Model Driven Software Engineering creates tomorrow's legacy

Mark G.J. van den Brand
Eindhoven University of Technology
Eindhoven, The Netherlands
m.g.j.v.d.brand@tue.nl

ABSTRACT

Software is everywhere we go. Among others, it powers the devices we use in our daily life, it channels our social interactions via social media, it enables our medical care. With the increasing number of applications, the amount of software is exponentially increasing, which challenges the way we develop and maintain our software efficiently and effectively.

The software engineering (research) community is fully aware of these challenges and attempts to tackle these by introducing new development techniques, such as Agile software development, test-driven software development and model driven software technology. The latter advocates the use of models and domain specific languages (DSLs) to speed up the development of software, to increase understandability and quality of the resulting software.

Modeling and modeling languages are common practice in other engineering domains, such as mechanical engineering. Modeling in software development has accelerated with the launch of Unified Modeling Language (UML). The use of multiple modeling languages in UML has led to the development of Meta Object Facility (MOF) and its derivative Eclipse Modeling Framework (EMF). Model driven software engineering advocates the use of small languages that are domain specific, provide a higher level of abstraction and facilitate code generation. UML already offered the possibility to create domain specific extensions via profiles. MOF and specifically EMF have given rise to the creation of small languages using the Eclipse environment. EMF and the tooling using EMF, such as Xtext, ATL, QVTo, ETL, etc., has led to an acceleration of the development of domain specific languages.

The following challenges can be observed with respect to this development.

(1) The tooling used to create and use DSLs is far from mature, is unstable, and gets rapidly deprecated. There is still a lot of effort needed to improve the existing tooling.
(2) The creation of a DSL involves understanding of the domain for which the languages is created and having the capability of translating this knowledge to concepts at the right level of abstraction.
(3) The increased level of abstraction and introduction of domain concepts makes the models harder to understand and maintain. The use of DSLs involves also a risk, if the developer(s) of a DSL leaves the company then the maintenance of the DSL may be jeopardized. The number of developers that are able to understand and maintain DSLs is low; the number of developers understanding general purpose languages, e.g. C, will always be higher.
(4) The interactions between software models and models from other (system) engineering domains, e.g. describing physical behavior, are becoming more and more important.

The (high-tech) industry has adopted model driven software engineering and started introducing DSLs. These DSLs are prototypical, because capturing and defining domain concepts as well as language development is new for them. Before the DSLs stabilizes, tens and sometimes even hundreds of models may already be created. These models have to be migrated and can not be thrown away, if the DSLs are adapted. These languages and corresponding models will become legacy if we do not act by means of developing proper development methodologies and tooling to support analysis of languages and models and the evolution of both languages and models. So, both language and domain are in state of flux and have to co-evolve, both on the syntactic but definitively also on the semantic level, but this is hard and difficult work. We do not have the tools for this and if we do not act now, we will have the same legacy as we have software from a code perspective

One can wonder whether model driven software engineering is indeed the next silver bullet to transform software development or rather a silver-painted egg that when being fired creates a big mess. Research on model driven techniques should move away from just focusing on tooling, although that still work has to be done to make the tools more robust and usable, but start focusing on proper methodologies to extract domain concepts in order to create usable DSLs. It should deal with evolutionary aspects of DSLs and created models and work on stabilizing the tooling needed to create languages and corresponding models, ensuring consistency between languages and between models, and between languages and models. These are just a few of the challenges that we are facing. If we are able to make this happen then we might have a silver bullet after all and the promised increase in quality and productivity can be realized.

ACM Reference Format:
Mark G.J. van den Brand. 2018. Model Driven Software Engineering creates tomorrow's legacy. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn