Putting the Pieces Together – Technical, Organisational and Social Aspects of Language Integration for Complex Systems

Håkan Burden

Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden
burden@cse.gu.se

Abstract. Dealing with heterogenuous systems is often described as a technical challenge in scientific publications. We analysed data from 25 interviews from a study of Model-Driven Engineering at three companies and found that while the technical aspects are important, they do not encompass the full challenge – organizational and social factors also play an important role in managing heterogenuous systems. This is true not only for the development phase but also for enabling early validation of interdependent systems, where processes and attitudes have an impact on the outcome of the integration.

Keywords: Empirical and Exploratory Case Study, Model-Driven Engineering

1 Introduction

Complex systems, consisting of numerous and interdependent subsystems [15], require a plethora of languages for efficient implementation [7]. From the aspect of Model-Driven Engineering (MDE), the challenges are often described in technical terms [4] since heterogenuous languages imply different abstraction levels, representations and aspects of software [8], but also since the languages have their own domain-specific and platform-dependent constraints [11]. The one-sided focus on technical aspects is surprising since Kent already in 2002 pointed out that if MDE is to be successful it needs to encompass also the organisational and social aspects of software engineering [10], a claim that has since been reiterated [1, 9].

To explore to what extent language integration for comlex systems is a challenge in terms of technical, organisational and social aspects we analysed data collected at three different companies, looking for evidence regarding the motivations and challenges of heterogenuous development of embedded systems. Among the findings are that engineers tend to favour integration at the concrete code level instead of at the more abstract model level, that management needs to fit the right team with the right task and that which language you use identifies you as a software engineer.

The following section will describe the context of the three companies as well as how the data was collected and analysed. Section 3 structures the findings according to technical, organizational and social aspects of language integration. We then conclude and present our intentions to further investigate the interdependency between the factors in Section 4.

2 Model-Driven Engineering at Three Companies

During 2013 we conducted a case study of MDE at three companies – Volvo Cars Corporation, Ericsson AB and the Volvo Group. The respective organisations had different experiences with MDE but were all transitioning towards a more agile way of software development.

2.1 Heterogenuous Languages for Complex Systems

Electronic Propulsion Systems (EPS) is a relatively new unit at Volvo Cars with the responsibility of developing software for electric and hybrid cars. MDE was introduced in a step-wise manner as software development went from prototype vehicles to mass-production. The overall system design is described using AUTOSAR[1] while the software developed in-house at EPS is implemented using Simulink[2]. Simulink is also used for validation and integration purposes. The interfaces of the Simulink models are generated from the system-wide model. Besides graphical modelling languages, C is used for low-level details while numerous scripts help in everything from translating between different AUTOSAR-standards encoded in XML to deploying software on hardware.

Radio-base stations at Ericsson AB has employed different MDE technologies since the late 1980's, primarily focusing on UML as a descriptive and prescriptive modelling language including code generation. Besides using UML for design, implementation and testing various other languages form the engineers' tool box; C is used for functionality relying on optimal hardware performance, Java has a niche in functionality requiring GUIs, Erlang is regularly used for testing while home-made domain-specific languages – DSLs [12] – are used for specification, implementation and testing purposes.

Volvo Group Trucks Technology develops software for the Volvo Group's truck brands. While most of the software development is outsourced a few features are developed in-house using C. The interfaces of the top-level software components are automatically generated from the system model which is described in a company-specific dialect of EAST-ADL[3]. The two Volvo companies are independent in all aspects besides the name space which they share for historical reasons.

[1] http://www.autosar.org/
[2] http://www.mathworks.se/products/simulink/
[3] http://www.east-adl.info/

2.2 Data Collection and Analysis

The main source of data comes from 25 semi-structured interviews – 12 at Volvo Cars, nine at Ericsson AB and four at Volvo Trucks. The reason for fewer respondents at Volvo Trucks is that they entered the project later than the two other companies and have fewer engineers involved in MDE tasks. The interviews were audio-recorded and then transcribed. The interviews were complemented by a combination of observations, informal interaction [5] as well as seminars and regular meetings with representatives from the three companies.

The data has previously been analysed regarding the impact of tools on MDE adoption [16] as well as for comparing and contrasting MDE at the three companies [3]. For the purpose of this contribution we re-analysed the data deductively [14] searching for evidence concerning the technical, organisational and social aspects of heterogenuous systems and language integration across the model-driven engineering activities at the three companies.

