Report on Workshop: Planning the Future of Agent Simulation

Fiona Polack*
Keele University
School of Computer
Science and Mathematics
f.a.c.polack@keele.ac.uk

Steffen Zschaler*
King's College London
Bush House
szschaler@acm.org

Abstract In May 2019, a workshop on principled development of future agent-based simulations was held at Keele University. Participants spanned companies and academia, and a range of domains of interest, as well as participant career stages. This report summarizes the discussions and main outcomes from this workshop.

Keywords
Agent-based simulation, complex systems, principled simulation

1 Overview of Workshop and Participants

The Workshop, Planning the Future of Agent Simulation (Keele University, UK, 29–30 May, 2019), brought together academics and industrialists working on creating, maintaining, and supporting agent-based simulations of complex systems.

The workshop's aim was to establish a vision for the future of principled modeling and simulation. As agent-based simulation is increasingly used in research on real-world complex systems, there is an associated need to create well-understood, flexible simulation software that can underpin ongoing research programs and support research in systematic and principled ways. The goal of the workshop was to share best practice, in order to identify how software engineering practices can be used to support both researchers who use or could use agent-based simulation of complex systems and software engineers who create, maintain, or redevelop existing research simulations.

Participants (a list is available on request) represented many associated communities:

- Developers of research simulations in immunological and cellular contexts: representatives of the Bentley Lab; The Francis Crick Institute, King's College London; and the former York Computational Immunology Lab.
- Researchers in smart energy and transport: Keele's SEND (ERDF/BEIS) and LiveLab (DfT/Adept) projects.
- Researchers in agent-based modeling and simulation from CUSP London.
- Software engineering researchers with expertise in model-driven engineering, simulation development, and validation.
- Commercial and academic simulation developers and consultancy and simulation support developers: Cosmo Tech, Slingshot Simulations, Simomics Ltd., and the Sheffield FLAME GPU Team.

* Corresponding authors.

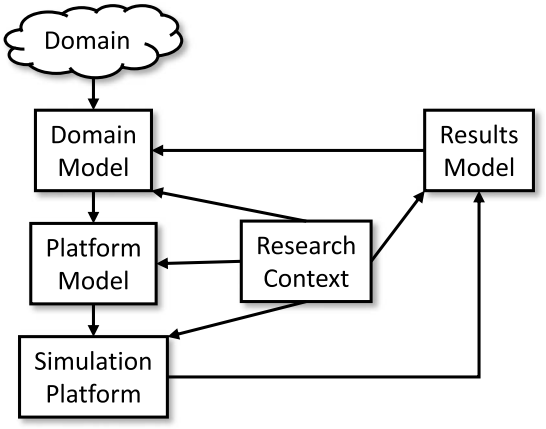

Figure 1. CoSMoS process overview.

On the first day of the workshop, participants presented their perspectives on the state of the art in simulation and simulation tools, introducing their existing work. On the application side, this included work using simulation in immunology (e.g., [1, 7, 11, 13, 15, 17]), vascular biology (e.g., [4, 5]), and synthetic biology [12], as well as in the social sciences [8] and, more generally, the evaluation of simulation results (e.g., [9, 10]). Equally, Cosmo Tech, Slingshot Simulations, and the FLAME GPU Team presented on different approaches to high-performance simulation platforms that allow for domain specialization.

On the second day, the workshop focused on establishing a shared vision of the state of the art, and identifying research opportunities.

2 A Shared Vision for Agent-Based Simulation

The workshop participants agreed on a vision for the principled development of agent-based simulations as domain-specific scientific instruments, building on top of the approach developed in the CoSMoS project[1] [16] and used in simulation development at York Computational Immunology Lab and elsewhere. The approach is commercially supported by Simomics Ltd.

2.1 The CoSMoS Approach

Figure 1 gives a high-level overview of the CoSMoS approach. The *domain* is how CoSMoS represents the subject of the simulation activity, and is typically a particular scientific lab or expert's view of the real-world context and problem. The CoSMoS approach typically starts by developing a picture of the domain to arrive at the purpose and real-world components of a potential simulation.

From the domain, a *domain model* is created: an abstracted representation of the real world that captures the essence of the problem under study. The CoSMoS domain model can be used as a means of communication between domain experts and software engineers, to ensure a shared understanding (*domain expert validation*) of the scope and purpose of the simulation activity, and of real-world mechanisms to be explored through simulation. The domain model—which may be a suite of abstract software engineering diagrams, or a less formal expression of the real-world problem—focuses on abstraction. For instance, a cell-level simulation abstracts from the detail of specific cell–cell interaction and signaling, identifying the cell agents and agent interactions that capture a computationally feasible view of the cell system of interest.

1 https://www.cosmos-research.org/

The translation from a domain model to a *platform model* represents a shift from the focus of the real-world expert to the focus of the software engineer. The platform model is normally a suite of diagrams expressed in an appropriate software-engineering language. The focus is usually on modeling the behavior of agents (cells) according to the abstraction identified in the domain model. The *simulation platform* is a realization of the platform model for a specific simulation platform: an executable simulation on which experiments can be run.

