems (CDSCS)

Our second project has been carried out in collaboration with another large system supplier company in M&E, particularly known for their Subsea Control Systems (SCSs). These systems are a

class of integrated control systems used for exploiting new energy reserves and for managing and improving oil and gas production from the existing fields. The systems may include thousands of electrical and electronic devices controlled by several highly-configurable software components.

The primary goal of this project is providing support for configuration and derivation of the software components in SCSs. SCSs can be configured differently for different environments, different oil and gas field layouts, and different kinds of hardware devices built by different vendors. Due to these complexities and inadequate automation support, configuration and derivation of the final software products in SCSs is an error-prone and costly task.

Our solution to the above problem is a UML-based product-line modeling methodology, named SimPL [2], for creating architecture models of SCS families. SimPL provides a notation and guidelines for modeling commonalities and variabilities in SCS families, along with semi-automated algorithms for SCS configuration and product derivation. The design of SimPL was driven by the technical requirements identified through close collaboration with our industry partner. SimPL uses built-in UML features for modeling commonalities and variabilities, grouping them, and relating them to elements in software and hardware models (also specified using UML). Using UML has allowed us to extensively reuse existing UML model analysis technologies, e.g., UML validation and transformation tools.

We have evaluated SimPL on a case study from the partner company identified at the beginning of the project [2, 3]. We are currently working on developing a tool to support our approach. The engineers at the partner company plan to integrate our methodology and tool within their current configuration process.

2.3 Technology Qualification (TQ)

The third project concerns the assessment of new technologies. New technologies usually include innovative aspects that are not addressed by existing standards and cannot be certified in the sense that more conventional safety-critical systems are. To demonstrate the safety and reliability of new technologies, these technologies are often subject to a specific kind of assessment, which in many industries is known as Technology Qualification (TQ). The TQ project was conducted in collaboration with DNV, which engages in various qualification projects, with a focus on M&E, particularly offshore platforms and subsea systems.

As with the other two projects discussed earlier, our first step was developing a better understanding of the needs and priorities of our industry partner. We began our work with a number of meetings and semi-structured interviews from the experts. The following observations were made about the current qualification practice:

- **Capturing of Rationale.** The qualification process must establish a link between the following: (1) the safety and reliability goals of a technology, (2) the identified risks, and (3) the collected evidence. Currently, a compliance matrix is used to establish these links, but this approach is limited in that it does not capture the rationale as to why different elements are linked.

- **Handling of Expert Judgment.** The process taken to elicit expert judgment is not always explicit, with a potential negative impact on the transparency of the qualification process.
- **Qualification costs.** Evidence collection accounts for the majority of TQ costs. Time and budget overruns can occur if effort is not focused on building and improving the right evidence information.

The technical solution we developed in response to the above needs was a model-based assessment framework that combines goal models [23], expert probability elicitation [1], and Monte Carlo simulation [17]. Within this project, we have been able to achieve the following so far:

- Developing a tool-supported framework for probabilistic assessment [18, 7]. To facilitate the adoption of the framework, we have further created a handbook for internal use by DNV providing practical guidelines on how to use the research results and the tool.
- Applying our solution to two industrial case studies concerning the technology qualification of off-shore technologies. The first case study has been described in [18].

The approach is currently being validated and refined internally at DNV on other projects. An annex based on the results of the project is being considered for DNV's technology qualification recommended practice [16, 22].

3 A Collaboration Model for Research-Based Innovation

To collaborate more efficiently with our partners, and further to ensure that we can record the insights gained from the collaborations in a useful way for future projects, we apply a consistent collaboration model for research-based innovation across all the projects in our research group. One of the first tasks we perform when engaging with a new industry partner (or a new group within a partner company we are already working with) is to discuss this model. We have found this to be useful in two ways, first to put in place a progression path for the project, and second, to clarify what we need from the industry partner at each stage, and what outcomes they can expect from the collaboration as the project progresses.

Our collaboration model, shown in Figure 1, builds on and refines the collaboration model proposed by Gorschek et al. [9]. Below, we first describe our collaboration model and then discuss how it modifies Gorschek et al's. In Section 4, we will organize the lessons learned according to the steps of our collaboration model.

The first step, *Problem Identification*, aims to identify and obtain an initial understanding of the problem that the partner company wishes to address. This is often done through meetings and organizing workshops where the industry experts at the partner company give presentations about their perceived challenges. In the second step, *Problem Formulation*, the identified problem is

