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

Lionel Briand[1], Davide Falessi[2,3], Shiva Nejati[1], Mehrdad Sabetzadeh[1], and Tao Yue[2]

[1]SnT Centre, University of Luxembourg, Luxembourg
[2]Certus Software V&V Centre, Simula Research Laboratory, Norway
[3]University of Rome (Tor Vergata), Italy
{lionel.briand,shiva.nejati,mehrdad.sabetzadeh}@uni.lu
{falessi,tao}@simula.no

Abstract. In recent years, we have been exploring ways to foster a closer collaboration between software engineering research and industry both to align our research with practical needs, and to increase awareness about the importance of research for innovation. This paper outlines our experience with three research projects conducted in collaboration with the industry. We examine the way we collaborated with our industry partners and describe the decisions that contributed to the effectiveness of the collaborations. We report on the lessons learned from our experience and illustrate the 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 experience with three projects that have reached maturity, selected from a larger set of ongoing projects that we are currently conducting in collaboration with the industry. We reflect on the way we managed our interactions with our industry partners in these projects, our observations, and the decisions that we believe contributed to the collaborations being more

effective. We discuss a number of lessons learned that emerged from our collective experience and illustrate these lessons with concrete examples.

All three projects use Model-Driven Engineering (MDE) technologies [1]. 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. The lessons learned we discuss in the paper not only cover the researchers' role in research-based MDE projects but also the expectations from the industry in such projects, including upfront investment in learning and tailoring of MDE solutions and the existence of champions for the devised solutions.

Our focus on MDE makes our work a useful complement to recent initiatives by other researchers, most notably by Mohagheghi and Dehlen [2] and Hutchinson et al. [3, 4], 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 an overview 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 summarizes the paper and highlights important observations.

2 Context: The ModelME! Project

The projects discussed in this paper are part of a larger project, called ModelME! (*Model*-Driven Software Engineering for the *M*aritime and *E*nergy Sectors, `http://modelme.simula.no`). Broadly, the objective of ModelME! is to improve software engineering best practices for software-intensive systems in the Maritime and Energy (M&E) sectors. In this section, we briefly describe three (sub)projects within ModelME! that have reached maturity and have been validated in realistic settings. These 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. 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 attended a number of certification meetings between the company's engineers and a certification body. Our observation of the meetings suggested that the majority of the issues identified during design inspections in the certification process arise due to poor structuring of the specifications and missing traceability, in particular, between safety requirements and software design.

Following our observations, we set our research goals to be: providing an information model to characterize the traceability links required in design safety inspections, a model-based methodology for establishing such traceability links, and a mechanism for extracting minimized and relevant slices of the design for a given safety requirement. Grounding our work on the Systems Modeling Language (SysML) [5], we have developed a tool-supported framework for design safety inspections in the context of safety certification [6, 7], applied the framework to a number of selected software modules from our industry partner, and created guidelines tailored to the partner company for using our framework [8].

Our partner has now started using our guidelines in the design of its modules. The models built in our case study are planned to be used in the upcoming round of safety certification at the company. The project has thus far engaged three full-time researchers for six months and one full-time engineer for two months.

2.2 Configuration & Derivation of Subsea Control Systems (CDSCS)

Our second project has been carried out in collaboration with another large systems supplier in M&E, particularly known for their Subsea Control Systems (SCSs). The embedded software in SCSs has very large configuration spaces, including configuration for hardware architectures, for data communication protocols, and for the individual physical devices.

In our initial investigation, we observed that the hardware and software configuration processes at our partner company were isolated from one another, resulting in many configuration mismatches and errors that were often detected late and only after the integration of software and hardware. We therefore set the primary goal of this project to be: providing support for configuration and derivation of the software components in SCSs such that the complex dependencies between hardware and software are captured and preserved. To achieve this goal, we have developed a model-based approach for configuring the software embedded in SCSs. We use built-in UML features for modeling the architecture of SCSs and the architectural dependencies between the software and hardware elements. Our approach can capture complex software-hardware dependencies. The approach automates some of the configuration decisions and interactively guides users to make the remaining configuration decisions [9, 10].

