Domain-Specific Languages for Flexibly Experimenting with Stochastic Models

Danhua Peng*, Tom Warnke, Adelinde M. Uhrmacher

Modeling and Simulation Group, Institute of Computer Science, University of Rostock, Albert-Einstein-Straße 22, 18059 Rostock, Germany; *danhua.peng2@uni-rostock.de

Simulation Notes Europe SNE 25(2), 2015, 117 - 122
DOI: 10.11128/sne.25.tn.10299
Received: August 10, 2015 (Selected ASIM STS 2015 Postconf. Publ.); Accepted: August 15, 2015;

Abstract. Developing a model for simulation is a difficult task, in which simulation experiments play a critical role. In modeling and simulation, domain specific languages are widely used for model description. More and more efforts have been put in facilitating simulation reproducibility in recent years. This motivates the use of domain specific languages as the means to express experiment specifications.

Domain specific languages can be used to specify different tasks of simulation experiments, such as experiment configuration, observation, analysis, and evaluation of experimental results. More importantly, they can serve to specify crucial observations from experiments regarding model behavior. Therefore, with a formal description of model behavior, an evaluation based on model checking techniques can also benefit from domain specific languages.

In this paper, we will first discuss how domain specific languages can be used to specify simulation experiments and illustrate it by using the domain specific language SESSL. We aim at dealing with stochastic models. Several problems arise in specifying simulation experiments with stochastic models, such as probability estimation, tolerating stochastic noises, and robustness measurement. Domain specific languages can help handling those problems.

Introduction

Building simulation models is a complex process, which is both an art and a science. To ensure that models are created at the appropriate level of abstraction and are valid regarding certain questions of interest, the design and execution of simulation experiments is essential. On the one hand, the reproducibility of simulation experiment results is or should be a basic requirement for model publication.

On the other hand, in the model development process, the model may be revised iteratively. After the model revision, it may be necessary to repeat the simulation experiments conducted with previous versions of model. Besides, when new models are built based on this model, those simulation experiments can provide useful information to assist experimentation with the new models as well [1].

Therefore, it is of significance to describe simulation experiments so that they can be easily reproduced. To enable this, an unambiguous, explicit experiment description is important. All the aspects that define the simulation experiment should be recorded completely and accurately, including the conditions and the results generated from the experiments.

The advantages of domain specific languages for model design are well-known. They enable domain experts to build models using the vocabulary of the domain, while hiding implementation details. For example, languages for cell biological models such as ML-Rules [2] adopt a rule-based modeling style that resembles biochemical reaction equations, whereas the object-oriented style of Modelica [3] can easily be mapped to components of technical systems.

However, domain specific languages can not only be used to create models, but also to support flexibly experimentation with models. In modeling and simulation, there is a trend that treats the experimentation process as a first class object, e.g., in [4] and [5], where several individual tasks can be distinguished in this process, such as configuration, data collection, analysis, and evaluation.

In this paper, we will first present the domain specific language SESSL (Simulation Experiment Specification via a Scala Layer) [6]. As our focus so far has been on experiments with stochastic models, we will discuss the problems and challenges in dealing with stochasticity and how they can be handled by extensions of SESSL.

1 Domain Specific Languages in Experiment Specification

1.1 Domain Specific Languages

A domain-specific language (DSL) is a programming language that is targeted specifically at an application domain, in contrast to a general purpose language. It contains syntax and semantics that represent the concept at the same level of abstraction that the application domain offers [7]. According to [8], a DSL is small and declarative, and offers expressive power focused on and usually restricted to a particular problem domain through appropriate notations and abstractions.

Typically, two types of domain specific languages are distinguished: internal (or embedded) DSL and external DSL. An embedded DSL is implemented based on a general-purpose programming language, i.e., the host language, as an embedding. It inherits the constructs of its host language and adds domain-specific primitives to provide the user a suitable modeling abstraction. However, its use requires typically some knowledge of the host language. The advantage of internal domain specific languages is that less implementation effort is required for designing. More importantly, they can easily be extended.

An external domain specific language, in contrast to internal domain specific languages, is developed as an independent language, which requires separate interpretation or compilation. They are designed ground-up; therefore their development has more freedom without constraints from the host language. In modeling and simulation, both types of domain specific languages are used [9].

1.2 Specifying simulation experiments

One goal of specifying simulation experiments is to allow reproducibility of the experiment and their exchange among different scientific groups. For that, one has to identify what kind of information is required to reproduce experiment results.