3 Findings

As seen in the previous section, a variety of languages is used across the software. The variation comes both in terms of adapting languages depending on the nature of the included subsystems, but also due to where in the lifecycle the language is to be applied.

3.1 Technical Aspects of Language Integration

From the interviews a recurring theme is that the tool used for encoding a solution is just a means for producing low-level code. The following quote is from Ericsson, "*I don't see Rose RT or another modeling tool as a language. It's what they produce that is the language, and mostly it's always been C++ for us. So I don't think – I can consider, for example, Rose RT as a tool like Eclipse or something. Lets you develop code.*" A similar experience was encountered at Volvo Cars when one interviewee was asked about the impact of changing implementation language from C to Simulink, "*You still have C code.*"

The emphasis on the generated language is also dominant in how multiple languages are to be integrated. Where academic research is focused on composing modeling languages on a meta-level [4, 7], industrial practitioners prefer to integrate at the code level. At Ericsson where multiple languages are used in various combinations, integration is mainly done at the code level through the build environment. This coincides with our findings from a previous study on the integration of a graphical modelling language and a textual DSL conducted within another organisation at Ericsson [2].

Due to the number of suppliers and sub-contractors both Volvo companies rely on being able to integrate their subsystems in the form of binaries, so that merging on a meta-level is impossible. However, having access to the source would still be beneficial but for debugging purposes, not for merging the partial solutions.

One of the few examples where two languages are integrated in the same development environment comes from Ericsson where a DSL for testing was developed internally, "*The reason for it being that it could often take quite long time to compile some of our models [. . .] from a few minutes up to a few hours, depending on how big the model was.*" The answer was to define their own testing language on top of the modeling language and the interviewee was told that the DSL was developed during all the hours the team sat waiting for the model compiler to terminate.

3.2 Organisational Aspects of Language Integration

An engineer at Ericsson expressed the need for management to assign tasks according to the skills of the developers. "*In the big systems, we have different languages. Yes, and that is complex. And that's a problem. It's hard to expect that the software designer or verifier is equally good in all languages. And we've had to handle that in our team.*"

While one engineer at Ericsson saw the need for different languages according to domain and platform constraints – "*so you have all these levels and you have very different requirements for different parts of this product. It's so big that I don't even think that one solution fits all. You should be able to use several approaches*" – there is still an organisational wish to limit the number of languages to limit accidental complexity "*you shouldn't do it without a need. So you shouldn't – if you try to solve the same problem, I think you should try to use similar language or similar ways of working.*"

Stable interfaces in the decomposition of complex systems is desirable [4] but not always possible to obtain due to changing requirements or underspecification of new and ground-breaking features [6]. In these cases an agile organisation that lets developers work at both ends of the interface can be a way forward even if this demands that the developers master more than one language. But as one engineer at Ericsson said, "*using a new language is probably the smaller problem compared to learning the product and the domain and everything*".

3.3 Social Aspects of Language Integration

"*If you go to a different language, like, I don't know, whatever language, it will say that the for loop looks like this, but the functionality of it is the same. It doesn't matter how you put it in the words. Instead of 'for', you put like an 'f' or whatever. It's the same functionality. So when you know the base, you don't need to learn, like really study the new ones. You only adapt to them. From my point of view.*" The quote is from an engineer at Volvo Cars who explains his perception of using different languages for similar tasks.

However, not all engineers are interested in learning new languages. One interviewee from Ericsson described his experiences from developing a customer interface with a team located in a different town in the following way, "*an option was to do it in C++ because that's the most cost effective way. And the owner* *said something like 'I don't think that's a good idea because the organization we're from, people are working there because they want to do Java'.*"

At Volvo Trucks a similar sentiment was aired as the topic of introducing a new modelling language was raised, "*we have a lot of people that like to write C code and they like script languages. And they always do scripts for something. And they are pretty comfortable. They like writing a code with a blank page just writing C code.*"

4 Conclusions and Future Work

From a technical aspect of heterogenuous language integration there is a difference between the emphasis of academic contributions and industrial praxis in that while the former focus on merging languages on a source or meta-level the latter successfully integrating the target representations. This pragmatic approach is supported by proven techniques developed for integrating third generation programming languages. Organisationally the challenge seems to be for management to assign the right team – with the right skills – to the right task, a parallel challenge to the challenge of applying the right tools to the right problem [16]. Finally, from a social aspect it is not just enough that the engineers have the right skills – they also need to be open for using new languages. This is due to the fact that learning a new language is not a major obstacle but which language(s) you use is part of your identity as as software engineer and not all engineers are willing to redefine their competencies. In relation to Kent's critique of MDA [10], it seems that while the technical aspects of language integration have an important role to play in the development of complex systems, the possibilities for improved development and product quality can only be realised if the organisational and social aspects are seen as equally important.