We have evaluated our approach on a case study from the partner company [10, 11]. Our experience shows that our approach successfully enforces consistency of configurations, can automatically infer up to 50% of the configuration decisions, and reduces the complexity of making configuration decisions. We are now working on integrating our approach and tool into the current configuration process at the partner company. This project has thus far engaged two full-time researchers for one year, and one full-time engineer for three months.

2.3 Technology Qualification (TQ)

The third project concerns the assessment of new technologies. New technologies usually include novel 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 (Det Norske Veritas), 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. The following observations were made about the current TQ practice based on meetings and interviews with domain experts: (1) There is not adequate traceability between the safety and reliability goals of a new technology, the identified risks, and the evidence collected to show that the technology is fit for purpose; (2) The process taken to elicit expert judgment is not always explicit, with a potential negative impact on the transparency of the qualification process; (3) Time and budget overruns can occur if the evidence collection effort is not focused on building and improving the right evidence information.

In response to the above, we have developed a model-based approach for probabilistic assessment of new technologies [12, 13], which combines goal models [14], expert probability elicitation [15], and Monte Carlo simulation [16]. To facilitate the adoption of the approach, we have developed a prototype but highly usable tool to support it. We have further created a handbook for internal use by DNV providing practical guidelines on how to use the research results and the tool. Our solution has so far been applied in two industrial case studies both concerning off-shore technologies. The first case study is described in [12].

The approach is currently being validated and refined internally at DNV in other projects. An annex based on the results of our work is being considered for DNV's technology qualification recommended practice [17, 18]. The TQ project has thus far engaged two full-time researchers for a year, two master students for six months, and five DNV staff members for approximately four months.

3 A Collaboration Model for Research-Based Innovation

To collaborate more efficiently with our partners, and further to ensure that we record the insights gained from the collaborations for future projects, we apply a unified collaboration model across the projects in our group. One of the first tasks we perform when engaging with a new industry partner 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 in Figure 1, builds on and refines the collaboration model proposed by Gorschek et al. [19]. Below, we first describe our collaboration

model and then discuss how it modifies Gorschek et al's. In Section 4, we will present the lessons learned according to the steps of our collaboration model.

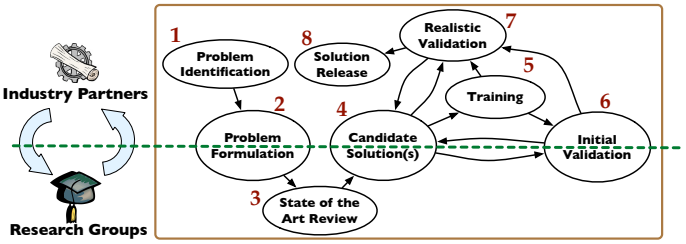

Fig. 1. Adaptation of Gorschek et al's model [19] for research-based innovation

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 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 experience 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.

We note that our collaboration model allows one to bypass Initial Validation and move directly to Realistic Validation. This flexibility is desirable for two reasons. First, it allows for more agility in the execution of a research-based project if there are constraints on timelines and/or available resources. Second, such flexibility is required when a solution cannot be meaningfully evaluated 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. Here, the research team plays a primarily supportive role, e.g., through consultancy and maintenance services. While the Solution Release step deserves careful consideration in research-based projects, the three projects upon which we draw in this paper have only been recently released, thus offering limited insight about the longer-term interactions between research and industry in this step. A longer-term investigation is required to provide a more conclusive picture of this step.

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:

(1) 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.