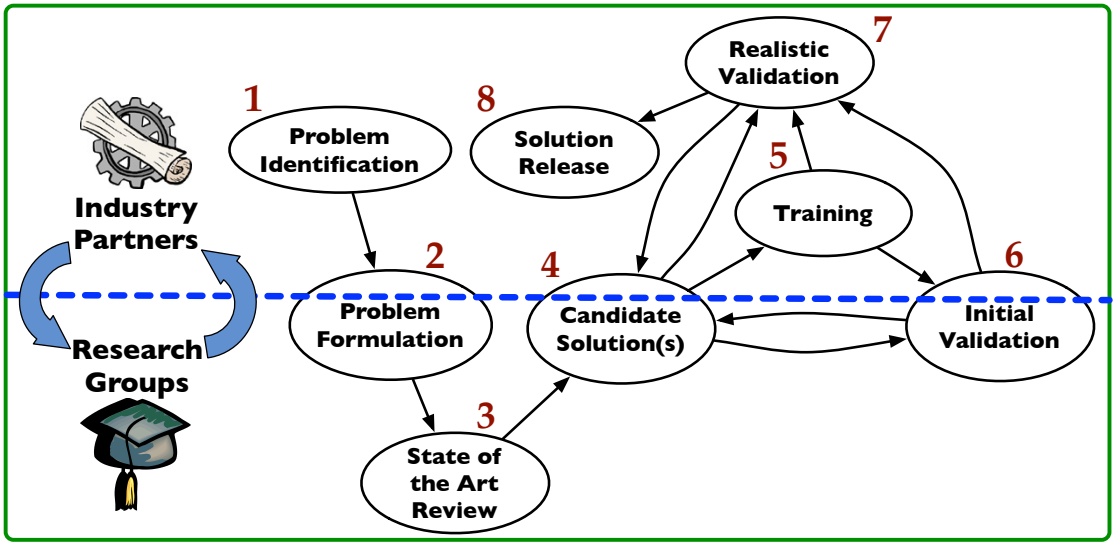

Figure 1: An adaptation of Gorschek et al's collaboration model [9] for research-based innovation

formulated in a more precise manner, and the context factors and working assumptions are clearly specified. If the problem is large and multi-faceted, it may further need to be decomposed into sub-problems that can be prioritized and tackled independently.

In the third step, *State-of-the-Art Review*, a critical review of the research literature and existing commercial or open source technologies takes place in order to identify to what extent the goals are already addressed and what are the open issues to deal with through research. In other words, the research will benefit from both the current practice and the published literature.

In the fourth step, *Candidate Solutions*, one or more potential solutions are devised. These solution(s) will be iteratively refined based on the subsequent evaluation steps (steps 6 and 7).

The fifth step, *Training*, is an incremental step. In the early stages, training focuses on building up the background necessary for the practitioners to form an (early) judgment about the feasibility of the solutions. Particularly, early training should cover the modeling constructs that the solutions expect as input. While not a definitive feasibility assessment, this allows practitioners to determine if the constructs are "natural" and "easy to build" in realistic settings given the resources they have available. In later stages and as the solutions mature, training shifts towards practical guidelines and detailed methodological steps for applying the solutions. In addition, training is a nice way to discuss with the industry partners the value of MDE in general.

In the sixth step, *Initial Validation*, we conduct a preliminary evaluation of the solutions, either in an artificial or an industry setting. In an industrial setting, we use a mix of seminars, hands-on workshops, and surveys for initial validation. If validation in an artificial setting is possible, we may use controlled studies, e.g., controlled experiments.

If the results of initial validation are promising, we move up the evaluation ladder to *Realistic Validation*. In this step, we run case studies in industrial settings, starting with pilot studies first and then spreading to wider use. The details of the proposed solutions will be refined, in particular

by providing practical guidelines, and tool support will be developed. During the pilot studies, only a small group of stakeholders will be engaged, and experiences and viewpoints on practices and tools will be collected through interviews and questionnaires. A typical result from pilot studies is that the practices are better streamlined to reduce the overhead associated with learning and using these practices. Subsequently, the streamlined practices and tools are rolled out to a wider group, data collection is performed on these projects to further assess and refine, on a wider scale, the proposed technologies.

Note that our collaboration model allows one to bypass Initial Validation and move directly to Realistic Validation. This becomes necessary when a solution cannot produce any meaningful results outside a realistic setting (e.g., when expert judgment is required).

In the eighth (and final) step, *Solution Release*, the refined tools and training material are released to the industry partner for broader application according to the exploitation plans of the industry partner.

As we stated earlier, our collaboration model is an adaptation of the collaboration model proposed by Gorschek et al. Our model offers two refinements over Gorschek et al's:

- We propose an explicit step for training, which starts long before the validation steps. A major obstacle in conducting industrial MDE research is the perception that one has to learn languages like UML in their entirety, before being able to benefit from MDE. This perception often leads to the conclusion that the learning curve associated with MDE is too steep. In reality, practitioners have to learn only what they really need from a language like UML, which typically constitutes a small fragment of the language. This fragment is determined by the modeling methodology, which specifies what part of the notation is used for a given objective and how the variation points in the semantics are specialized to address the objectives. A key goal of training should then be to minimize the learning curve by narrowing the training material to the language fragment that the practitioners actually need.
- We distinguish only between two validation steps, initial and realistic; whereas Gorschek et al. distinguish three: validation in academia, static validation (early stage industrial validation) and dynamic validation (late stage industrial validation). Dynamic validation corresponds to realistic validation in our model; but our model combines validation in academia and static validation into one, namely initial validation. This is because we find the distinction between validation in academia and static validation blurry. A pure validation in academia requires good benchmarks which are in general scarce for MDE. The chances of finding suitable benchmarks decreases even further noting that industry-driven research problems are defined in light of the context factors and working assumptions of the industry partner. As a result, even when benchmarks are used, they need to be tweaked and aligned with the partners' needs first. Subsequently, all validation activities at all steps are informed by real world considerations. Further, as we said earlier, we allow for bypassing initial validation when the solution inherently requires a realistic setting.