Several work exists on identifying requirements in describing simulation experiments, such as Minimum Information About a Simulation Experiment (MIASE) [10] and Minimum Simulation Reporting Requirements (MSRR) [11]. More detailed information can be found in [12].

These standards define guidelines in providing an accurate and complete description of simulation experiments. As identified by MIASE, to make the description of simulation experiments available to third parties, it must contain: the models to be simulated and their configuration parameters, the simulation configuration such as simulator to be used, the post-processing on the raw numerical results and the description of the final output results [10]. Simulation experiments can be interpreted as a process that comprises different tasks. In [5], six tasks are identified in a simulation experiment: specification, configuration, simulation, data collection, analysis and evaluation.

By combining the two perspectives above, we argue that domain specific languages, as being able to allow the reproducibility of simulation experiments, can be employed to support simulation experimentation on models from different aspects: model configuration, simulation configuration, experiment execution, observation, analysis, and evaluation of results. We will illustrate this with the domain specific language SESSL.

2 SESSL

SESSL is an embedded domain-specific language for simulation experiments [6]. It exploits the feature of its host language Scala [13], such as meta-programming, to allow flexible experiment set-ups. A SESSL specification can incorporate simulation algorithm, model parameters, simulation run time, parallel execution, stopping conditions, replication numbers, observation, result analysis a.s.o., as needed; however only the specification of the model file is mandatory while a default option is provided for the rest. The actual experiment is then performed with arbitrary simulation software that is controlled by SESSL based on a specific binding. Currently, a number of bindings to different simulation systems exist, such as the binding to the modeling and simulation framework JAMES II [14]; additional bindings can be added straightforwardly.

We illustrate the features of SESSL with an experiment specification as shown in Listing 1. In this example, a simulation experiment is specified based on JAMES II (line 2). This specification contains configuration of the model (line 4-6), configuration of the execution machinery (line 7-10), observation (line 11-12), evaluation of results (line 13-18) and execution (line 20).

```
1 import sessl._ // SESSL core
2 import sessl.james._ // JAMES II binding
3 val exp = new Experiment with Observation with ParallelExecution with Hypothesis {
4   model = "file-mlrj:/./SimpleModel.mlrj"
5    scan("a" <~ range(100, 50, 200), "b" <~ range(1, 4, 10), "c" <~ range(0.01, 0.2, 1))
6    set("d" <~ 10.0)
7   simulator = MLRulesTauLeaping()
8   stopCondition = AfterWallClockTime(seconds = 10) or AfterSimTime(1)
9   replications = 100
10  parallelThreads = -2
11   observe("x")
12   observeAt(range(0.0, 0.1, 10.0))
13   assume{
14         P(Peak("x", "peakHeight"), time >= 2 and time <= 4, "peakTime"),
15         E(Decrease("x", "decreaseAfterPeak"), end = 10, "afterPeak"),
16         Id("peakTime") STARTS Id("afterPeak"),
17         Id("peakHeight") >= Id("decreaseAfterPeak")
18    }
19   }
20   execute(exp)
```

Listing 1: SESSL specification of a simulation experiment based on a binding to JAMES II.

As SESSL can easily support experiment set-up and execution, other analysis such as optimization is provided as well and examples on this can be found in [6]. We moved our focus to the analysis of results. Model checking is a well-established verification technique to automatically analyze the dynamic behavior of models, based on formalizing model behavior with temporal logics, such as Linear Temporal Logic (LTL) [15]. To support this, we extended SESSL with an additional trait 'Hypothesis' in [1], which allows to specify behavior properties with LTL. Another language was proposed to describe the properties of trajectories in [16], and integrated into SESSL. As shown in Listing 1, a property is specified (line 13-18), which states that the model variable 'x' observed from experiments reaches a peak between time 2 and time 4, followed by a decrease which ends at time 10. This example shows how domain specific languages, like SESSL, are capable to support different tasks of simulation experiments. Meanwhile, with the explicit, declarative SESSL experiment specification, simulation experiments can be also reproduced.

3 Experimentation on Stochastic Models

In many areas such as systems biology, stochasticity plays an important role. Stochastic models, e.g., Continuous-Time Markov Chains (CTMC), provide a powerful means to model and to analyze the dynamics of the system of interest. When conducting experimentation with stochastic models, certain problems need to be considered.

