2017 IEEE International Conference on Software Architecture Workshops

Language Engineering with the GEMOC Studio

Olivier Barais, Benoit Combemale
INRIA and University of Rennes 1
firstname.lastname@irisa.fr

Andreas Wortmann
University of Rennes 1 and
RWTH Aachen University
wortmann@se-rwth.de

***Abstract*—This tutorial provides a practical approach for developing and integrating various Domain-Specific (modeling) Languages (DSLs) used in the development of modern complex software-intensive systems, with the main objective to support abstraction and separation of concerns. The tutorial leverages on the tooling provided by the GEMOC studio to present the various facilities offered by the Eclipse platform (incl., EMF/Ecore, Xtext, Sirius) and introduces the advanced features to extend a DSL with a well-defined execution semantics, possibly including formal concurrency constraints and coordination patterns. From such a specification, we demonstrate the ability of the studio to automatically support model execution, graphical animation, omniscient debugging, concurrency analysis and concurrent execution of heterogeneous models. The tutorial is composed of both lectures and hands-on sessions. Hands-on sessions allow participants to experiment on a concrete use case of an architecture description language used to coordinate heterogeneous behavioral and structural components.**

I. Tutorial Topic

The development and evolution of an advanced modeling environment for a Domain-Specific (modeling) Language (DSL) is a tedious task, which becomes recurrent with the increasing number of DSLs involved in the development and management of complex software-intensive systems. This proliferation of DSLs is illustrated in a study on the industrial application of Architecture Description Languages (ADLs) that found over 120 different languages [1]. Recent efforts in language workbenches result in advanced frameworks that support software engineers in the design and implementation of DSLs. In particular, language workbenches such as Xtext[1] or Sirius[2] automatically provide syntactic tooling such as advanced textual and graphical editors respectively. However, defining and coordinating the execution semantics of DSLs, and developing the associated tooling, remains mostly hand crafted. Similarly to editors that share code completion or syntax highlighting, the development of advanced debuggers, animators, co-simulation tools and others execution analysis tools shares common facilities, which are highly sought by (architecture) modeling practitioners [2], should be reused among various DSLs.

In this tutorial, we introduce the facilities provided by the GEMOC studio[3] to extend a DSL with a well-defined execution semantics, possibly including formal concurrency constraints and coordination patterns. From such a specification, we demonstrate the ability of the studio to automatically support model execution, graphical animation, omniscient debugging, concurrency analysis and concurrent execution of heterogeneous models.

The GEMOC Studio is an Eclipse package atop the Eclipse Modeling Framework (EMF), which includes:

- *The GEMOC Language Workbench*: to be used by language designers to build and compose new DSLs,
- *The GEMOC Modeling Workbench*: to be used by domain designers to create, execute and coordinate, possibly heterogeneous, models.

The different concerns of a DSL, as defined with the tools of the language workbench, are automatically deployed into the modeling workbench. They parametrize a generic execution framework that provides various generic services, such as graphical animation, debugging tools, trace and event managers, timeline visualizations, etc.

More generally, the GEMOC studio is intended to federate the research results regarding the support of the coordinated use of various DSLs that will lead to the concept of the globalization of DSLs [3], that is, the use of multiple DSLs to support the socio-technical coordination required in systems and software engineering. As such, the project is intended to develop techniques, frameworks, and environments to facilitate the creation, integration, and automated processing of heterogeneous DSLs.

II. Tutorial Implementation

The proper duration for the proposed tutorial, including lectures and hands-on sessions, is half-day. The tutorial will be organized in three consecutive parts: the first one will introduce the basics of DSL development to create a tool-supported DSL supporting model simulation, graphical animation and omniscient debugging. The second session will be an hands-on session where the participants can experiment on a concrete use case. Finally the last part will introduce advanced features for weaving concurrency constraints in language semantics definition, and leverage them for concurrency analysis and concurrent execution of, possibly heterogeneous, models.

Part 1: Breathe Life Into Your Metamodel to Support Model Simulation, Animation and Debugging

Domain-specific models are used in the development processes to reason and assess specific properties over the system under development as early as possible. This usually leads to a better and cheaper design as more alternatives can be

[1] https://www.eclipse.org/Xtext/
[2] https://www.eclipse.org/sirius
[3] http://gemoc.org/studio

978-1-5090-4793-2/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSAW.2017.61

explored. While many models only represent structural aspects of systems, a large amount express behavioral aspects of the same systems. Behavioral models are used in various areas (e.g., enterprise architecture, software and systems engineering, scientific modeling, etc.), with very different underlying formalisms (e.g., business processes, orchestrations, functional chains, scenarios, protocoles, activity or state diagram, etc.).