Figure 2 shows the distribution of project effort over the steps of our collaboration model for the three projects that we described in Section 2. The horizontal axis in the figure represents the steps

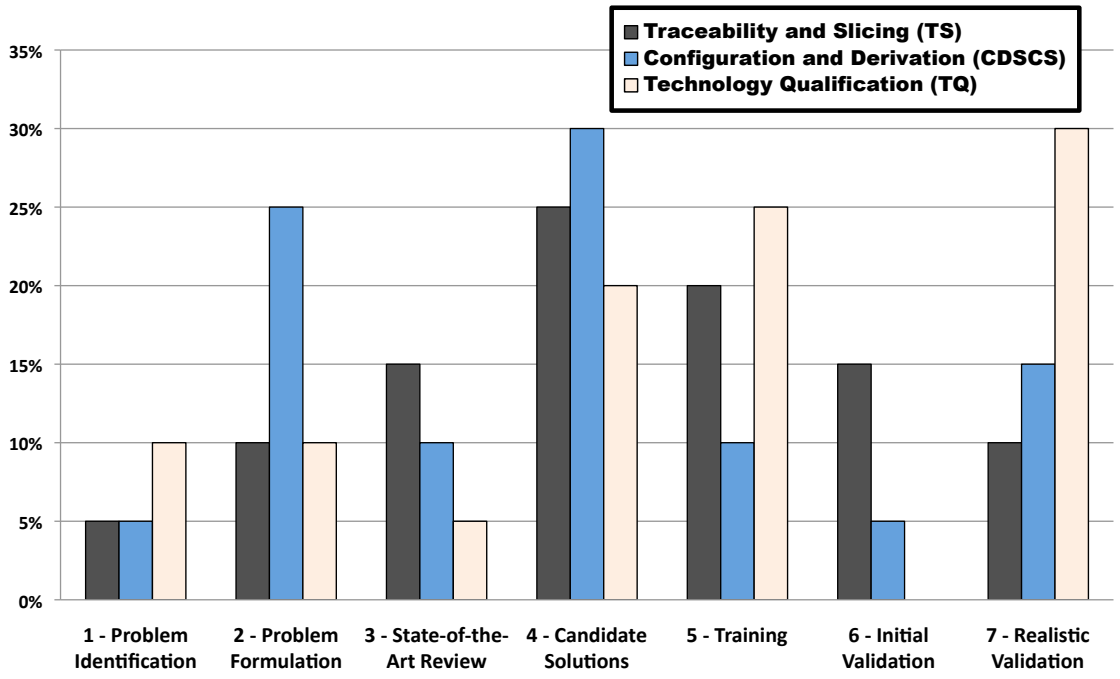

Figure 2: Approximate effort distribution for the three projects

of the collaboration model, and the vertical axis represents the percentage of overall project effort spent in a given step.

We note that the percentages in Figure 2 are estimations and not based on rigorously collected effort data. Keeping track of real effort was not possible due to the lack of control on how much work the industry partners did related to the collaboration outside the meetings and workshops we had with them.

As can be seen from the figure, the relative effort spent on Problem Identification, State-of-the-Art Review, and Candidate Solutions are comparable across projects; whereas, there are discrepancies across projects between the relative effort spent on Problem Formulation, Training, and Validation (both initial and realistic).

The Problem Formulation step in CDSCS used more of the project effort compared to TS and TQ, because in this project, multiple concrete software products had to be examined in order to develop a complete picture about the nature of variability between the products. The training step in CDSCS instead took comparatively less effort than in the other two projects. This was mainly due to the larger participation of the CDSCS partner in formulating the problem and defining a solution, and thus receiving more exposure to modeling concepts before detailed training. As for Initial Validation, the TS project required a larger percentage of effort, because the project involved a benchmark case study and was validated in an artificial setting first. Finally, the TQ project used a larger percentage of the project effort over Realistic Validation than the other two projects. This was because none of the aspects of the solution developed for TQ could be validated outside a realistic setting. Subsequently, the evaluation for TQ was concentrated entirely in Realistic Validation. Note that this also had implications on the level of training required in the TQ project. Since

there was no initial validation, the experts involved in the TQ project had had less exposure to the solution ahead of realistic validation than the experts in the TS and CDSCS projects. This gap in solution familiarity had to be compensated for via training.