Ideally, the simulation platform and the intermediate models support full traceability (each concept in the domain is traceable to a specific aspect of the simulator, and vice versa). Furthermore, the simulation project should support flexible modification. During development, the simulation platform is calibrated against the domain and may be adjusted (typically, uncertain parameter values are tuned so that a range of real-world behavior observations can be replicated). Subsequently, the *fit-for-purpose* simulation platform may need to be modified to run related experiments. Participants also reported many instances of reuse, where a new research simulation development started from the simulation platform of an existing simulator, with effort focused on updating models to reflect the new domain, and ensuring that the fitness for purpose of the new simulator was properly assured.

Once a simulation platform is fully engineered, simulation experiments are run. Most experiments are run many times, as complex simulations are nondeterministic or stochastic: A significant advantage of simulation-based research is the ability to run the same experiment many times with controlled variation. The results of simulation are the results of the computational execution, and are presented in the CoSMoS *results model*. Simulation results are not real-world results, and the combined understanding of software engineers and domain experts (e.g., those who designed the simulation and the real-world experiments) may be needed to interpret results and draw conclusions that can be applied to the domain research.

The last of the CoSMoS top-level concepts is the *research context*. Initially, the research context states the purpose of and participants in the simulation. As modeling and development proceed, the research context collects sources, design decisions, assumptions, compromises, and so on; the entirety of the research context may be needed to appropriately understand and report the simulation results. The research context also supports the demonstration of the fitness for purpose of the simulation: For instance, an argument may be constructed showing the rationale and design decisions of the domain models. This forms an open record that can be adjusted (as understanding develops), published, or challenged (e.g., by researchers from other groups).

The CoSMoS process has been used in cell-level and immune systems research simulation: See, for instance, [1, 7, 11, 13, 15, 17]. It proposes a principled approach that supports a close collaboration between domain experts and software engineers. There is a strong focus on fitness for purpose, with decisions and assumptions recorded and, often, argumentation used to express belief in the appropriateness of models or implementations steps. The approach has recently been reexpressed as a catalog of patterns [16]. Sophisticated simulation analysis and support tools are provided by Alden's Spartan and Aspasia tools[2] [2, 6], including support for calibration, experimentation, and data analysis. In addition, tools and commercial consultancy are provided by Simomics[3] Virtual Lab. The workshop saw demonstrations of Simomics Reason, supporting the argumentation approach devised by Polack, Alden, Andrews, et al. [3], and Simomics Evidence, which supports annotation of models and code with links to supporting material from the research context.

2.2 The Vision for Simulation

A CoSMoS-style simulation development typically uses many models (images), ranging from domain-typical informal sketches to diagrams conforming to well-defined modeling or implementation languages. However, there is currently no automation of the development process: Typically, mappings between models and to code are ad hoc or by manual transformation. The informal development process reduces confidence that the implementation adequately captures the intention of the

2 http://www.kieranalden.info/index.php

3 https://www.simomics.com/

domain models and the domain experts. CoSMoS researchers have long noted, but not yet realized, that *model-driven engineering* (MDE) offers an automatable and validatable approach to implementation, and that the use of *domain-specific modeling languages* (DSMLs) can facilitate domain modeling as well as underpinning such automation [14, 16]. The Workshop identified some conditions for moving in this direction, as follows:

1. Focus support specifically on agent-based models and simulations rather than other approaches to simulation-based research.
2. Develop a hierarchy of DSMLs and corresponding model transformations (in effect, model compilers) from languages that facilitate expression of *domain models*—for specific, fairly narrow domains—via models that focus successively on more technical software engineering languages to a *platform-model* DSML that can be directly compiled or interpreted as a *simulation platform*.
3. Develop a corresponding hierarchy of DSMLs and model transformations from languages that allow capturing research questions, hypotheses, and experiments at the domain-model level, targeting experimental activities: that is, the scripting of simulation runs and the extraction of relevant data from simulations. The Workshop showed that experimental design DSML families can take advantage of existing statistical and analysis tools such as Spartan or MC^{2}MABS.[4]

As a starting position, we assume that domain models can be adequately expressed in a well-founded DSML (and thus that the initial stages; of the CoSMoS process—which establish the abstraction level and purpose, capturing existing knowledge—have been completed). MDE supports transformation between models; the transformations themselves form a transformation model, expressed in well-defined, tool-supported transformation languages such as ETL[5] or ATL.[6] MDE supports rule-based validation of well-defined source models and transformation models, and thus increases confidence that target models are faithful to source models.

Once full transformation (from domain model to code) is achieved, we can return to the initial CoSMoS stage, to explore how the representations used to capture the concepts of the domain itself can be integrated. For instance, it may be possible to integrate standardized domain languages such as SBML (for biological modeling), when combined with related metadata, into the transformation chain.

An initial graphical rendering of this vision is shown in Figure 2. Here, the boxes on the outside align with the CoSMoS process. The central part shows an initial outline of the three hierarchies of increasingly domain-specific modeling languages and automated transformation chains that allow one to automatically bridge the gap from domain model to simulation in an inspectable and replicable manner. Using a hierarchy of modeling languages reduces the effort required for each new simulation by allowing reuse at the right level of abstraction.

