Compositional Modeling Languages in Action: Engineering and Application of Heterogeneous Languages with MontiCore

Nico Jansen
Software Engineering
RWTH Aachen University
Aachen, Germany
jansen@se-rwth.de

Bernhard Rumpe
Software Engineering
RWTH Aachen University
Aachen, Germany
rumpe@se-rwth.de

Nico Jansen is a research assistant at the Department of Software Engineering at RWTH Aachen University. His research interests cover software language engineering, software architectures, and model-based software and systems engineering.

Bernhard Rumpe is a professor heading the Software Engineering department at the RWTH Aachen University, Germany. His main interests are rigorous and practical software and system development methods based on adequate modeling techniques. This includes agile development methods as well as model-engineering based on UML/SysML-like notations and domain-specific languages.

Abstract—**As modeling languages become increasingly sophisticated, the reusability of individual language components and their integration into full-fledged languages is essential in language engineering. While different composition techniques have been established over the past years, language engineering endeavors still too often start from scratch instead. One reason is that although these techniques, such as extension, embedding, or aggregation, are elaborated and studied, they are rarely used in practice outside of research. This tutorial aims to bridge this gap by providing a practical course in which heterogeneous languages are developed and further integrated via advanced composition techniques. Using the MontiCore language workbench, we demonstrate hands-on the benefits of reusability in language development such that the advantages are already experienceable in a single session. The tutorial starts with a short introduction about language composition techniques with their respective fields of application and then proceeds to the practical part, where participants develop modeling languages themselves, combine them, and use the integrated result.**

Index Terms—**Domain-Specific Languages, Model-Driven Engineering, Software Language Engineering, Language Composition, Language Reuse**

I. Basic Information

- **Proposed length:** 90 min
- **Level of the tutorial:** Advanced
- **Required prerequisite background:** Basic knowledge of modeling languages, their application, and their engineering

II. Tutorial Content

A. Description of the Tutorial

There is a gap in the development of modeling languages between research and application. Model-Driven Development [1] gets increasingly involved in the engineering process of different domains. As a result, corresponding modeling languages, independent of whether domain-specific (DSLs) or general-purpose (GPLs), are becoming more sophisticated. A prominent example is the new SysML v2 [2]. However, this progress also means that their development, maintenance, and evolution are becoming more and more complicated and time-consuming. Therefore, various composition techniques [3] were designed to take advantage of reuse and improve development through language modularization and employing language component libraries [4]. Most state-of-the-art language workbenches and frameworks [5] support multiple composition techniques. While their names and exact realizations may vary in different tools, prominent language composition practices are (cf. [6]):

- **Inheritance** is the most basic composition type, which allows for extending a single language. It involves adopting the constructs of an existing language definition, overwriting these, and adding new ones. A special variant of this technique is *conservative extension*, in which models of the original language remain valid, i.e., only new constructs are added optionally.
- **Embedding** provides for reusing multiple language definitions and combining those in a single definition.

Here, the constructs of all languages are adopted and suitably coupled with each other. A classic example is the embedding of different language components (such as expressions, statements, etc.) [4] in an existing host language. Both inheritance and embedding are usually realized via the particular language definition, i.e., a grammar or metamodel.

- **Aggregation** also integrates multiple languages, but unlike the other techniques, it preserves the models as separate artifacts. This means that models of different languages operate loosely coupled in a shared context and can reference each other. The corresponding coupling and cross-referencing are usually realized via the symbol table [3] of a language.

Despite these advances, in practice, language engineers still often start from scratch when developing a new DSL. Composition techniques, although understood in concept, are only little used.

This tutorial tackles this problem by introducing the various composition techniques hands-on, explaining their uses, and applying them directly in this context. After a short introduction, participants will face the challenge of engineering different modeling languages, which can be created efficiently by employing existing language components. Furthermore, the constructed languages are loosely coupled via language aggregation and thus extended to an integrated language family. The tutorial will be conducted using the MontiCore language workbench [6]. The developed language family is directly operational and will be used for modeling at the end of the session, demonstrating that proper application of language composition increases engineering efficiency and quality.

B. *Intended Outline*

1) Short introduction via slides (ca. 25 min)
    - Overall topic
    - Brief introduction to MontiCore
    - Different language composition techniques
    - Goal in the practical part
2) Short setup phase for the audience (ca. 5 min)
3) Engineering of two small languages also reusing existing language components(ca. 20 min)
4) Aggregating these languages (ca. 20 min)
5) Modeling and applying the newly established language family (ca. 20 min)

III. Further Information

A. *Novelty of the Tutorial*

To the best of our knowledge, there has been no tutorial yet, that focuses on the application of different language composition techniques in action that allows practitioners to hands-on experience the integration of multiple modeling languages.

B. *Required Infrastructure*

A standard notebook is sufficient to participate in the tutorial. In addition, we require the installation of a Java Development Kit (JDK) at version (at least) 11 and a Gradle 7 installation. A pre-installed integrated development environment (IDE), such as IntelliJ[1], is recommended for conducting the tutorial but is optional.

C. *Adaptations for a Potential Virtual Environment*

The tutorial can also be online or hybrid. For remote sessions, we will prepare a Zoom room (unless otherwise specified by the organizers). We will also provide the necessary sources online before the tutorial. This additionally allows participants to test whether the setup works for them ahead of time.

References

[1] B. Selic, "The Pragmatics of Model-Driven Development," *IEEE software*, vol. 20, no. 5, pp. 19–25, 2003.

[2] O. M. Group, "Systems Modeling Language (SysML®) v2 Request For Proposal (RFP)," 2017.

[3] A. Butting, J. Michael, and B. Rumpe, "Language Composition via Kind-Typed Symbol Tables," *Journal of Object Technology (JOT)*, vol. 21, pp. 4:1–13, October 2022.

[4] A. Butting, R. Eikermann, K. Hölldobler, N. Jansen, B. Rumpe, and A. Wortmann, "A Library of Literals, Expressions, Types, and Statements for Compositional Language Design," *Journal of Object Technology (JOT)*, vol. 19, pp. 3:1–16, October 2020.

[5] S. Erdweg, T. v. d. Storm, M. Völter, M. Boersma, R. Bosman, W. R. Cook, A. Gerritsen, A. Hulshout, S. Kelly, A. Loh, *et al.*, "The State of the Art in Language Workbenches," in *International Conference on Software Language Engineering*, pp. 197–217, Springer, 2013.

[6] K. Hölldobler, O. Kautz, and B. Rumpe, *MontiCore Language Workbench and Library Handbook: Edition 2021.* Aachener Informatik-Berichte, Software Engineering, Band 48, Shaker Verlag, May 2021.

[1]https://www.jetbrains.com/idea/