3.1 Probability estimation

Simulation experiments with a stochastic model require multiple replications to gain the confidence on experiment results. To analyze and evaluate the experiment results, a typical question arises: what is the probability that the model shows a certain behavior? Statistical model checking [17], which is a simulation-based verification technique, has been widely used to provide answers to this question. Several statistical model checking approaches exist, such as the Bayesian approach [18] and the Sequential Probability Ratio Test [19].

To support statistical model checking, we extended SESSL to allow the definition of probabilistic statements based on Continuous Stochastic Logic and hypothesis testing [1]. The property of model behavior can be specified in either LTL or the trajectory language proposed in [16]. Listing 2 specifies that with a probability of at least 0.8, the variable 'x' shall peak between time 2 and time 4 and afterwards decrease until time 10. Using hypothesis testing, the number of required simulation replications can be determined. A corresponding number of simulation trajectories are generated, against each of which the property specification is checked. Thus, it is possible to determine whether the model satisfies the specification with a probability greater than a given threshold.

```
assume(Probability >= 0.8){
        P(Peak("x", "peakHeight"), time >= 2 and time <= 4, "peakTime"),
        E(Decrease("x", "decreaseAfterPeak"), end = 10, "afterPeak"),
        Id("peakTime") STARTS Id("afterPeak"),
        Id("peakHeight") >= Id("decreaseAfterPeak")
   }
```

Listing 2: A probability property specification in SESSL experiment: with a probability of at least 0.8, the number variable 'x' shall peak between time 2 and time 4 and afterwards decrease until time 10.

3.2 Tolerating stochastic noise

Besides the uncertainty of results among different simulation replications, stochasticity exists within one replication as well. In each trajectory produced in the simulation experiment, there may be some stochastic noises.

As shown in Figure 1, the observed model variable 'x' shows an oscillation property but with stochastic noises. In LTL, typically an oscillation behavior can be specified based on derivations, i.e.,

$F(((dx/dt > 0)\&(x > 0))\&(F((dx/dt < 0)\&(x < 100))))$.

However, with the existence of noises, the calculation of first order derivation is not applicable any longer. Alternatively, it can be expressed in LTL as

$$G(((x < 0) \rightarrow F(x > 100)) \wedge ((x > 100) \rightarrow F(x < 0)) \wedge F(x > 100)),$$

which is complex and error prone. However, it can be described with a domain specific language in a much more succinct manner by simply defining a predicate named 'Oscillation'. What's more, several parameters can be added to allow specifying constraints on the oscillation amplitude and period.

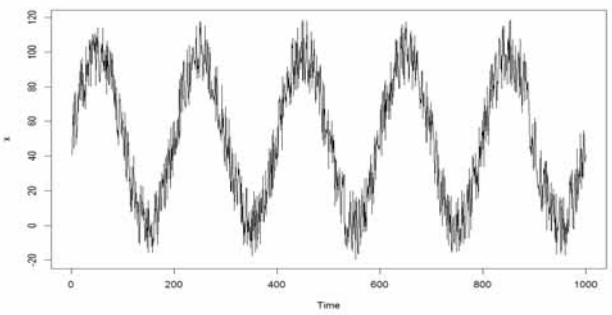

Figure 1: An example of simulation trajectory generated from experiments with a stochastic model, where the variable 'x' oscillates along the time with noises.

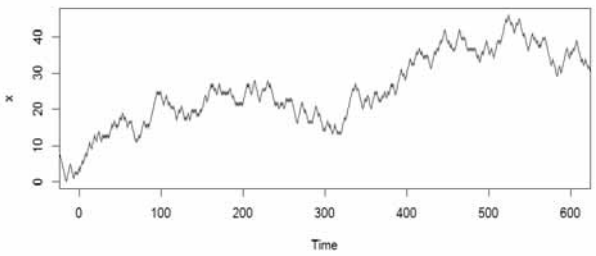

Figure 2: An example of simulation trajectory generated from experiments with a stochastic model, where the trend of x is increasing however shows some noises.

Domain specific languages can be used as an interface with easy-to-use predicates defined for the user, while those predicates can be transformed into the corresponding specification in temporal logics (depending on their semantics). In this way, the existing checking algorithm developed for temporal logics, such as [20], can be employed.

