Generating Domain-Specific Property Languages with ProMoBox: application to interactive systems

Bart Meyers
Modeling, Simulation and Design Lab (MSDL),
University of Antwerp
Antwerp, Belgium
firstname.lastname@uantwerp.be

Romuald Deshayes, Tom Mens
Département d'Informatique,
Université de Mons
Mons, Belgiumy
firstname.lastname@umons.ac.be

Hans Vangheluwe
Modeling, Simulation and Design Lab (MSDL),
University of Antwerp / McGill University
Antwerp, Belgium / Montréal, Canada
firstname.lastname@uantwerp.be

ABSTRACT
Domain-Specific Modeling allows domain experts with limited technical background to precisely model applications by using domain concepts. These domain-specific models can be simulated, optimized, transformed into other formalisms, and from these models executable code and documentation can be generated. Because of their syntactic simplicity they are suitable for analysis, which is nonetheless often neglected in current approaches. Especially in Human-Computer Interaction, verifying whether the model satisfies its requirements (specified as so-called properties) is essential. The *ProMoBox* approach presents a highly automated solution for the specification and verification of such properties. It provides a framework for model checking of temporal properties, where all visible artifacts (system designs, properties, simulation traces, etc.) are specified in the domain-specific way.

THE *ProMoBox* APPROACH
Domain-specific modeling (DSM) helps designing systems at a higher level of abstraction. By providing languages, "DSMLs" (defined by a metamodel), that are closer to the problem domain than to the solution domain, low-level technical details can be hidden. An essential activity in DSM is the specification and verification of properties to increase the quality of the designed systems [3]. Providing support for these tasks is therefore necessary to provide a holistic DSM experience to domain engineers. Unfortunately, this has been mostly neglected by DSM approaches. At best, support is limited to translating models to formal representations on which properties are specified and evaluated with logic-based formalisms [6], such as Linear Temporal Logic (LTL). This contradicts the DSM philosophy as domain experts desiring to specify and verify domain-specific properties are not familiar with such formalisms. We propose the *ProMoBox* framework to shift property specification and verification tasks up to the DSM level. The scope, assumptions and limitations of this approach are presented in [5].

To cite this article use the following information:

Bart Meyers, Romuald Deshayes, Tom Mens, and Hans Vangheluwe. 2015. Generating Domain-Specific Property Languages with ProMoBox: application to interactive systems. In *Proceedings of the Workshop on Formal Methods in Human Computer Interaction* (FoMHCI'15), 47-48.
http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:hbz:82-rwth-2015-030425

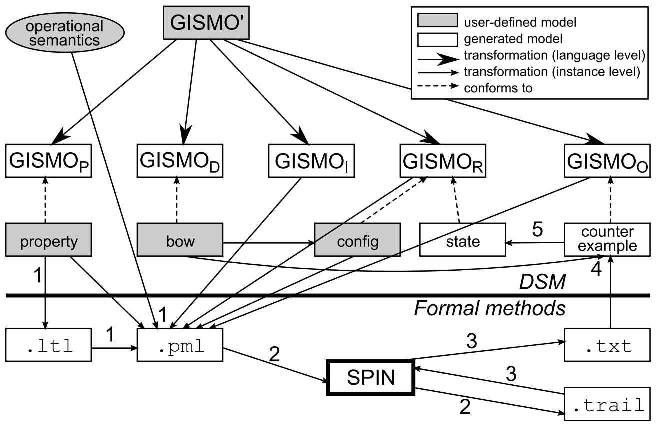

Figure 1. The *ProMoBox* approach applied to *GISMO*.

We applied *ProMoBox* to *GISMO* [1], a DSML for executable modeling of gestural interaction applications [2]. The *ProMoBox* approach for *GISMO* is illustrated in Fig. 1. The *ProMoBox* framework consists of (*i*) generic languages for modeling all artifacts that are needed for specifying and verifying properties, (*ii*) a fully automated method to specialize and integrate these generic languages in a given DSML, and (*iii*) a verification backbone based on model checking that is directly pluggable to DSM environments such as AToMPM [7]. Properties in *ProMoBox* are translated to LTL and a Promela model is generated that includes a translation of the system, its environment and its rule-based operational semantics. The Promela model is checked with the SPIN model checker [4] and if a counter-example is found it is translated back to the DSM level.

The *ProMoBox* framework [5] relies on a family of fully automatically generated modeling languages based on the DSML metamodel. These languages are required to modularly support specification and verification of model properties. The *design language* ($GISMO_D$ in Fig. 1) allows DSM engineers to design the static structure of the system. The *runtime lan-*

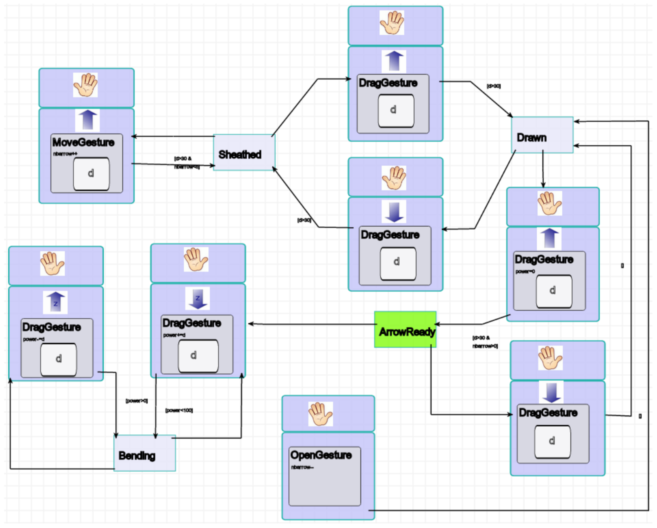

Figure 2. A bow model in state *ArrowReady* (highlighted) conform to *GIS MO$_R$*.