Our experience from these projects indicates that an important success factor in industry research is having a precise and concrete problem formulation. That is, a reasonable amount of effort should be spent during the initial steps of a project to adequately specify the problem and capture its context factors and working assumptions. Investing effort early on for problem formulation not only improves the credibility and validity of the final solution, but may also also reduce the training effort (as was the case in CDSCS).

4 Lessons Learned

In this section, we describe the lessons learned from the three MDE projects outlined in Section 2. Note that several of these lessons are quite general and not necessarily limited to MDE research per se. However, since the lessons came out of MDE projects, the usefulness of the lessons and whether they convey the right priorities in other types of research projects has to be further investigated.

4.1 Problem Identification

LL1. The stated problem is often only a manifestation of one or more fundamental problems.

This observation bears witness more than anything else to the importance of conducting observational studies at the early stages of a project.

In the TS project (Section 2.1), the problem initially stated by the engineers was to extend and adapt the failure-mode and effect analysis (FMEA) techniques [5] that they had been using for hardware design. After attending certification meetings with third-party certifiers and collecting data on the issues pointed out by the certifier, we concluded that the majority of the (software-related) issues that the certifiers raised were due to the lack of traceability from safety requirements to the design.

In the CDSCS project (Section 2.2), the initial discussions seemed to suggest that the most pressing issues arose when the company was integrating multiple components provided by third-party suppliers. The data collected from the systematic domain analysis that followed these initial discussions provided us with more precise insights and showed that a major fraction of the problems that arise during integration stem from improper configuration of deployed software components used for monitoring and controlling of subsea devices.

In the TQ project (Section 2.3), the issue initially raised was the need to increase the transparency and cost-effectiveness of the technology qualification process. This was a very high-level objective to take action on and needed to be decomposed and clarified in multiple iterations through interviews, observation of meetings between the technology qualification team and technology suppliers, reviewing the current technology qualification best practices, and analyzing the documentation

of some existing technology qualification projects. The refined research needs that came out from analyzing the high-level objective were discussed earlier in Section 2.3.

4.2 Problem Formulation

LL2. Build a domain model as early as possible.

A domain model is a useful tool to structure the detailed discussions about a problem with an industry partner, to uncover the tacit knowledge of experts about their domain, and to avoid ambiguity and misunderstandings over terminology. Industry partners often see immediate value in domain modeling: with relatively small effort, they get a reusable "mind map" of concepts they have to deal with on a regular basis. Domain modeling is highly interactive and uses intuitive notations like UML class diagrams (or SysML block diagrams if SysML is used). Both factors contribute to leaving a good "first impression" about MDE.

In the TS project, domain modeling was performed using SysML and through interacting with the lead engineer of the IO software modules at the partner company. The activity spanned two days, involving approximately 10 person-hours of effort. The resulting SysML (context) diagram was used during problem formulation for determining the modules that were subject to safety certification, and subsequently, for identifying the links that these modules had to other modules in the system.

In the CDSCS project, a domain model was constructed during problem formulation for one family of the company's subsea systems. The domain model took about 60-80 hours to create and served as a basis for identifying, specifying, and classifying the concepts relevant to configuration of subsea systems.

In the TQ project, given that the project did not concern the design and construction of any particular system, and instead focused on assessment aspects, we did not initially see the value of building a domain model and were not planning to do so. This position changed after we started building a safety goal decomposition tree for a real system. Early on, the experts realized that they needed a unified glossary to distinguish assessment concepts such as goal, requirement, evidence, critical parameter, operating condition, safety margin, etc. As the project progressed, it was realized that a glossary alone, while very useful, was not sufficient, because the relationships between abstract and concrete terms, and the associations between different concepts could not be easily specified. For example, we needed to distinguish various types of safety evidence, e.g., testing results, analytical models, historical data. This can be much better done using a class diagram than a glossary. In the light of these observations, we are now extending our current assessment methodology [18] to accommodate an explicit step for domain modeling.

4.3 State of the Art Review

LL3. Carefully consider the context factors and working assumptions.

Context factors and working assumptions constrain what can be a feasible solution to a problem. These factors and assumptions are crucial in determining whether an existing solution can be adopted from the literature or a new solution needs to be developed through research.

In the TS project, there were three main factors that needed to be considered with regards to feasibility: First, the system safety requirements were expressed as natural language statements and there was little flexibility in using more rigorous requirements specification approaches (e.g., formal methods). Second, there was a technical constraint that a standardized notation such as UML or SysML should be used for the design to minimize ambiguity and communication overhead when the licensing or regulatory body is performing safety certification. Lastly, the goal pursued from traceability was very specific, namely compliance to the IEC61508 standard. We did not find any existing solutions in the literature that satisfied this particular combination.