In the case of setting up a simulation environment at Volvo Cars the challenge is not just to integrate different modelling languages but also agreeing on the same version of Simulink since different versions imply different properties in the generated code. Here, the challenge is organisational due to the fact that the developers want the newest features which enable new solutions while management responsible for integration need to know that the new version is stable before updating. Updating legacy models to comply with the newer versions can be both a time consuming and an error-prone task. With external organisations submitting their intellectual property in the form of human-readable models or code the question also becomes an issue of trust – not a technical factor of how to best compose two or more languages. How to organise an ecosystem [13] of simulation environments for continuous integration is still an open question we hope to address in future work by exploring how technical, organizational and social factors coincode in language integration.

References

1. Ameller, D.: Considering Non-Functional Requirements in Model-Driven Engineering. Master's thesis, Universitat Politcnica de Catalunya (June 2009)

2. Burden, H., Heldal, R., Lundqvist, M.: Industrial Experiences from Multi-Paradigmatic Modelling of Signal Processing. In: 6th International Workshop on Multi-Paradigm Modeling MPM'12. ACM, Innsbruck, Austria (October 2012)
3. Burden, H., Heldal, R., Whittle, J.: Comparing and Contrasting Model-Driven Engineering at Three Large Companies. In: ESEM 2014, 8th International Symposium on Empirical Software Engineering and Measurement. Torino, Italy (September 2014)
4. Combemale, B., Deantoni, J., France, R., Boulanger, F., Mosser, S., Pantel, M., Rumpe, B., Salay, R., Schindler, M.: First Workshop On the Globalization of Modeling Languages (GEMOC 2013). In: CEUR-WS (ed.) GEMOC - 1st International Workshop On the Globalization of Modeling Languages. pp. 3–13. Benoit Combemale, Julien DeAntoni, Robert France (September 2013)
5. Davis, K.: Methods for studying informal communication. Journal of Communication 28(1), 112–116 (1978)
6. Eliasson, U., Burden, H.: Extending Agile Practices in Automotive MDE. In: XM 2013, Extreme Modeling Workshop. Miami, USA (October 2013)
7. Hardebolle, C., Boulanger, F.: Exploring Multi-Paradigm Modeling Techniques. Simulation 85(11-12), 688–708 (November 2009)
8. Hardebolle, C., Syriani, E., Sprinkle, J., Mészáros, T.: Summary of the 6th International Workshop on Multi-Paradigm Modeling. In: Proceedings of the 6th International Workshop on Multi-Paradigm Modeling. pp. 5–6. MPM '12, ACM, New York, NY, USA (2012)
9. Hutchinson, J., Whittle, J., Rouncefield, M., Kristoffersen, S.: Empirical assessment of MDE in industry. In: Proceedings of the 33rd International Conference on Software Engineering. pp. 471–480. ICSE '11, ACM, New York, NY, USA (2011)
10. Kent, S.: Model Driven Engineering. In: Proceedings of the Third International Conference on Integrated Formal Methods. pp. 286–298. IFM '02, Springer-Verlag, London, UK (2002)
11. Mellor, S.J., Kendall, S., Uhl, A., Weise, D.: MDA Distilled. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA (2004)
12. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM computing surveys (CSUR) 37(4), 316–344 (2005)
13. Messerschmitt, D.G., Szyperski, C.: Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press Books 1 (2005)
14. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering: Guidelines and Examples. Wiley (2012)
15. Simon, H.: The Sciences of the Artificial. Karl Taylor Compton lectures, MIT Press (1996)
16. Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., Heldal, R.: Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In: Moreira, A., Schaetz, B. (eds.) MODELS 2013, 16th International Conference on Model Driven Engineering Languages and Systems. Miami, USA (October 2013)