(2) 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 valida-

tion 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. As noted earlier, the projects are too young to provide detailed insights about the Solution Release step in our collaboration model. Therefore, we do not consider Solution Release in the project effort distribution. We further note that the percentages in Figure 2 are estimations. 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 Steps 1, 3 and 4 are comparable across projects; whereas, there are discrepancies across projects between the relative effort spent on Steps 2, 5, 6 and 7.

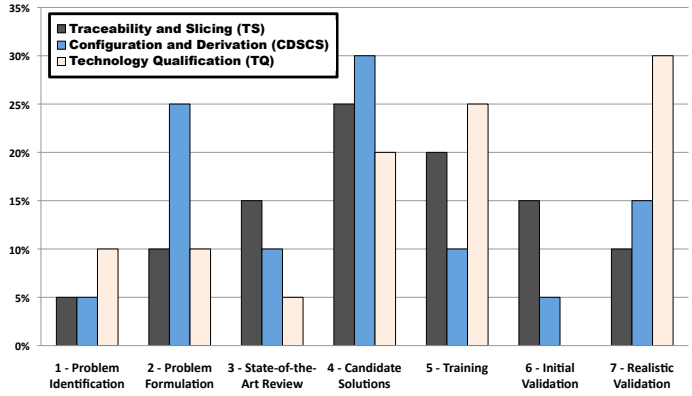

Fig. 2. Approximate effort distribution for the three projects. Horizontal axis represents the steps of the collaboration model; vertical axis represents the percentage of overall project effort in a given step.

Step 2 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 types of the configurable parameters and the architectural dependencies. 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. Therefore, the evaluation for TQ was focused entirely on 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 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 could 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. Several of these lessons are general and not limited to MDE per se. However, since they came out of MDE projects, their usefulness 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 lesson underlines the importance of observational studies in early stages of a project. In the TS project (Section 2.1), the problem initially stated by the engineers was to extend and refine their Failure-Mode and Effect Analysis (FMEA) techniques [20]. After attending software certification meetings with third-party certifiers, we observed that the majority of the issues that the certifiers raised were due to the lack of traceability from safety requirements to design.

In the CDSCS project (Section 2.2), initial discussions suggested that the most pressing issues concerned the integration of third-party components. The data collected from the systematic domain analysis that followed these initial discussions provided us with more insights and showed that a major fraction of the integration problems stem from improper configuration of software components.

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 high-level objective was refined through interviews, observation of technology qualification activities, and reviewing of some past technology qualification projects. The refined research needs that came out from analyzing the high-level objective were discussed 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 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 using UML 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 and classifying the concepts relevant to configuration of subsea systems.

In the TQ project, given that we were not concerned with the development of any particular system, and were focused on assessment, we did not initially see the value of building a domain model. Instead of building a domain model, we developed a glossary to distinguish assessment concepts such as goal, requirement, evidence, safety margin, etc. As the project progressed, it was realized that a glossary alone, while 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 model than only a glossary. In the light of this observation, we are now extending our current assessment methodology [12] 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 and are crucial in determining whether an existing solution can be adopted from the literature or a new solution needs to be developed.

The contextual factors in the TS project were: (1) The system requirements were expressed as natural language statements and there was little flexibility in using more rigorous requirements specification approaches (e.g., formal methods); (2) There was a technical constraint that a standardized notation (e.g., SysML) should be used for the design to minimize ambiguity and communication overhead when the licensing or regulatory body is performing safety certification; and (3) 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 for configuring architectural variabilities in SCSs and could further capture the hardware-software dependencies in such systems.

In the TQ project, our literature review identified good solutions for the sub-problems of the original problem. The main challenge 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 also satisfy their criteria about cost, training, integration with existing process, and organization culture.

4.4 Candidate Solutions

LL4. If practitioners cannot conveniently provide the input required by a solution, the solution is unlikely to be adopted. MDE solutions are often accused of requiring input models that are too complex for the engineers to build, or that are based on information which cannot be realistically obtained at an organization. A major consideration in defining an MDE solution is the simplicity and naturality of the models that it requires as input and making sure the input language is a suitable match for the expertise, processes, and the culture at the partner company (also see *LL3*).