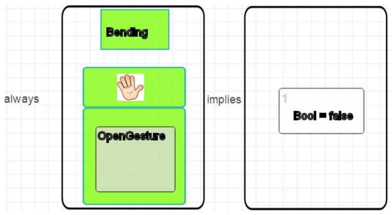

Figure 3. Property: when you fire the bow, there is no arrow left.

guage (*GIS MO$_R$*) enables modelers to define a state of the system, *e.g.,* an initial state as input of a simulation, or a particular "snapshot" during runtime (as shown in Fig. 2). The *input language* (*GIS MO$_I$*) lets the DSM engineer model the behavior of the system environment, *e.g.,* by modeling an input scenario as an ordered sequence of events containing one or more input elements. The *output language* (*GIS MO$_O$*) can be used to represent execution traces (expressed as ordered sequences of states and transitions) of a simulation or to show verification results in the form of a counter-example. Output models can also be created manually as part of an oracle for a test case. The *property language* (*GIS MO$_P$*) can be used to express properties based on modal temporal logic, including structural logic and quantification. A property is shown in Fig. 3.

Maintaining five DSMLs instead of one unacceptably increases the maintenance cost. Therefore, a fully automated method specializes and integrates these languages to any given DSML, thus minimizing the effort of the language engineer. This is realized by manually annotating the DSML metamodel entities (classes, associations and attributes) with the necessary UML-like stereotypes. This annotated metamodel (GISMO' in Fig. 1) contains all information needed to generate the five sublanguages, by merging a tailored version of the metamodel with a fixed template containing generic language constructs.

For fifteen properties, we verified whether the model shown in Fig. 2 satisfies them. The above properties are transformed to LTL, and are inserted in Promela code consisting of the system shown in Fig. 2 with initial state, the environment and rule-based model of the DSML's semantics as shown in step 1 of Fig. 1. In step 2, SPIN verifies whether the system satisfies the formula, returning "True" if it does. If there is a counter-example, steps 3 to 5 are followed: the counter-example trace is played back by SPIN, and a readable trace is printed (step 3), this trace is converted automatically to the counter-example output model (step 4), and this counter-example can be played out state by state by showing a runtime model for each state (step 5).

Because of these counter-examples, we were able to find and fix an error in our bow model of Fig. 2. In another instance, we were able to find and correct an error in one of the semantics model's rules. The performance in terms of time and memory consumption is good: evaluation never takes more than a second on an average laptop, and never requires more than 100 MB of memory.

The limitations of the framework are related to the mapping to Promela as explained in [5]. In its current state, *ProMoBox* does not allow dynamic structure models. Because of the nature of Promela, boundedness is ensured in the translation. Other constraints can be circumvented by abstracting the metamodel to make it suitable for model checking.

REFERENCES

1. R. Deshayes. 2013. A domain-specific modeling approach for gestural interaction. In *Visual Languages and Human-Centric Computing (VL/HCC)*. 181–182. DOI: http://dx.doi.org/10.1109/VLHCC.2013.6645275
2. Romuald Deshayes, Bart Meyers, Tom Mens, and Hans Vangheluwe. 2014. ProMoBox in Practice : A Case Study on the GISMO Domain-Specific Modelling Language. In *Proceedings of the 8th Workshop on Multi-Paradigm Modeling, MPM@MODELS 2014*. 21–30. http://ceur-ws.org/Vol-1237/paper3.pdf
3. Robert France and Bernhard Rumpe. 2007. Model-driven Development of Complex Software: A Research Roadmap. In *2007 Future of Software Engineering (FOSE '07)*. IEEE Computer Society, Washington, DC, USA, 37–54. DOI: http://dx.doi.org/10.1109/FOSE.2007.14
4. Gerard J. Holzmann. 1997. The Model Checker SPIN. *IEEE Trans. Softw. Eng.* 23, 5 (May 1997), 279–295. DOI: http://dx.doi.org/10.1109/32.588521
5. Bart Meyers, Romuald Deshayes, Levi Lucio, Eugene Syriani, Hans Vangheluwe, and Manuel Wimmer. 2014. ProMoBox: A Framework for Generating Domain-Specific Property Languages. In *Software Language Engineering*. Lecture Notes in Computer Science, Vol. 8706. Springer International Publishing, 1–20. DOI: http://dx.doi.org/10.1007/978-3-319-11245-9_1
6. Matteo Risoldi. 2010. *A methodology for the development of complex domain-specific languages*. Ph.D. Dissertation. University of Geneva. http://archive-ouverte.unige.ch/unige:11842
7. Eugene Syriani, Hans Vangheluwe, Raphael Mannadiar, Conner Hansen, Simon Van Mierlo, and Hüseyin Ergin. 2013. AToMPM: A Web-based Modeling Environment. In *MoDELS Demonstrations*. 21–25.