Let us look at another example. As shown in Figure 2, in this simulation trajectory generated from experiments on stochastic model, the variable 'x' evolves over time, exhibiting a trend of increase. However, because of the stochastic noises, it is not strictly increasing all the time, i.e., the first order derivation of 'x' is not always larger than zero. In this case, it is difficult to specify the increase behavior using temporal logics with noises taken into account.

On the other hand, there are different ways to tolerate the noise. In this trajectory, one can say that variable 'x' increases from time 0 to 200, or from time 0 to time 600, depending on how the noises are tolerated and how the increase is defined and interpreted.

Domain specific languages, which "speak" the language of the domain, provide the possibility to flexibly define specifications. Users can define the behavior property in different manners, which may not be expressible formally. In the trajectory language proposed in [16], as shown in Listing 1, several predicates are defined to describe properties, e.g., peaking, increase, decrease and reaching a steady state. Users can provide algorithms to check those predicates to tolerate the stochastic noises based on different requirements.

Taking the simulation trajectory shown in Figure 2 as example, a predicate can be that variable 'x' increases from time 0 to time 600.

Users can either define the semantics of increase as a simple comparison between the starting point and the ending point, where the predicate would hold. Or a more complex algorithm can be defined which checks whether the derivation of 'x' is within a certain a threshold, in which case the predicate may not hold because there is a relatively obvious decrease from time 200 to time 300 and the deviation could be out of the given threshold.

3.3 Robustness measurement

Besides probability estimation, robustness measurement is of interest in analysing stochastic systems. A general definition is that 'robustness is a property that allows a system to maintain its functions against internal and external perturbations' [21]. To formally analyze robustness, a precise definition is required and this can be performed from different perspectives.

While checking whether a behavior property holds or not provides the yes/no answer, i.e., the satisfiability, it may also be interesting and important to check to which extent the property holds, or how far it is from the property holding. Thus, the robustness degree regarding property satisfactory can be defined. There is plenty of work on this type of robustness analysis, e.g., [22], [23] and [24]. The behavior properties are first specified with temporal logics, such as LTL, Metric Temporal Logic (MTL) [25], or Signal Temporal Logic (STL) [26]. The robustness degree is measured based on definitions of distance between a simulation trajectory and the formalized properties, either in space or in time. Furthermore, for stochastic models, similar definition of robustness has been proposed in [27], where a robust satisfiability distribution for the formalized property can be estimated from that of multiple replications.

So far, this type of robustness measurement is not supported in SESSL. However, as an internal domain specific language, it is easy to extend it. Based on existing work, a specification of robustness can be added into current SESSL by defining functions and integrating corresponding robustness measurement implementations.

Additionally, another common definition of robustness is to measure the capacity of the model maintaining the given behavior with respect to changes in model parameters [28]. SESSL, as shown in Listing 1, allows specifying model parameters in a range in addition to single values (line 5-6). This provides the possibility to define robustness regarding model parameters.

4 Conclusion

In comparison to deterministic models, experiments with stochastic models require to take stochasticity into account. This may include stochastic deviations between simulation runs as well as stochastic noises during one run. Domain-specific languages for the specification of simulation experiments such as SESSL allow handling these problems. In particular, they can employ existing powerful analysis and evaluation tools without much effort, as well as integrating new ones. As domain-specific languages provide a natural, domain-friendly way to document simulation experiments, their adoption is an important step towards the reproducibility of experiments and their results.

References

[1] Peng D, Ewald R, Uhrmacher AM. Towards semantic model composition via experiments. In Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of advanced discrete simulation. *Conference on Principles of advanced discrete simulation;* 2014; ACM.151-162.

[2] Maus C, Rybacki S, Uhrmacher AM. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology*. 2011; 5(1): 166. doi:10.1186/1752-0509-5-166.

[3] Elmqvist H, Mattsson S-E. MODELICA-the next generation modeling language-an international design effort. In Proceedings of First World Congress of System Simulation. *First World Congress of System Simulation;* 1997; 1-3.

[4] Teran-Somohano A, Dayıbaş O, Yilmaz L, et al. Toward a model-driven engineering framework for reproducible simulation experiment lifecycle management. In Proceedings of the 2014 Winter Simulation Conference. *Winter Simulation Conference;* 2014; Savannah, Georgia. 2694195: IEEE Press.2726-2737.

[5] Leye S, Uhrmacher AM. GUISE-a tool for GUIding simulation experiments. In Proceedings of Winter Simulation Conference. *Winter Simulation Conference;* 2012; Berlin, Germany. Winter Simulation Conference.305.

