Agnes Koschmider, Judith Michael, Bernhard Thalheim (Hrsg.): EMISA Workshop 2020,
CEUR-WS.org Proceedings 19

The RMT Approach: A Systematic Approach to the Development of DSML with Integrated Simulation Based on Petri Nets

David Mosteller, Michael Haustermann, Daniel Moldt, Dennis Schmitz[1]

Abstract:

The application of domain-specific models requires appropriate tool support for modeling and execution. The development of DSML languages is generally costly, especially with regard to execution semantics. The RMT approach (RENEW Meta-Modeling and Transformation) follows a systematic model-driven development process for the provision of modeling tools and also for the integration of semantics. It applies transformational semantics using Petri net formalisms as target languages in order to produce quick results for the development of modeling techniques. In order to visualize simulation events in the executed DSML model, annotations within the metamodels facilitate the necessary link between the domain-specific model and the generated Petri net. This paper gives an overview of the artifacts that are employed in the RMT approach and documents the structured process for DSML development.

Keywords: Meta-Modeling; Petri Nets; Reference Nets; Simulation; Graphical Feedback

1 Challenge of DSMLs: Animation and Simulation

As Meta-Modeling is used to provide new domain-specific modeling languages (DSML) several tools provide tool support to build corresponding models. The definition of semantics for domain-specific modeling languages (DSML) and the provision of appropriate execution tools are two of the key challenges in model-driven tool development. Defining the semantics for a DSML by transforming it into an existing language with a well-defined semantics facilitates fast results and is well suited for a prototypical approach. Bryant et al. identified "the mapping of execution results (e.g., error messages, debugging traces) back into the DSML in a meaningful manner, such that the domain expert using the modeling language understands the result" [Br11, p. 228] as one of the challenges for the translation semantics approach. Concerning the user experience, meaningful visual representation of the domain concepts is vital for the communication between different stakeholders, especially for the domain experts who are often non-software engineers [Ab17, p. 233]. The representation of DSML in execution is still considered a challenge in tool generation in general [MC18, p. 196]. We have developed several solutions for the integrated simulation of DSML based

[1] Universität Hamburg, Fachbereich Informatik, https://www.inf.uni-hamburg.de/inst/ab/art

on transformational semantics with mappings to Petri nets. With this contribution we provide a summary of the facilities involved for DSML development with integrated simulation and a description of the DSML development process according to the RMT approach. The goal is to support language developers with a structured process that enables them to achieve fast results when developing modeling tools with integrated simulation. Parts of this contribution are summarized from our previously published result [Mo19].

2 The RMT Approach

The RMT approach (RENEW Meta-Modeling and Transformation) [MCH16] is a model-driven approach for the agile development of DSML. It follows concepts from software language engineering (SLE, [Kl08]) and enables a short development cycle to be appropriately applied in prototyping environments. The technical basis for the RMT approach is provided by the RMT framework, which builds upon the RENEW modeling und simulation environment. RENEW was originally designed as a Petri net editor and simulator and has evolved into an extensible integrated development environment (IDE) for various modeling techniques [CHM18]. The RMT framework is particularly well-suited to develop languages with simulation feedback due to its lightweight approach to SLE and the tight integration with the extensible RENEW simulation engine, which supports the propagation of simulation events.

Figure 1 gives a brief overview of the application of the RMT approach to a sample DSML development project. In this example we present the development of a BPMN modeling tool. With the RMT framework, the specification of a language and a corresponding modeling tool may be derived from a set of models, defined by the developer of a modeling technique. They are categorized accordingly to the principles of SLE and identified by individual symbols in Figure 1. A meta-model defines the structure (abstract syntax, ⛏) of the language, the concepts of its application domain, and their relations. The visual representations (concrete syntax, ✍) of the defined concepts and relations are drawn interactively within RENEW by creating a composition of RENEW's graphical primitives. Alternatively, graphical standard representations are configurable by style sheets that enable a declarative customization of various pre-defined visual attributes, such as ruling, arrow shapes, colors, etc. The syntax model is complemented with icons and a tool configuration model (⚙) that defines general properties of the resulting modeling tool, such as the file extension or the ordering of tool buttons. Additionally, within the tool configuration model, tool mappings provide the link between abstract and concrete syntax and define the connection points for constructs (ports).

