Industrial Use of Domain-Specific Modeling: Panel Summary

Juha-Pekka Tolvanen
MetaCase

Niels Brouwers
Altran

Robert Hendriksen
SoLay-Tec and Sioux

Gökhan Kahraman
ASELSAN A.S

Jeroen Kouwer
Thales

Abstract

Domain-specific languages and modeling provide a viable solution for continuing to raise the level of abstraction beyond coding, making development faster and easier. This paper summarizes the panel on Industrial Use of Domain-Specific Modeling held at the workshop on Domain-Specific Modeling at SplashCon (Amsterdam, Netherlands 30th October 2016). Panelists included Niels Brouwers from Altran, Robert Hendriksen from SoLay-Tec and Sioux, Gökhan Kahraman from ASELSAN A.S and Jeroen Kouwer from Thales. Panel was moderated by Juha-Pekka Tolvanen from MetaCase.

General Terms Design, Languages, Verification.

Keywords industry experience; panel; domain-specific modeling; generators, modeling languages

1. Introduction

Raising the level of abstraction with languages, yet enabling the generation of code and other needed artifacts, has been a successful recipe for productivity and quality improvements for decades. In this panel, experienced industry experts were asked to share their experiences, both good and bad, on applying Domain-Specific Modeling (DSM) in various application areas. The panel discussion offered insight into the nature of DSM language design, implementation, and application, as well as the possibilities of diverse organizational introduction and use. The audience was invited to ask questions and join with their own opinions and experiences. The panelists included the following participants:

- Niels Brouwers, Software Architect, Altran
- Robert Hendriksen, Software Architect at SoLayTec and Sioux
- Gökhan Kahraman, Team Leader, ASELSAN A.S
- Jeroen Kouwer, Software Engineering Consultant, Thales

The bios of each panelist are provided at the end of the document.

2. Position Statements

Juha-Pekka Tolvanen from MetaCase acted as a moderator, introducing the panelists, who then gave their position statements as follows:

Niels Brouwers works at Altran in a competence center of 25 persons creating domain-specific languages (DSLs) and software factories, while further developing Altran's MDE competence. Language bridges, an MDE continuous integration environment, and portfolios are examples of assets that increase productivity to develop and industrialize the software factories and integrate them in the client's engineering process. Altran plays a significant role in the MDE ecosystem, mainly concentrated in the Brainport area in the southern part of the Netherlands. It is actively participating in research studies, the industrialization of MDE techniques in industry, and reporting on industrial challenges and experiences to research institutes.

Although generic modeling tools play a successful and useful role in the MDE ecosystem, Altran believes most gains in productivity, quality and reducing accidental complexity is only achievable with usage of domain specific software factories. These software factories provide a perfect fit to both the practitioner and the process in which it is being used. Abstraction of the created DSLs reaches up toward the problem domain to smoothen communication with stakeholders and reduces the engineering gap from problem to solution space.

His experiences are from several projects in which modeling languages, highly advanced generators and related tools are developed and applied in a complex industrial context. In this setting, 8 DSLs, consisting of 50-100 smaller DSLs, have been developed that are used by 20-50 engineers on a daily basis. Together, it is estimated that multi-million lines of code have been generated and integrated in products build for the high-tech industry. Typical DSLs show a productivity gain of at least a factor 5.

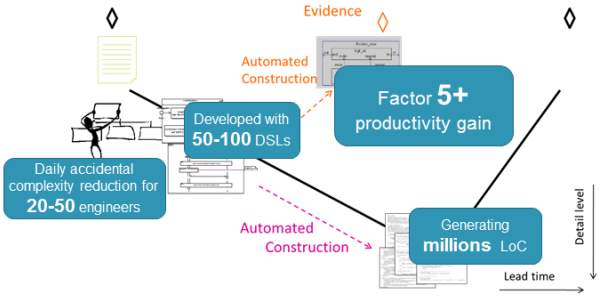


Figure 1. Figure 1: 8 DSL Software Factories industrialized in client's engineering process

Niels sees that the demand for software continues to grow rapidly, but the amount of people developing this software does not seem to grow at the same pace. Because it is challenging to develop all of the software for complex machines with reasonable time to market, there is a real problem in software development. Niels concluded that he has not seen a technique comparable to model-based approaches that provides similar results with productivity

improvements. Therefore, the real question is not "if" MDE/DSM/DSL approaches should be applied, but "how" these techniques can be introduced into an industrial ecosystem.