[6] Ewald R, Uhrmacher AM. SESSL: A domain-specific language for simulation experiments. *ACM Transactions on Modeling and Computer Simulation (TOMACS).* 2014; 24(2): 11. doi:10.1145/2567895.

[7] Ghosh D. *DSLs in Action.* Manning; 2011. 351.

[8] Deursen Av, Klint P, Visser J. Domain-specific languages: an annotated bibliography. *SIGPLAN Notices.* 2000; 35(6): 26-36. doi:10.1145/352029.352035.

[9] Miller J, Han J, Hybinette M. Using domain specific language for modeling and simulation: Scalation as a case study. In Proceedings of the 2010 Winter Simulation Conference. *Winter Simulation Conference;* 2010; IEEE.741-752.

[10] Waltemath D, Adams R, Beard DA, et al. Minimum information about a simulation experiment (MIASE). *PLoS computational biology.* 2011; 7(4): e1001122_1001121-e1001122_1001124. doi:10.1371/journal.pcbi.1001122.

[11] Rahmandad H, Sterman JD. Reporting guidelines for simulation-based research in social sciences. *System Dynamics Review.* 2012; 28(4): 396-411. doi:10.1002/sdr.1481.

[12] Schutzel J, Peng D, Uhrmacher AM, et al. Perspectives on languages for specifying simulation experiments. In Proceedings of the 2014 Winter Simulation Conference. *Winter Simulation Conference;* 2014; IEEE.2836-2847. doi:10.1109/WSC.2014.7020125.

[13] Odersky M, Spoon L, Venners B. *Programming in Scala.* Second Edition. Artima Press; 2010. 852.

[14] Himmelspach J, Uhrmacher AM. Plug'n simulate. In Proceedings of the 40th Annual Simulation Symposium. *Simulation Symposium;* 2007; Norfolk, Virginia. IEEE.137-143. doi:10.1109/ANSS.2007.34

[15] Clarke EM, Grumberg O, Peled D. *Model checking.* 1999.

[16] Warnke T, Peng D, Haack F, et al. Towards a Language for Specifying Properties of Simulation Trajectories. In Proceedings of 12th International Conference on Computational Methods in Systems Biology. *Computational Methods in Systems Biology;* 2014; London.

[17] Younes HL, Kwiatkowska M, Norman G, et al. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer.* 2006; 8(3): 216-228. doi:10.1007/978-3-540-24730-2_4.

[18] Jha SK, Clarke EM, Langmead CJ, et al. A bayesian approach to model checking biological systems. In *Computational Methods in Systems Biology;* 2009; Springer.218-234.

[19] Wald A. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics.* 1945; 16(2): 117-186. doi:doi:10.1214/aoms/1177731118.

[20] Spieler D. Model checking of oscillatory and noisy periodic behavior in Markovian population models [Doctoral dissertation]. Saarland University; 2009.

[21] Kitano H. Towards a theory of biological robustness. *Molecular systems biology.* 2007; 3(1): 137. doi:10.1038/msb4100179.

[22] Rizk A, Batt G, Fages F, et al. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Computational Methods in Systems Biology;* 2008; Springer.251-268.

[23] Donzé A, Maler O. Robust satisfaction of temporal logic over real-valued signals. Springer; 2010.

[24] Fainekos GE, Pappas GJ (2006). Robustness of temporal logic specifications for finite state sequences in metric spaces, Technical Report MS-CIS-06-05, Dept. of CIS, Univ. of Pennsylvania.

[25] Koymans R. Specifying real-time properties with metric temporal logic. *Real-time systems.* 1990; 2(4): 255-299. doi:10.1007/BF01995674.

[26] Maler O, Nickovic D. Monitoring temporal properties of continuous signals. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems.* Springer; 2004. 152-166.

[27] Bartocci E, Bortolussi L, Nenzi L, et al. On the robustness of temporal properties for stochastic models. *Electronic Proceedings in Theoretical Computer Science.* 2013; 3-19. doi:10.4204/EPTCS.125.1.

[28] Komorowski M, Costa MJ, Rand DA, et al. Sensitivity, robustness, and identifiability in stochastic chemical kinetics models. *Proceedings of the National Academy of Sciences.* 2011; 108(21): 8645-8650. doi:10.1073/pnas.1015814108