(Mega)modeling Software Language Artifacts

Jean-Marie Favre [1], Dragan Gašević [2], Ralf Lämmel [3]

[1] OneTree Technologies, Luxembourg
[2] Athabasca University, Canada
[3] University of Koblenz, Germany

1 Description

Modern software is typically made of heterogeneous sets of software artifacts including for instance databases, programs, transformations, grammars, models and metamodels, compilers, interpreters, formats, ontologies, frameworks, APIs, schemas, configuration files, makefiles, etc. In practice particular languages, tools, implementations, and standards are used such as SQL DDL, Saxon, XLST, Java, Hibernate, XSD, OWL, DOM, Antlr, UML, XMI, Ecore, Awk, and so on. In the absence of a conceptual framework it is difficult to understand the relationships between these software artifacts, if any. The goal of this tutorial is to provide such a framework, showing that the similarity and relationships between techniques can be modeled at a high level of abstraction, and even more importantly that recurring patterns occur in such models. Some of these patterns, for instance those involving "bridges" between technologies, would be really difficult to grasp without a proper conceptualization. As a result software engineers and researchers usually find it hard to understand the intricacies of technologies that are out of their area of expertise and it is more than likely that they do not realize the analogies that exist between heterogeneous technologies. This tutorial aims to unveil these recurring patterns and to show participants coming from different horizons how to model the technologies they design or work with in an uniform way and how to situate them into the overall software language landscape.

In the first part of the tutorial, the notions of software languages and technical spaces are briefly presented with a special emphasis on their unifying character. Then fundamental relations such RepresentationOf and ElementOf are introduced forming the basis of a (mega)modeling framework. Recurrent patterns based on these relations are then presented, allowing to describe for instance the "conformance" relation between let's say a program and a grammar, an xml file and an xsd schema, or an uml model and its metamodel, etc. More complex patterns such as bridges between technologies (e.g. XML <==> Relational, OO <==> XML, etc.) are defined following the same approach. Though this notion of bridges seems easy to grasp informally at the first sight, it often leads to a rather large and complex set of technologies that are hard to understand and compare without an appropriate framework.

In the second part of the tutorial, the use of (mega)modeling framework is illustrated through its application in three different technical spaces: Grammarware, Modelware and Ontologyware. Concrete examples of various degree of complexity

are provided in each case, with again an emphasis on similarities between technical spaces. The hope of this approach is that it should be possible for someone with some knowledge in technical spaces (let's say grammarware) to improve significantly his or her comprehension about another space (let's say ontologyware), and this by virtue of analogy. It is our believe that the (mega)modeling approach, by raising the level of abstraction and focusing on essential software language concepts, enables both to better understand complex structures involving many heterogeneous software artifacts, but also to better apprehend new technologies coming from other spaces.

2 Presenters

One of the objectives of the tutorial is to show that bridges can be successfully established between heterogeneous technical spaces such as Modelware, Ontologyware and Grammarware, and in particular to go beyond traditional divisions of fields of expertise. This tutorial being directly inscribed into a "community engineering" perspective, we believe that having three presenters coming from different horizons would be the best way to insist on the integrative aspect of the mega-modeling approach.

- Jean-Marie Favre is a software anthropologist and a software language archeologist. He is principal scientist at One Tree Technologies. He has published numerous papers and coedited a book (in French) Beyond MDA: Model Driven Engineering. He has given tutorials and keynotes in more than dozen of international events and summer schools and has organized various national and international events. His research interests include software language engineering, software linguistics, software evolution and reverse engineering, model driven engineering and research 2.0.
- Dragan Gašević is a Canada Research Chair in Semantic Technologies and an Associate Professor in the School of Computing and Information Systems at Athabasca University. His research interests include semantic technologies, software language engineering, technology-enhanced learning, and service-oriented architectures. He has (co-)authored numerous research papers and is a led author of the book "Model Driven Engineering and Ontology Development." He has given tutorials at many well-known conferences such as WWW, ISWC, and CAiSE.
- Ralf Lämmel is Professor of Computer Science at University of Koblenz-Landau. In his career, he also served at Microsoft Corp., Free University of Amsterdam, Dutch Center for Mathematics and Computer Science (CWI), and University of Rostock. Ralf Lämmel is generally interested in the combination of software engineering and programming languages. Together with the other tutorial speakers and further researchers, he is one of the founding fathers of the SLE conference. He is one of the founding fathers of the summer school series GTTSE--Generative and Transformational Techniques on Software Engineering.

3 Content

The tutorial will be divided in various parts introducing first the key issues relative to modeling heterogeneous software language artifacts, then presenting the conceptual framework for mega-modeling, and finally examples of application in different technical spaces.

Provisional outline

- Introduction and objectives (5')
- Technical Spaces and Software Languages (5')
- Megamodeling fundamentals (10')
- Application to Grammarware (10')
- Application to Ontologyware (10')
- Application to Modelware (10')
- Synthesis and conclusion (10')

4 Relevance to GPCE/SLE attendees

Because of its emphasis of software language artifacts, this tutorial should be of interest to SLE attendees. Compilers, transformations and generators being extensively used in generative programming the tutorial should also attract GPCE attendees. We believe that providing a conceptual and unified approach for mega-modeling and showing its application across various technical spaces could improve cross fertilization between communities.