Robert Hendriksen has applied DSLs at SoLayTec for developing machines for atomic layer deposition (ALD) on solar cells. The use of DSLs started as skunk works, with a tool called oaw, and in a short period of time they started to obtain immediate benefits. Later, oaw was abandoned, but the principles continued to be applied. Robert explained the use of DSLs and generators in this domain in more detail (see figure below). First, a gas expert creates models of the system on the hardware side (e.g., heating, valves and other instruments connected with pipes). These are also used as the source for generators – initially for HAL (Hardware Abstraction Layer), but then also for various other purposes:

- Simulation is very helpful because often the hardware is not available; by generating simulators, development can be started earlier.
- State modeling can be used for machine description and control code behavior. Also, many stakeholders who are not software engineers can understand state machines and the importance of the states to their domain.
- Models can serve as a user interface for both a real system and a simulator.
- Models include information also from the hardware target, such as a PLC platform from a different company. The hardware interfaces can be generated, too. Tests can also be created from models for new acquired hardware (e.g., new panels from suppliers).

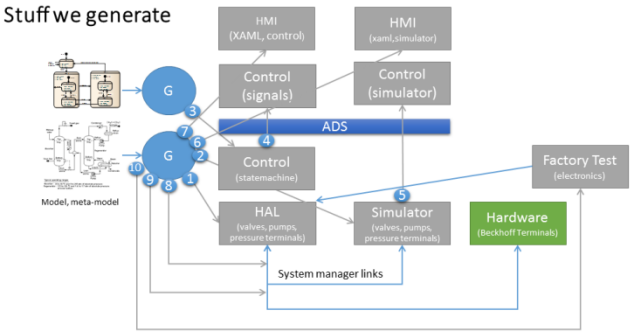


Figure 2. Stuff we generate, SoLayTec

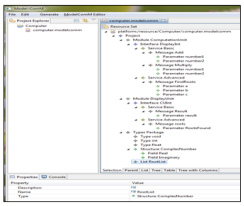
While SoLayTec has not measured the improvements in development speed, there really has not been a need to do so as it became obvious that the use of generators offered demonstrated benefit. The next challenge is to apply model checking, which has strong potential, but has not been applied yet at SoLayTec.

Gökhan Kahraman works at Aselsan in the role of senior expert engineer. He acts as a team leader of a DSL development research group that has developed many DSLs in various projects, as well as their maintenance while in use. Gökhan has completed a PhD in the area of DSL Quality by creating a framework for measuring and assessing the quality of DSLs. He has been directly involved in creating two DSLs targeting different needs (GVDYS and ATA, see Figure below). Both of these DSLs are in active use and maintained.

Göhkan summarized his group's experiences in terms of the benefits and challenges as follows:

+ Increase in productivity (5-7 times)
+ Maintain the created models easily and quickly
+ Decrease in the number of errors via automatic code generation
+ Multiple artifacts generated from interface definitions, like Communication Middleware, Test Driver, ICD
– Tooling maturity is often a challenge (Eclipse GMF: trouble in using it, especially when it comes to evolving the language, debugging!)
– Lack of support for large-scale projects (theoretically and tooling); in particular, when using multiple DSLs that need to be integrated

- GVDYS
  - DSL for supporting the rapid development of embedded software data intensive modules
  - Facilitate to make fast change on the models
  - Efficient, high-level programs
  - Used in production for > 5 years
- ATA
  - DSL for defining interfaces in distributed systems
  - Generate the communication middleware, test driver, and interface control document
  - Used in production for > 4 years
- Phd. on the Quality of DSLs



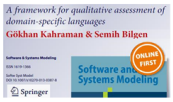


Figure 3. ASELSAN background to DSLs

Jeroen Kouwer is a software architect at Thales serving as an engineering consultant. Thales has applied modeling and generators for developing sensors for over 10 years, e.g., a sensor suite in a ship monitoring the environment using radar. After 10 years, they have evaluated whether the effort has been worthwhile and the answer is clear that their investment has produced great benefit. Jeroen introduced the TCO approach (Total Cost of Ownership) to modeling (Figure below).

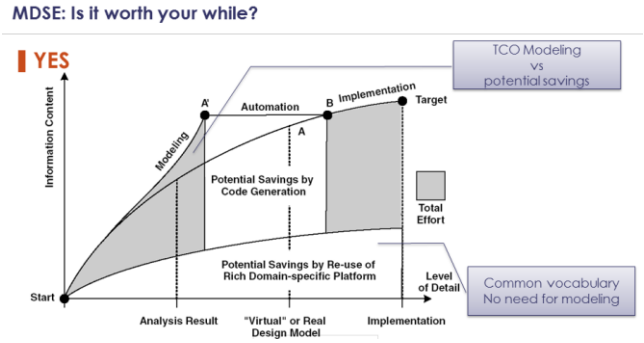


Figure 4. Total Cost of Ownership to investigate if MSDE is worth.

In particular, he emphasized that a DSL helps to define the terminology of the domain. He gave a simple example to illustrate the benefits: nobody is expected to speak about a 'boat' as the domain concept is a 'ship'. A prerequisite for successful modeling is a (rich) domain-specific platform. Having this domain data in a model is very useful because it can be used for various generation purposes.

Thales is using Capella, a system modeling language, and an internal framework. The modeling is done with UML and stereotypes on top of this platform.

Jeroen told the workshop attendees that they do not have measurements based on calculated gains and investments, but they know that the second sensor they built took only a few weeks, but the first effort took a few months. This gives a clear indication that there are productivity improvements in place with their TCO approach.

3. Audience questions and discussion

Because all the panelists were clearly in favor of modeling and the use of generators, the audience asked if someone could play the role of "devil's advocate." This led the audience to ask how panelists deal with the challenges they have faced. The following topics were addressed by the panel.

3.1 How to deal with legacy code and how to introduce it when moving to modeling with DSLs?

Robert told that if there is an existing system, its evolution can be handled by module revolution: take changes of the system apart and redo them with a model-driven way, he called "System evolution by module revolution".

Gökhan agreed with Robert and mentioned that their DSLs are also based on existing code, but they re-generate code based on the DSL. He has observed that a legacy system is not an obstacle to using DSLs. He believes that DSLs bring a great opportunity to legacy system evolution.

Jeroen stated that if a legacy system can be included as a reusable asset, then the maintenance is easier. A challenge observed at Thales is that the interfaces to legacy components and the code generator need to be consistent (i.e., code is generated according to the interfaces). He sees that before the language can be implemented first recurring patterns need to be identified and used as candidates for language constructs. Since with legacy there were clear interfaces and components having those interfaces it became natural that they should have a system modeling language using these concepts while modeling too.

There was also a question about general guidelines on identifying architectural concepts and if there is known literature on the topic. A suggested book on the topic was: Domain-Specific Application Frameworks, edited by Mohamed E. Fayad and Ralph E. Johnson.

3.2 If technology changes at a fast pace, how can we manage the changes to the modeling solution?

Jeroen spoke about how he and his colleagues have seen management support important to handle the evolution in the past. Now they are introducing a new framework that requires a new language and expects management support for that too. In this time the new language will be used by current engineers and rely on existing toolsets as well.