Likewise, in the CDSCS project, the survey that we conducted of the existing variability modeling languages did not identify any specific solution that could be directly used in our context. The main contextual factors constraining the solution were as follows: (1) the variability modeling languages we identified and investigated during the systematic domain analysis did not have enough expressive power to capture all the variabilities seen in the systems developed by our partner company; (2) the target was control systems with *both* hardware and software, and not just the latter. Therefore we needed to capture hardware and its relation to software in the product line architecture model. None of the approaches we investigated address this need; (3) we needed to derive deployable and executable software, and this requirement is only partially fulfilled by the approaches in the literature.

In the TQ project, our literature review identified good solutions for the sub-problems of the original problem. The main challenge that had not been investigated before was integrating the solution components into a complete and seamless solution. This was complicated by the fact that the solution components were multidisciplinary, and combining them required background from different fields.

In all three projects, the industry partners expected *end-to-end* solutions that were not only technically sound but could further satisfy their criteria about cost, training, integration with existing development environments, and organization culture. Failure to account for these criteria in the solutions would inevitably lead to the rejection of the solutions.

4.4 Candidate Solutions

LL4. If practitioners cannot conveniently provide the input required by a solution, the solution is unlikely to be adopted.

A major consideration while defining solution(s) to a problem is the simplicity and naturality of the input that the solutions expect. Often, MDE solutions are accused of requiring input models that are too complex to build for the engineers, or that are based on information that cannot be realistically obtained at the targeted stage of development in the partner's organization. This problem is not due

to an inherent limitation in MDE, but rather due to the solutions not fully considering the expertise, culture, and processes used at the partner company (also see *LL3*).

LL5. Rely as much as possible on standardized modeling languages.

Reliance on standardized modeling languages and their extensions increases buy-in from the industry. This is because using standards largely avoids "lock in". Before a company invests into MDE technologies, they need to ensure that the technologies are going to be supported for a long time. Proprietary modeling languages are usually viewed as posing a risk, because there is less certainty as to how long they may be supported. Using standardized modeling languages is further advantageous for tool building, because solutions can be built on top of the existing commercial or open-source environments that support these languages.

In the TS project, we based our work on SysML due to the rising popularity of SysML in systems engineering. Our traceability methodology was accompanied by detailed guidelines as to what needed to be specified by the engineers. To maintain simplicity, we use only a subset of SysML that is essential for capturing the structure and behaviour of the IO modules at the partner company. We held four modeling workshops with the lead engineer of these modules, aimed at ensuring that our requirements for the input models are reasonable. Thanks to the support for SysML in the existing modeling platforms, we could develop a tool for our solution with relatively little effort. Specifically, our tool is implemented as a plugin for the Enterprise Architect modeling environment (`http://www.sparxsystems.com/`).

In the CDSCS project, we were keen on addressing, in the design of our methodology, the practical considerations of our industrial partner, in particular, training costs and availability of the modeling tools. Hence, we chose to rely on UML and its extension for Modeling and Analysis of Real-Time and Embedded systems (MARTE) [13]. The methodology was designed after conducting interviews with the engineers at the partner company and eliciting detailed information about their systems to make sure that our methodology matches the needs of these systems. Our methodology is captured through a profile that extends UML/MARTE for the purpose of configuration of subsea control systems. The profile has been implemented on top of the Rational Software Architect modeling environment (`http://www.ibm.com/developerworks/rational/products/rsa/`).

In the TQ project, the main decision about the input language concerned the specification and decomposition of safety and reliability goals. Before adopting goal modeling as the basis for our work, we made sure that key goal modeling concepts such as "goals" and "obstacles" were natural for the experts at the partner company. We found out that the experts indeed reasoned in terms of goals and barriers to goal satisfaction although they did not always make this reasoning explicit. Among the existing goal modeling languages, we chose KAOS [23] for two main reasons: (1) the existence of an extended set of modeling guidelines in a textbook [23] which could be used for training; and (2) amenability of KAOS to quantitative assessment. This made KAOS a great fit for the existing technology qualification practice which is based primarily on probabilistic assessment. To provide an industrially usable tool, we implemented the KAOS notation as a UML profile for the Enterprise Architect environment, rather than developing a tool from scratch.

4.5 Training

LL6. Do training only incrementally and based on needs.

Training has to be incremental and tailored to the specific needs of the industry partner. On the one hand, engineers have little slack time and cannot be expected to attend extensive training. On the other hand, the engineers will not be able to apply the proposed solutions unless they have received adequate training.

LL7. For training, use examples from the domain being studied, not examples from textbooks or other domains.

No matter how complete and concrete the examples used for training are, if the examples are not related to the industry partner's domain, they will seldom be convincing enough. An example in an industry training course is not merely to convey an idea but also a critical means to "demonstrate" that the idea applies to the domain of interest. Naturally, examples drawn from a particular domain are also easier to remember and relate to for experts in that domain. Using such examples also increases the overall effectiveness of training by creating a greater incentive for engineers to actively participate in the training sessions.