LL5. Rely as much as possible on standardized modeling languages. Reliance on standardized modeling languages increases buy-in from the industry because it 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 considered risky, because there is uncertainty as to how long they may be supported. Using standards is further advantageous for tool building, because solutions can be built on top of the existing commercial or open-source environments.

The TS project was based on SysML due to its rising popularity in systems engineering. We used only a subset of SysML that was essential for capturing the design of the IO modules at the partner company. We held four modeling workshops with the lead engineer of these modules to ensure that our requirements for the input models are reasonable. Thanks to the existing modeling platforms for SysML, we could develop a tool for our solution, as a plugin for the Enterprise Architect modeling environment (`http://www.sparxsystems.com/products/ea/`), with relatively little effort.

In the CDSCS project, we base our work on UML and its extension for Modeling and Analysis of Real-Time and Embedded systems (MARTE) [21], primarily because this combination can seamlessly capture software and hardware concepts. Our methodology was designed after conducting interviews with the engineers at the partner company and eliciting detailed information about their systems to ensure that the methodology would match their needs. The methodology 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. Among the existing goal modeling languages, we chose KAOS (Knowledge Acquisition in Automated Specification) [14] for two main reasons: (1) the existence of an extended set of modeling guidelines in a textbook [14] which could be used for training; and (2) amenability of KAOS to quantitative assessment. This made KAOS a nice fit for the existing technology qualification practice which is based mainly on probabilistic assessment. To provide a 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 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: First, to validate and refine our design models for the IO modules, and second, to help engineers modify and build these models for other similar modules. During these sessions, the engineers were asked to comment on the models and to change them for other IO modules. At the end of these 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 held a number of modeling tutorials focused specifically on the UML diagrams that were essential for understanding our methodology. In the tutorials, we used illustrative examples from the models that we had built for a family of subsea control systems at the company. The tutorials were interactive and the attendees were provided with booklets containing modeling guidelines and examples. At the end of the tutorials, they were asked to perform a number of modeling exercises.

In the TQ project, training was performed in a number of modeling workshops. We gave an introduction to KAOS based on its reference book [14]. For exemplification, we developed examples from real technology qualification projects. Each workshop included a hands-on training session where we interactively built

and refined goal models with the experts. We observed that the experts became 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. Without good benchmarks, validation in an artificial setting may have limited value, because there may be too wide a gap in terms of assumptions and level of complexity between an artificial and a real case study. For some problems, an artificial setting may not even be possible, e.g., when expert judgment is involved.

LL9. Take particular note of scalability considerations during initial validation. 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) [22] – a well-known benchmark for reactive systems, which has been previously used to evaluate the capabilities of various specification methods for safety analysis [22]. We constructed a complete set of SysML models and traceability links for PCS. Throughout our benchmark study, we also interacted 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 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 for other modules, or exploitable for purposes other than certification, e.g., for staff training.

In the CDSCS project, no artificial cases studies were performed because we were unable to find representative benchmarks. In this project, we were given access to a real system by the industry partner at the beginning of the project. 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 was a key requirement for the solution and had a direct influence on the level of abstraction of the architectural models that we built, and on the design of our configuration tool [9].

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 case.

4.7 Realistic Validation

LL10. Choose your pilot studies wisely! New solutions are rarely put into use immediately and are almost always tried 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 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 more willing to spend resources on the pilot study. 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 ensure that the company has a horizon for using the results of the pilot in the future; otherwise, it may become difficult to stimulate enough interest from 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 one may face in our empirical studies due to actively helping in creating the models.

In the TS project, the IO modules under study had a complex multithreading structure. 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 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 our partner as the models were being prepared for the next round of certification.