To ensure that a behavioral model is correct with regard to its intended behavior, early dynamic validation and verification (V&V) techniques are required, such as simulation, debugging, model checking or runtime verification. In particular, debugging is a common dynamic facility to observe and control an execution in order to better understand a behavior or to look for the cause of a defect. Standard (stepwise) debugging only provides facilities to pause and step forward during an execution, hence requiring developers to restart from the beginning to give a second look at a state of interest. Omniscient debugging extends stepwise debugging by relying on execution traces to enable free traversal of the states reached by a model, thereby allowing designers to "go back in time".

Debugging, and all dynamic V&V techniques in general, require models to be executable, which can be achieved by defining the execution semantics of modeling languages (i.e., DSLs) used to describe them. The execution semantics of a modeling language can be implemented either as a compiler or as an interpreter to be applied on models conforming to the modeling language.

To drastically reduce the development cost of domain-specific animators and omniscient debuggers, the GEMOC studio provides a tool-supported approach to complement a modeling language with an execution semantics, and automatically get an advanced and extensible environment for model execution supporting graphical animation and omniscient debugging [4].

Part 2: Do-It-Yourself

During the second part of the tutorial, the participants will develop a simplified version of the MontiArc component & connector ADL. The ADL is tailored for designing component-based distributed interactive systems that reliy on state-based models to describe component behavior [5]. During the hands-on session, participants will use the GEMOC Studio to develop a simplified version of MontiArc, including syntax and semantics, and deploy it in the modeling workbench to edit, execute, animate and debug conforming models. In the modeling workbench, the participants will design a software architecture based on predefined components. Based on such a design, participants will be able to concurrently execute the various components according to the execution semantics (message passing) of MontiArc, to graphically animate the architecture, and to debug the system behavior.

Part 3: Weave Domain-Specific Concurrency Constraints Into A DSL!

The emergence of modern concurrent systems (e.g., Cyber-Physical Systems and Internet of Things) and highly-parallel platforms (e.g., many-core, GPGPU, and distributed platforms) call for Domain-Specific (modeling) Languages (DSLs) where concurrency is of paramount importance. Such DSLs are intended to propose constructs with rich concurrency semantics, which allow system designers to precisely define and analyze system behaviors. However, implementing the execution semantics of such DSLs is a particularly difficult task. Most of the time the concurrency model remains implicit and ad-hoc in the language design and implementation. In the language design, the concurrency model is usually implicitly inherited from the concurrency model of the meta-language employed to design the behavioral semantics (e.g., the default concurrency model of Xtend/Java), or customized with low-level primitives (e.g., using Threads). In the language implementation, the concurrency model is mostly embedded in the underlying execution environment (e.g., the concurrency model of the JVM).

The lack of an explicit concurrency model in language specifications prevents: the precise definition, the variation and the complete understanding of the DSL's semantics, the effective usage of concurrency-aware analysis techniques, and the exploitation of the concurrency model during the system refinement (e.g., during its allocation on a specific platform).

During the last part of the tutorial, we introduce the facilities provided by the GEMOC studio to extend an executable modeling language with a formal model of concurrency to leverage on the scheduling properties [6], and offer concurrency analysis and concurrent execution of, possibly heterogeneous, models [7].

Conclusion and Perspectives (15min)

A conclusion will be given more from a scientific point of view, drawing a big picture of the various breakthroughs achieved to elaborate the current version of the GEMOC studio (incl., executable metamodeling, language modularity and composition, cross-fertilization of concurrency theory and language engineering, etc.)

The tutorial ends with a presentation of the perspectives in terms of a research and technical road-map. Participants will be ask to discuss and improve this road-map.

III. Tutorial Presenters

Olivier Barais[4] is Full Professor at the University of Rennes 1, member of the DiverSE INRIA research team. He passes a PhD in computer science from the University of Lille 1, France in 2005. His research interests include Component Based Software Design, Model-Driven Engineering and Aspect Oriented Modeling. Olivier Barais has co-authored 12 journals, 55 international conference papers, 2 book chapters and 35 workshop papers in conferences and journals such as SoSyM, IEEE Computer, ICSE, ASE, MoDELS, SPLC and CBSE. Olivier Barais is also information technology enthusiast. He enjoys trying new frameworks in particular in MDE and CBSE community and playing with open-source hardware projects.

[4]http://olivier.barais.fr

Olivier Barais has been involved in the development of the Gemoc Studio. Olivier Barais has used to provide various tutorials to Comparch, Middleware, Models.