In the TS project, training was integrated with the modeling review sessions that we conducted with the engineers. The goal of these sessions was twofold: First, to validate and refine our design models for the IO modules at the company, and second, to help engineers modify and build these models for other similar modules. These sessions were run interactively where the engineers commented on the models of the case study module and attempted to change the models for other IO modules. After finishing the modeling review sessions, we provided the engineers with a technical report that included step-by-step guidelines for creating design diagrams and traceability links for their IO modules.

In the CDSCS project, we provided a number of formal modeling tutorial sessions. In the tutorials, we focused only on those UML diagrams that were required most for understanding of our methodology. To teach these diagrams, we extensively used illustrative examples from the models that we had built for a family of subsea systems at the company. The tutorials were interactive and the attendees were provided with booklets containing modeling guidelines and examples. Finally, they were asked to perform a number of modeling exercises at the end of the tutorial.

In the TQ project, the training was carried out in a number of modeling workshops. We started by giving an introduction to KAOS based on its reference book [23]. For exemplification, we used real technology qualification examples. Our strategy was to first analyze some of the case study documents, construct partial goal models based on our understanding, and then explain and exemplify the syntax and semantics of the KAOS language over the concepts in the case study. The second half of each workshop was hands-on training where we interactively improved the goal models with the experts and finalized them. This was done by having the experts perform some KAOS modeling tasks on their own. We observed that the experts were able to grasp the underlying semantics of KAOS and become increasingly self-sufficient in goal modeling over time.

4.6 Initial Validation

LL8. Validation in an artificial setting may be of limited value or not possible.

In the absence of good benchmarks, validation in an artificial setting may have limited value, because there may be too wide a gap in terms of assumptions and the level of complexity between an artificial case study and a real case study. For some problems, an artificial setting may not even be a possibility, e.g., when expert judgment is involved.

LL9. Take particular note of scalability considerations during initial validation.

Being able to cope with the large scale of real systems is an important consideration for the industry. Unfortunately, scalability often comes at the cost of lowering precision, which in turn reduces the conclusiveness of the analyses performed. During initial validation, it is important to discuss with the industry partner how a proposed solution will "degrade" in the face of reduced precision in the input models. A solution is less likely to be adopted if the degradation is too fast or is just binary (i.e., there is a precision point above which analysis is fully conclusive and below which analysis is fully inconclusive).

In the TS project, our initial validation involved applying our solution to the Production Cell System (PCS) [12], which is a well-known benchmark for reactive systems. This benchmark has been used before to evaluate the capabilities of various specification methods for the purpose of safety analysis and verification [12]. We constructed a complete set of SysML models for PCS during our validation. The PCS specification further includes a list of safety requirements which we used for establishing traceability between requirement and SysML design models. Throughout our benchmark study, we were also interacting with our industry partner to learn about the design of their systems. This interaction influenced the way the design models in the benchmark were built. A high priority for the partner was to keep the design specification and traceability effort low, while satisfying all the criteria for compliance with the relevant safety standards. The partner was flexible to accept a reasonable increase in the amount of modeling effort if this meant the resulting models would be reusable, or made it possible to exploit the models for purposes other than certification, e.g., staff training and automated report generation.

In the CDSCS project, no artificial cases studies were performed because there was no benchmark or exemplar that was representative enough for SCSs. In this project, we were given access to a real system by the industry partner at the beginning of the project, and focused our activities as early as possible on the industrial case study. Initial validation was done through seminars in which we presented to the industry partner our solution at different stages of progress and obtained feedback. Scalability to systems with thousands of sensors and actuators was mentioned by the partner as a key requirement for the solution. This requirement had a direct influence on the level of abstraction at which architectural models are built and organized in our solution.

In the TQ project, no initial validation was performed. As stated earlier, our solution in TQ involves expert probability elicitation and the only plausible way to evaluate the solution was to apply it to a real (albeit small) case, and tune the solution in response to the observations made in a realistic setting. Of course, we still had to consider and decide about the trade-offs between cost and the

granularity of goal decomposition and expert elicitation. But such decisions were made over the course of realistic validation.

4.7 Realistic Validation

LL10. Choose your pilot studies wisely!

New solutions are rarely put into production immediately and are always tested out on pilot projects first. To increase the chances of a solution getting adopted, one must take the following factors into account when selecting pilot studies:

- Pilot studies should be representative in the eyes of the industry partners, so that they can believe the results. In other words, the pilot studies should adequately reflect the characteristics of the broader range of systems that the solutions are targeted at.
- Pilot studies should be complex enough for validation and yet be commensurate with the available resources. Results from simplistic pilot studies may be unconvincing or even misleading. On the flip side, if the pilot studies require resources beyond what the research team can secure from the industry partner, the studies will not succeed.
- When feasible, pilot studies should be defined over *ongoing* activities at the partner company, as opposed to over past activities. From a practical standpoint, basing a pilot study on ongoing work is beneficial in two ways: First, the effort for the pilot study will be usually overlapping with the work that the staff at the partner company have to do anyways to deliver their products and services. Due to this overlap, the pilot study will no longer be viewed as a "side activity" and the staff will be able to spend more resources on the pilot study and justify this effort. Second, if the pilot study contributes to improving the current activities, the solution will have an immediate and tangible impact on the company, thus increasing buy-in.