The models described so far can be used to generate a tool for creating and editing DSML models (Figure 1, graphical layer), however, without an execution semantics up to this point. The latter comes with the specification of semantic Petri net components (↦).

The RMT approach utilizes transformations to Petri nets in order to provide an operational semantics for a DSML. Individual DSML constructs are mapped to semantic Petri net

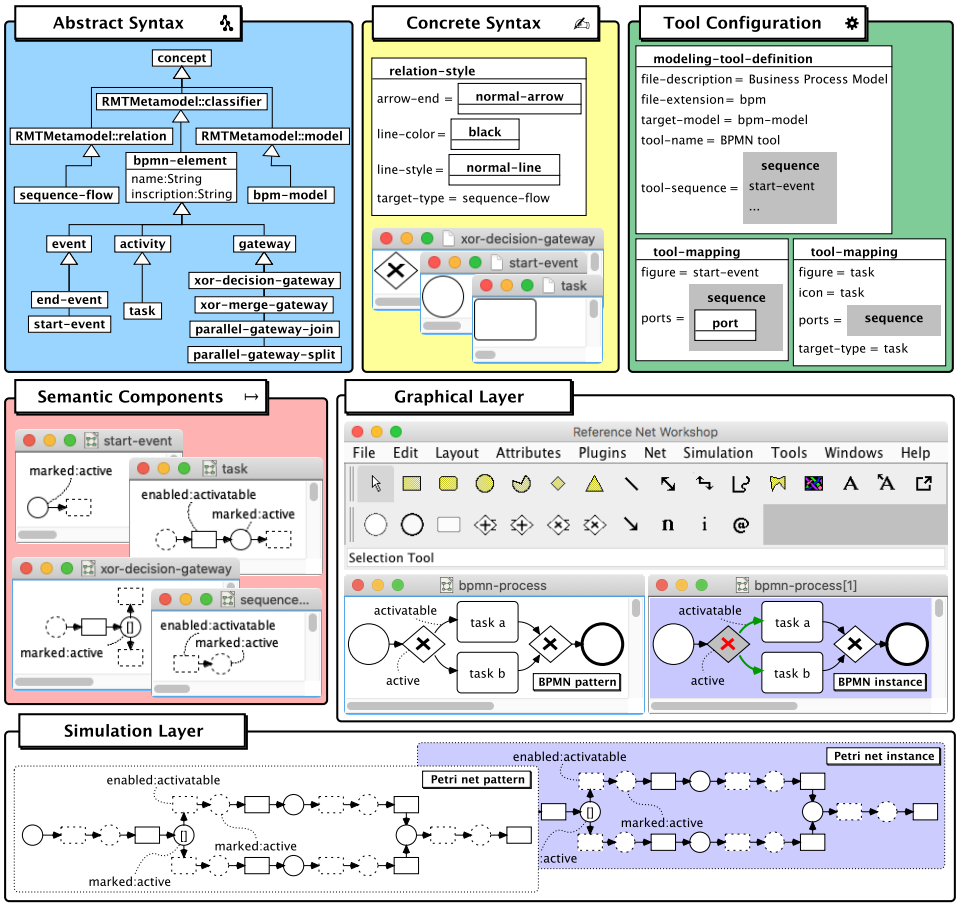

Fig. 1: A structural overview of the RMT approach and the artifacts employed

components, which results in a `1:n` mapping of DSML elements to Petri net constructs. There are several possibilities to connect the semantic components in order to facilitate the flexibility required to define semantics for various process-based modeling languages. The dotted lines around interface elements of the semantic components, as displayed in Figure 1, denote merge points. Bordering constructs are merged along these elements in the composition of semantic components; the outgoing transition of the `start-event` is merged with the incoming transition of the `sequence-flow` relation. The merge points are also visualized in the Petri net model (simulation layer), which was generated from the BPMN process model using the semantic components. The net generation from semantic components corresponds in principle to the composition of interface nets as described by Reisig [Re09].

The execution of a DSML (BPMN) model using the integrated simulation means a multi-step translation in the background, first into a Petri net pattern, which is then instantiated for execution (Petri net instance). In order to draw conclusions from the execution of the Petri net about the original (BPMN) model, a means for relating execution information with the original model is required. With RMT, this is achieved by reflecting simulation events from the underlying execution of Petri nets (target language) into the DSML (source language). The simulation visualization is based on the highlighting of model constructs as graphical feedback.

The link between the simulation events and the graphical representation is established by annotations added to the semantic and graphical components. An annotation contains two parts divided by a colon. The first part represents the simulation state or a simulation event for the particular net element, and the second part is a state concerning the whole BPMN component. In Figure 1 these annotations are depicted in the semantic components and for illustration in the generated Petri net models (net template and instance). The net elements have inscriptions, such as `marked:active`, which means that the BPMN construct is in a state `active` when the corresponding net element is marked. The component states (e.g. `active`) can be referenced in graphical components and with style sheets to specify the representation during execution. There are several ways of designing the visual behavior of the graphic components [Mo19]. Figure 1 shows "simple highlighting", which replaces the colors of the components according to a pre-defined scheme.

3 The DSML Development Process

For the provision of DSML modeling tools with direct simulation we see a major challenge in the integration into model-driven approaches in the sense that the DSML developer finds the proper support to develop these languages in a model-driven fashion. The RMT approach follows a structured development process and applies concepts from SLE in order to achieve these goals.

Figure 2 illustrates the iterative development process of the RMT approach in domain-specific BPMN notation, extended by symbols for different types of tasks according to the SLE categories (Figure 1) and icons to classify types of artifacts. The development process itself is modeled with a BPMN tool that was created using the RMT framework. It represents all necessary steps to generate a modeling and simulation tool for a domain-specific language following the RMT approach. As the process is iterative and based on the development of prototypes, it is repetitively applied in one project, however, the backward edges that return to a previous step are omitted for improved readability.

The development process starts with the creation of a RMT project, which initially consists of skeletons for the obligatory models: meta model, style sheet model and tool configuration model. This step will naturally be only performed once and it results in an initial set of artifacts that are already sufficient to generate a modeling tool using the fallback options

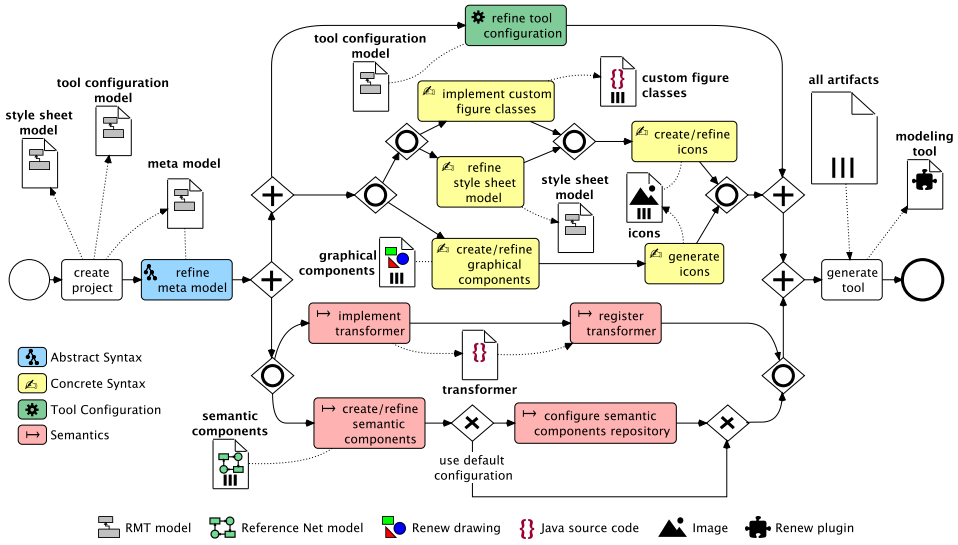

Fig. 2: RMT process for the development of DSML with execution semantics

provided by the framework. Additional artifacts are created during the process or refined in later iterations. The RMT approach utilizes model driven techniques if possible but always offers an alternative option for more flexibility if required. For example, the concrete syntax can be created by drawing graphical constructs with RENEW but there is also the alternative to adapt the representation by style sheets, or even to implement individual figures. In order to keep the effort for creating prototypes low, icon images can be generated from the graphical constructs. Of course, there is also the option to use individually customized icons. Similarly, the semantics is by default provided by a standard transformer that uses semantic components. However, a customized transformer can be tailored to the needs of individual application domains, or it can bridge to other tooling environments. For the integrated simulation the link between source model and target model is already prepared in the standard transformer. When using a customized transformer, this connection must be established by hand. Finally, all artifacts are used to generate a modeling and simulation tool, which is packaged as a Java archive (jar) to be used as a plugin for RENEW.

4 Conclusion

This contribution provides a systematic approach for the development of DSML tools with integrated simulation. The structured development process of the RMT approach shows how languages can be developed in a model-driven fashion. The RMT approach integrates the definition of semantics and the execution behavior directly into the model-driven DSML development process. With a transformation of DSML constructs into Petri net

components an operational semantics for a modeling language is provided. Annotations within the metamodels facilitate the necessary link between the domain-specific model and the generated Petri net. This enables an interactive simulation in the original representation.

Beyond the visualization of simulation results, we want to investigate the visualization of other evaluations on DSML models. For example, the results of a structural analysis (shortest paths) or a state space verification (model checking) could be integrated in the DSML representation. Again, the challenge is the integration into the structured DSML development process. In addition to the visualization of models, the control of models is also an issue, which should be considered in the RMT approach. Up to now, only rudimentary options are available for user interaction with the simulation.

References

[Ab17] Abrahão, Silvia; Bourdeleau, Francis; Cheng, Betty H. C.; Kokaly, Sahar; Paige, Richard F.; Störrle, Harald; Whittle, Jon: User Experience for Model-Driven Engineering: Challenges and Future Directions. In: 20th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, Austin, TX, USA, September 17-22, 2017. IEEE Computer Society, pp. 229–236, 2017.

[Br11] Bryant, Barrett R.; Gray, Jeff; Mernik, Marjan; Clarke, Peter J.; France, Robert B.; Karsai, Gabor: Challenges and directions in formalizing the semantics of modeling languages. Comput. Sci. Inf. Syst., 8(2):225–253, 2011.

[CHM18] Cabac, Lawrence; Haustermann, Michael; Mosteller, David: Software development with Petri nets and agents: Approach, frameworks and tool set. Sci. Comput. Program., 157:56–70, 2018.

[Kl08] Kleppe, Anneke: Software Language Engineering: Creating Domain-Specific Languages Using Metamodels. Pearson Education, December 2008.

[MC18] Mayerhofer, Tanja; Combemale, Benoit: The Tool Generation Challenge for Executable Domain-Specific Modeling Languages. In (Seidl, Martina; Zschaler, Steffen, eds): Software Technologies: Applications and Foundations. Springer International Publishing, Cham, pp. 193–199, 2018.

[MCH16] Mosteller, David; Cabac, Lawrence; Haustermann, Michael: Integrating Petri Net Semantics in a Model-Driven Approach: The Renew Meta-Modeling and Transformation Framework. Transaction on Petri Nets and Other Models of Concurrency XI, 11:92–113, 2016.

[Mo19] Mosteller, David; Haustermann, Michael; Moldt, Daniel; Schmitz, Dennis: Integrated Simulation of Domain-Specific Modeling Languages with Petri Net-Based Transformational Semantics. Transactions on Petri Nets and Other Models of Concurrency, 14:101–125, 2019.

[Re09] Reisig, Wolfgang: Simple Composition of Nets. In (Franceschinis, Giuliana; Wolf, Karsten, eds): Applications and Theory of Petri Nets, 30th International Conference, PETRI NETS 2009, Paris, France, June 22-26, 2009. Proceedings. volume 5606 of Lecture Notes in Computer Science. Springer, pp. 23–42, 2009.