In the CDSCS project, representativeness referred to covering as many variability types in the product family as possible. To achieve this, we chose three different products that were deemed by the industry partner as collectively covering the majority of the variability types in their systems. These products were planned to be used as a basis for their future product development. 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 purely mechanical equipment 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 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 [23].

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 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 build 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 integrating the solution within the existing process. 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 de la Vara for their useful comments on an earlier draft of this paper. L. Briand was supported by a FNR PEARL grant.*

References

1. France, R., Rumpe, B.: Model-driven development of complex software: A research roadmap. In: FOSE. (2007) 37–54
2. Mohagheghi, P., Dehlen, V.: Where is the proof? - a review of experiences from applying mde in industry. In: ECMDA-FA. (2008) 432–443
3. Hutchinson, J., Rouncefield, M., Whittle, J.: Model-driven engineering practices in industry. In: ICSE. (2011) 633–642
4. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of mde in industry. In: ICSE. (2011) 471–480
5. Object Management Group (OMG): Systems Modeling Language (SysML). `http://www.omg.org/docs/formal/08-11-02.pdf` (2008) version 1.1.
6. Nejati, S., Sabetzadeh, M., Falessi, D., Briand, L., Coq, T.: 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)
7. Falessi, D., Nejati, S., Sabetzadeh, M., Briand, L., Messina, A.: Safeslice: a model slicing and design safety inspection tool for sysml. In: SIGSOFT FSE. (2011) 460–463
8. Sabetzadeh, M., Nejati, S., Briand, L., Evensen Mills, A.: Using SysML for modeling of safety-critical software-hardware interfaces: Guidelines and industry experience. In: HASE. (2011) (to appear).
9. Behjati, R., Nejati, S., Yue, T., Gotlieb, A., Briand, L.: Model-based automated and guided configuration of embedded software systems. (2012) In Submission.
10. Behjati, R., Yue, T., Briand, L., Selic, B.: Simpl: A product-line modeling methodology for families of integrated control systems. Technical Report 2011-01, SRL (2011)
11. Behjati, R., Yue, T., Nejati, S., Briand, L., Selic, B.: Extending sysml with aadl concepts for comprehensive system architecture modeling. In: ECMFA. (2011) 236–252
12. Sabetzadeh, M., Falessi, D., Briand, L., Alesio, S.D., McGeorge, D., Åhjem, V., Borg, J.: Combining goal models, expert elicitation, and probabilistic simulation for qualification of new technology. In: HASE. (2011) (to appear).
13. Falessi, D., Sabetzadeh, M., Alesio, S.D., Briand, L.: Modus: A tool for goal-based safety and reliability assessessment of new technologies (2011) Submitted.
14. van Lamsweerde, A.: Requirements Engineering - From System Goals to UML Models to Software Specifications. Wiley (2009)
15. O'Hagan, A., Buck, C., Daneshkhah, A., Eiser, J., Garthwaite, P., Jenkinson, D., Oakley, J., Rakow, T.: Uncertain Judgements: Eliciting Experts' Probabilities. Wiley (2006)
16. Robert, C., Casella, G.: Monte Carlo Statistical Methods. Springer (2005)
17. Det Norske Veritas: Qualification procedures for new technology DNV-RP-A203, DNV, 2001.
18. Det Norske Veritas: Technology qualification management DNV-OSS-401, DNV, 2010.
19. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: A model for technology transfer in practice. IEEE Software **23**(6) (2006) 88–95
20. Ericson, C.: Hazard Analysis Techniques for System Safety. JOHN WILEY & SONS (2005)
21. Object Management Group (OMG): A UML profile for MARTE: Modeling and analysis of real-time embedded systems (May 2009)

22. Lewerentz, C., Lindner, T., eds.: Formal Development of Reactive Systems - Case Study Production Cell. In Lewerentz, C., Lindner, T., eds.: Formal Development of Reactive Systems. Volume 891 of LNCS., Springer (1995)
23. Tarr, P.: So you want to marry an industrial. Presentation. (2011)