If a reenactment of past activities is chosen as the basis for a pilot study, the research team has to make sure that the company has a horizon for using the results of the pilot in the future; otherwise, it becomes difficult to stimulate enough interest from the staff at the partner company to actively participate in the pilot.

LL11. Be ready to provide substantial help in model construction during realistic validation.

Mentorship is a critical element for success in research-based innovation, particularly in the context of MDE. Specially, if the industry collaborators do not have a long history of using MDE, the research team has to set a good example during early case studies, which the collaborators can learn from, refer to, and reuse in the future. We believe that mentorship is best done through the deep involvement of the research team in the construction of the case study models. While time consuming, "getting one's hands dirty" with model construction is also an excellent way for the researchers to understand the modeling needs, and further to show to the partner company that the researchers are genuinely interested and committed to addressing the partner company's problems, in turn helping with building trust (see *LL12*). Once the mentorship process is complete,

researchers' assistance in model construction should be phased out to avoid the validity threats we may face in our empirical studies due to actively helping in creating models.

In the TS project, it was very important to the industry partner that the IO module chosen for realistic validation should be representative. The IO modules developed by the company use a complex multithreading structure to increase performance and to allow for more configurability. We were told early on that, unless the models built to specify the multithreaded behavior of the modules were representative of all the modules, the development team would be unable to apply our solution, because the multithreading models were too expensive to build for the modules individually. The module that we selected for our case study was deemed as having all the multithreading features that the broader set of modules use. However, to get a grip on the complexity of the case study, we had to compromise on the communication protocols that the module could work with: we only considered the simplest communication protocols in our case study. The researchers led the effort on model construction for the study. The work was aligned with the current needs of the industry partner as the models were being prepared for the next round of safety certification.

In the CDSCS project, representativeness was discussed at length with the industry partner. Here, representativeness refers to covering as many variability types in the product family as possible. To ensure representativeness, three different products were chosen for our case study. These three products were deemed by the industry partner as collectively covering the significant majority of the variability types seen across their products. The chosen products were all completed recently. The relevant knowledge was still fresh in the minds of the engineers, and the products were planned to be used as a basis for future products. Again, as in the TS project, we needed to manage the complexity of realistic validation. Instead of building a product-line architecture model that exhaustively captures all the commonalities and variabilities in the products, we created a model that contains instances of all variability types. The modeling effort was led by the researchers with participation from both management and development staff at the partner company.

The TQ project differed from the TS and CDSCS projects in that in was done in collaboration with an assessment body rather than a system supplier. The body qualifies a diverse set of technologies ranging from mechanical equipment and chemical processes to software-controlled systems. Due to this diversity, it was infeasible to define representativeness in an actionable manner. The selection process for our pilot studies was opportunistic with the following constraints: (1) the pilot studies must not be too time-consuming for the experts and should preferably be on current/recent qualification projects, (2) experts in the domain areas of the pilot studies have to be available throughout the studies for goal model construction and expert elicitation. The modeling effort in our first pilot study was led by the researchers. In the second pilot study, the models were constructed by the experts with some help from the researchers.

4.8 Solution Release

LL12. Find internal champions for the solution.

For most industry research problems, the researchers get to collaborate with the technical staff at the partner companies, but the final decision about whether to adopt a proposed solution is made by

the management team at the partner company who may not have been involved in the collaboration. To be able to carry a new solution through to broad use, the technical staff at the partner company must champion the solution. In other words, they have to make a convincing case to the management about the solution's benefits. For this purpose, the technical staff often have to provide compelling business cases where the solution leads to cost savings or quality improvements, and further to propose a strategy for integrating the solution into the current development workflows at the company. This level of commitment does not materialize unless the industry collaborators develop a strong sense of trust in the researchers and the research being conducted. Building such trust takes *years* and requires the researchers to develop a deep appreciation of the business culture at the partner company [21].

The TS project is championed by the lead engineer of the IO modules, who has also developed and presented an exploitation plan to the management after the industrial case study was concluded. The CDSCS project is championed by the quality assurance team, who are currently investigating how the developed solution can be integrated into the existing tool chain at the company. For the TQ project, management was involved in the technical work of the project, giving us the opportunity to continuously synchronize our solution and tool support with the required business cases at the partner company.

5 Conclusion

This paper reports on experiences we have had with three industry partners in performing what we call research-based innovation. The fundamental position of this research paradigm is that, in software engineering as in other engineering disciplines, research and industrial innovation can be beneficially intertwined in order to ensure that the problems addressed by researchers are well-defined and relevant. We must also strive to account for all important context factors in devising novel solutions to software engineering problems and this requires close interactions with industry partners.