Niels mentioned that they handle the evolution of languages and generators for their customers. When the models do not match a new metamodel, they create migrators as needed. Altran also ships the migrators, if needed, with the developed modeling solutions to their customers. Metamodel co-evolution of other artefacts such as concrete syntax, code generators and validation rules is still a problem that currently negatively impacts the productivity of the toolsmiths.

Gökhan shared his experience that minor changes are manageable, but larger changes are still challenging; in particular with their case when the domain changes several times over a year (~4/year). Tools could help ease the challenge and make the process easier. When asked how the model changes are done – automatically or manually – he stated that changes are done manually.

3.3 Why certain industries, like those presented in the panel, are ready to use modeling, but others not?

Robert voiced his opinion that one reason for his group is that the language they are using fits well to the domain they are working with. He also sees their language as easy to adopt by domain experts. Perhaps other fields do not yet have such suitable languages.

It was also discussed among the panelists that perhaps in some areas it is just hard to identify a good language, or that there is no support from the company side to try new things.

The panelists and participants (30) of the workshop were all software developers. It was observed that if technology experts do not have a problem domain it may make challenging to identify appropriate language constructs.

3.4 How to introduce DSM/DSL/MDD?

The discussion ventured into a reflection on how to best introduce DSLs and modeling into the culture of a development organization. A comment from the audience was that there is a simple solution: Budget. "You make the budget for the project so small that no other technology than model-based development with domain-specific languages can solve the issues." This somewhat strong proposal seemed to receive general acceptance among the workshop attendees.

Niels added that the introduction of DSLs will be successful only if both architects/engineers and management are convinced about the benefits and return on investment. From a technical perspective, it is helpful to be able to demonstrate how DSL/MDE techniques work using concrete examples (e.g., application of general software techniques instead of magic, how it increases quality, supports architecture and improves communication with stakeholders). Management can be convinced by productivity gains, platform independence (risk) and adoption of techniques at peer companies. Secondly, depending on the company, the strategy used to introduce DSL/MDE techniques might be chosen differently. In one company, it may be beneficial to start bottom-up; i.e., implement a horizontal DSL within a single project and gradually increase the level of abstraction and/or expand to multiple projects. Alternatively, start from top-down, where the MDE solution is defined at the correct level of abstraction (up until problem area) and gradually implemented to support all aspects. Niels has seen both strategies work successfully.

3.5 Role of "reverse" engineering / creating or updating models

In addition to code generation, the audience asked about other reasons for using models. A question was asked, "Is there work being done on updating the models based on external sources, like asking computers to do part of the work, rather than humans?"

Robert emphasized the use of models to visualize the system to show errors while design or during execution.

Niels also told that in theory they could use models also to examine external behavior of legacy components with models, if needed.

Also, applying models in debugging mode, as presented in the workshop, was mentioned by the audience as a benefit, as well as incorporating the test results or simulation results to the models or to the generated artefacts from the models. It was also emphasized that models can be used just to visualize things for customers so that validation (are we solving the right problem) is done together with the customer.

3.6 How to sustain the momentum and move to the next domain within the company?

Although panelists have testified about several cases of success (5x productivity or improved quality), it is sometimes hard to "sell" the idea of DSLs again to a new project in the company. A key question to the panelists was how they created several DSLs across different projects, and how they then managed to move to the next DSL creation project?

Jeroen mentioned that they are currently in a process of having a software architecture language already in place and leverage it now projects using it in France and in the Netherlands. Once management smells money due to benefits gained then it makes re-applying modeling easier. In Thales case they could show improvements with data reducing development time from 3 weeks to 3 days. Those kinds of numbers make a success and they have now 50% of all code generated.

Robert told that they have had similar experiences – albeit not having used DSLs so long yet.

Gökhan told that when presenting the modeling and code generation idea he collected data from the projects done and presented this data to the management. It helped then to make the change for the next project.