Benoit Combemale[5] is Associate Professor at University of Rennes 1. He is evolving within the research team DiverSE, joint to the IRISA and Inria labs. His research interests belong to software engineering, including model driven software engineering (MDE), software language engineering (SLE) and software validation & verification (V&V); mostly in the context of (smart) cyber-physical systems and Internet of things. Benoit Combemale co-authored 3 books, and more than 80 journal and conference publications in the fields of MDE, SLE and V&V. He is a member of the Steering Committee of the SLE conference, and the Editorial Boards of the international journals SoSyM (Springer), COMLAN (Elsevier), and SCP (Elsevier). He has been the program co-chair of SLE 2014, and general co-chair of MODELS 2016 and SLE 2017. He also used to serve as program committee member for various conferences and workshops in software engineering. Benoit Combemale is also very active in setting up and participating to satellite events of flagship conferences, including organizing workshops, tutorials and panels. He is a founding member of the GEMOC initiative.

Andreas Wortmann[6] is postdoctoral researcher at University of Rennes 1 and member of the DiverSE INRIA research team. He received his PhD from the RWTH Aachen University in 2016. His research interests include Component Based Software Design, Model-Driven Engineering, Architecture Description Languages, Software Language Engineering, and their application to cyber-physical systems. He co-authored 3 books as well as over than 30 publications on MDE, ADLs, and SLE and has served as program committee member for various conferences and workshops. He is a member of the GEMOC initiative and the IEEE robotics & automation society.

IV. Scope of the Tutorial

The intended audience is both software engineers/researchers and PhD students. The attendants must have a laptop with Virtual Box installed or a laptop with a JDK 1.8. The attendants must be familiar with Java or any Object Oriented Technology. Academics and practitioners alike will benefit from the tutorial. We expect that the presentation is of particular interest for language designers, both from academia and industry, who would like to have abstractions to deal with complex systems.

V. Tutorial Background

The tutorial is new, and has never been offered previously. For the first time, the tutorial will cover the various features of the GEMOC studio from a user perspective. The tutorial ends on the past and future challenges addressed by the community working on the GEMOC studio.

[5]http://combemale.fr

[6]http://www.andreaswortmann.de

VI. Conclusion

Languages workbenches facilitate the development of DSLs, especially for providing syntactic services. However *runtime* services are mostly handcrafted for each different DSL and each different metaprogramming approach.

Based on a common API, we proposed a framework to integrate any kind of metaprogramming approach used to define discrete-event operational semantics into an execution engine. Notably, implementing this API allows to use and reuse of generic or user-defined runtime services as *addons* that send and receive generic messages to and from the execution engines.

As our project is open-source and available online, we are very open to any contributors for implementing additional execution engines (e.g., to support operational semantics defined with other metaprogramming approaches) and additional runtime services.

References

[1] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What Industry Needs from Architectural Languages: A Survey," *Software Engineering, IEEE Transactions on*, vol. 39, no. 6, pp. 869–891, 2013.

[2] P. Lago, I. Malavolta, H. Muccini, P. Pelliccione, and A. Tang, "The road ahead for architectural languages," *IEEE Software*, vol. 32, no. 1, pp. 98–105, 2015.

[3] B. Combemale, J. Deantoni, B. Baudry, R. France, J.-M. Jézéquel, and J. Gray, "Globalizing Modeling Languages," *Computer*, pp. 68–71, Jun. 2014. [Online]. Available: http://hal.inria.fr/hal-00994551

[4] E. Bousse, T. Degueule, D. Vojtisek, T. Mayerhofer, J. Deantoni, and B. Combemale, "Execution Framework of the GEMOC Studio (Tool Demo)," in *Proceedings of the 9th ACM SIGPLAN International Conference on Software Language Engineering (SLE 2016)*, ser. SLE 2016, Amsterdam, Netherlands, Oct. 2016, p. 8. [Online]. Available: https://hal.inria.fr/hal-01355391

[5] R. Heim, B. Rumpe, and A. Wortmann, "Extending Architecture Description Languages With Exchangeable Component Behavior Languages," in *Conference on Software Engineering & Knowledge Engineering (SEKE'16)*. KSI Research Inc., Fredericton, Canada, 2016, pp. 1–6.

[6] B. Combemale, J. Deantoni, M. Vara Larsen, F. Mallet, O. Barais, B. Baudry, and R. France, "Reifying Concurrency for Executable Metamodeling," ser. Lecture Notes in Computer Science, R. F. P. Martin Erwig and E. van Wyk, Eds. Indianapolis, États-Unis: Springer-Verlag, 2013. [Online]. Available: http://hal.inria.fr/hal-00850770

[7] M. E. Vara Larsen, J. Deantoni, B. Combemale, and F. Mallet, "A Behavioral Coordination Operator Language (BCOoL)," in *18th International Conference on Model Driven Engineering Languages and Systems (MODELS 2015)*, Aug. 2015. [Online]. Available: https://hal.inria.fr/hal-01182773