This research paradigm is also an effective way to transfer novel technologies to practitioners as they are involved early on in the development of the solutions, thus creating many opportunities for mentoring and a sense of ownership. Another way to describe this research paradigm is to highlight its *inductive* nature, that is the fact that we work from specific observations in concrete settings but attempt to devise general solutions with clear working assumptions.

The lessons learned we report in this paper are focused on applications of Model-Driven Engineering. They are structured according to the various phases of our research-based innovation model (Figure 1), starting with Problem Identification and ending with Solution Release. The ultimate goal of structuring and sharing these experiences is to help future researchers and practitioners better cooperate and ensure the success of their collaboration endeavors. Some of these lessons learned focus on how to thoroughly understand the problem and context before working on a solution. In software engineering, characteristics of the system, organization, and human factors can have a strong influence on whether a solution is applicable and scalable.

We also discuss other factors that influence the development of a solution such as the need to account for modeling standards and assessing the feasibility of the input to be provided by future users. How to perform training is also addressed as being a key component of the paradigm. Industry practitioners usually have little time to devote to professional education and this is usually a significant obstacle to change. Last, we addressed the validation of the proposed solutions, both at an early stage and then later on in realistic project settings. A two-stage validation, though not always possible, is a way to alleviate the risks associated with novel solutions.

From a more general standpoint, this paper discusses ways to bridge the existing gap between software engineering research and practice, an issue we believe to be of crucial importance for the future of our profession.

Acknowledgments. We thank Vic Basili, Richard Torkar and Jose Luis Gonzalez for their useful comments on an earlier draft of this paper.

References

[1] A. O'Hagan et al. *Uncertain Judgements: Eliciting Experts' Probabilities*. Wiley, 2006.

[2] R. Behjati, T. Yue, L. Briand, and B. Selic. Simpl: A product-line modeling methodology for families of integrated control systems. Technical Report 2011-01, SRL, 2011.

[3] R. Behjati, T. Yue, S. Nejati, L. Briand, and B. Selic. Extending sysml with aadl concepts for comprehensive system architecture modeling. In *ECMFA*, pages 236–252, 2011.

[4] L. Briand, D. Falessi, S. Nejati, M. Sabetzadeh, and T. Yue. Traceability and sysml design slices to support safety inspections: A controlled experiment. Technical Report 2010-08, SRL, 2010.

[5] C. Ericson. *Hazard Analysis Techniques for System Safety*. JOHN WILEY & SONS, 2005.

[6] D. Falessi, S. Nejati, M. Sabetzadeh, L. Briand, and A. Messina. Safeslice: a model slicing and design safety inspection tool for sysml. In *SIGSOFT FSE*, pages 460–463, 2011.

[7] D. Falessi, M. Sabetzadeh, S. D. Alesio, and L. Briand. Modus: A tool for goal-based safety and reliability assessessment of new technologies, 2011. Submitted.

[8] R. France and B. Rumpe. Model-driven development of complex software: A research roadmap. In *FOSE*, pages 37–54, 2007.

[9] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.

[10] J. Hutchinson, M. Rouncefield, and J. Whittle. Model-driven engineering practices in industry. In *ICSE*, pages 633–642, 2011.

[11] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical assessment of mde in industry. In *ICSE*, pages 471–480, 2011.

[12] C. Lewerentz and T. Lindner, editors. *Formal Development of Reactive Systems - Case Study Production Cell*, volume 891 of *LNCS*. Springer, 1995.

[13] A UML profile for MARTE: Modeling and analysis of real-time embedded systems, May 2009.

[14] P. Mohagheghi and V. Dehlen. Where is the proof? - a review of experiences from applying mde in industry. In *ECMDA-FA*, pages 432–443, 2008.

[15] S. Nejati, M. Sabetzadeh, D. Falessi, L. Briand, and T. Coq. A sysml-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies. Technical Report 2011-01, SRL, 2011.

[16] Qualification procedures for new technology. DNV-RP-A203, DNV, 2001.

[17] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer, 2005.

[18] M. Sabetzadeh, D. Falessi, L. Briand, S. D. Alesio, D. McGeorge, V. Åhjem, and J. Borg. Combining goal models, expert elicitation, and probabilistic simulation for qualification of new technology. In *HASE*, 2011. (to appear).

[19] M. Sabetzadeh, S. Nejati, L. Briand, and A. Evensen Mills. Using SysML for modeling of safety-critical software-hardware interfaces: Guidelines and industry experience. In *HASE*, 2011. (to appear).

[20] OMG Systems Modeling Language (OMG SysML). `http://www.omg.org/docs/formal/08-11-02.pdf`, 2008. Object Management Group (OMG), version 1.1.

[21] P. Tarr. So you want to marry an industrial. Presentation., 2011.

[22] Technology qualification management. DNV-OSS-401, DNV, 2010.

[23] A. van Lamsweerde. *Requirements Engineering - From System Goals to UML Models to Software Specifications*. Wiley, 2009.