3.7 What are the current challenges?

When asked about the challenges, **Niels** summarized the main issues as follows:

- How to quantify gains of adopting MDE in the engineering process? Especially when working with a new customer, it is hard to estimate the gains in terms of productivity and quality.
- How to remove resistance by software engineers that prevent adoption of MDE? While some people fear being replaced by "code generators" (he does not see this as a valid concern), this issue could be addressed better.
- How to reduce complexity to develop the software factories?

Gökhan mentioned the following challenges met at ASELSAN:

- How compatible are new DSLs when integrated into the software development lifecycle of large-scale and distributed systems?
- How can we provide language interoperability whereby DSLs and GPLs can co-exist and work together, such as when multiple DSLs and GPLs capture different system aspects in a large system.
- How can we obtain high-quality languages, which may be a key toward obtaining high quality software?
- How do we address language evolution concerns when the DSL specification changes? DSLs evolve as the concepts in a domain evolve. This is a relevant challenge in EMF/GMF within Eclipse.
- Poor tooling (user friendly tooling, insufficient debugging tools) remains a deep concern.

Jeroen raised the essential question about whether we are modeling the right kinds of things. Thales does not create only system architectures, but sensors and sensor systems as well as integration, yet they model system and software architectures. After many years of modeling he is wondering whether they are modeling the right thing. Should they model sensors and sensors systems rather than architectures?

4. Closing Remarks

Jeroen restated the key question he made earlier: Are we modeling the right thing? Is the language operating at the right level of abstraction?

Gökhan emphasized the challenge of creating good quality languages and keeping up their quality during the maintenance phase.

Robert pointed to the future and mentioned integration with models and AI, and the use of guidance for making generators. He does not see that AI provides all of capability for implementing generators, but AI can offer new capabilities over past approaches.

Niels wants to see more use of modeling with DSLs because he does not envision any other technology that is able to provide similar results.

5. About the Panelists

Niels Brouwers is a software architect at Altran and specialized in the field of model-driven engineering. His passion for model-driven engineering originated in 2007 and was further pursued by joining Altran in 2011, a global leader in innovation and high-tech engineering consulting that strongly believes in model-driven engineering. For more than 5 years, he has led multiple teams in the development of DSLs and advanced code generators that are applied in a large industrial software organization.

Robert Hendriksen joined Sioux, which is based in Eindhoven, the Netherlands, in 2006. He has been involved in various projects for Sioux, but at the moment, he works on a full-time basis as software architect for SoLayTec's products. Any time left after his core responsibilities is devoted to the construction and application of DSLs.

Gökhan Kahraman received the M.Sc. and Ph.D. degrees in Electrical and Electronics Engineering from Hacettepe University, and Middle East Technical University (METU), Ankara, Turkey, respectively. He is currently working as a senior expert software engineer at ASELSAN A.S. in Turkey. He has over 10 years of experience in embedded software development using model-driven development and DSM approaches, taking on developer, architect and team leader roles in large scale and complex system projects. He is the team leader of the DSL development team in the ASELSAN-REHIS group. His team designed and implemented many DSLs that are used in several projects. These DSLs continue to evolve and are maintained. His Ph.D. work focused on the assessment of DSLs and his current research interests include the quality of DSLs and cyber-physical systems.

Jeroen Kouwer started his career in 1998 at Thales and has worked since then for various companies, and then rejoining Thales at the end of 2006. Upon returning to Thales, he started work on a software service framework and the modeling methodology on top of this framework. Since then he has applied his modeling and software skills in various projects. He has experience with C, Java, modeling, meta-modeling and DSL development. He has a strong focus on enhancing and simplifying developing and testing of software.

Juha-Pekka is the CEO of MetaCase, a company providing MetaEdit+ tool for DSM. Juha-Pekka has worked with model-driven development and tools, notably metamodeling and domain-specific languages and models, since 1991. He has acted as a world-wide consultant for modeling language development, authored a book on Domain-Specific Modeling, and written over 70 articles for various software development magazines and conferences. Juha-Pekka holds a Ph.D. in computer science from the University of Jyväskylä, Finland.