3 Key Challenges

The workshop identified key challenges in making this vision a reality:

1. Fitness for purpose of simulations relies on creating domain models that the domain experts can understand and validate. There may then be a need to integrate languages (and models in these languages) that are amenable to different expert contributors. From these DSMLs, there needs to be a consistent mapping to complex simulation implementation: Currently, there are no specialized "agent modeling" languages (though some ABM tools

4 https://sites.google.com/site/herdbenjamin/mc2mabs
5 https://www.eclipse.org/epsilon/doc/etl
6 https://www.eclipse.org/atl/

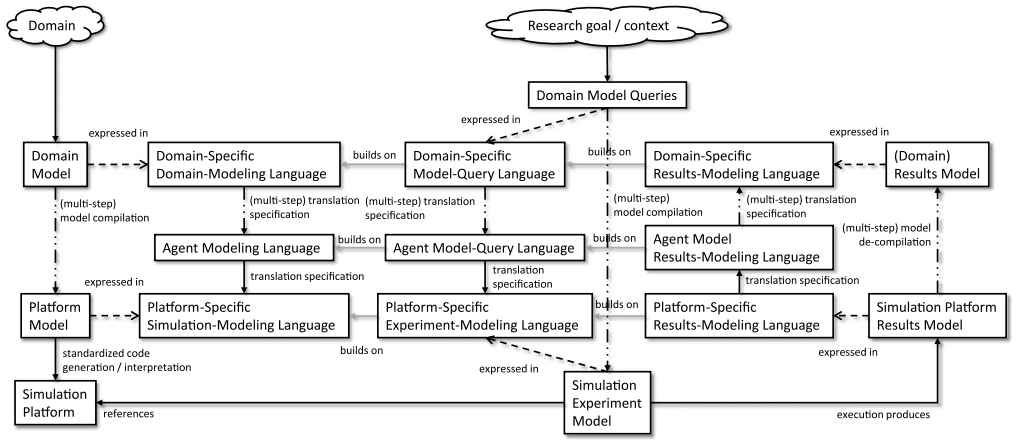

Figure 2. A vision for model-driven principled engineering of agent-based simulations.

exist). Ideally, we need standardized language(s) for simulation development activities and domains, and integrated languages to support different levels of expertise (through to coding or modeling experts) as well as different domain expertise; we anticipate needing intermediate (bridging) languages.

2. The research use of complex systems simulations requires flexible, maintainable models: approaches to modeling and DSMLs that support simulations that can be evolved, changed, modified, linked, and layered; approaches that support multi-scale modeling and simulation; approaches that are principled and support fitness-for-purpose statements, revision, and analysis; and support for decoupled or distributed development activities (e.g., models developed separately from simulators).
3. There is no single common language to support identification, design, and realization of simulation experiments, and to manage simulation runs and results: Tools such as Spartan, Aspasia, and MC^{2}MABS each have their own interfaces and interfacing languages.
4. Working with suites of DSMLs and transformations raises software engineering issues such as the maintenance of DSMLs for currency and interoperability, extension to new domains, and inclusion of new target platforms.
5. Given appropriate DSMLs and transformations using appropriately standardized languages, there is a need to facilitate effective development support, such as smart tools that map different component concepts to appropriate architectures and the like.

The workshop identified specific opportunities, re-engineering existing simulations and developing complex systems simulations in new domains and drawing on MDE and DSML researchers, to create families of interconnected languages:

1. Re-engineering existing agent-based biological system applications, notably the CoSMoS simulations (above) and the work of Bentley's Cellular Adaptive Behavior Lab at the Francis Crick Institute,[7] to support extension, further analysis, improved documentation, and so on.
2. Scaling-up and performance enhancement of existing simulations, notably Bentley's MemAgent-Spring Model simulation; exploration of parallelization support (e.g., FLAME GPU or Slingshot graph-based approach).

7 https://www.crick.ac.uk/research/labs/katie-bentley

3. Demonstrating application to new domains, and end-to-end DSML-based simulator creation (e.g., supporting Keele's smart transport live lab).
4. Linkage to DSL researchers to create families of interconnected languages.

In addition, the participants agreed to undertake a review of existing tools, systems, and approaches for each of the different challenges of the overall vision. This mapping exercise is under way (led by Ph.D. and other project researchers at King's College and the Francis Crick Institute), and will lead to a publication in due course. Finally, the Workshop participants, and a wider community of interest, identified the need for ongoing workshops to follow up, as well as outputs to display outcomes that facilitate the development and use of fit-for-purpose complex systems simulations.

References

1. Alden, K. (2012). *Simulation and statistical techniques to explore lymphoid tissue organogenesis*. Ph.D. thesis, University of York. http://etheses.whiterose.ac.uk/3220/.
2. Alden, K., Timmis, J., Andrews, P. S., Veiga-Fernandes, H., & Coles, M. C. (2016). Extending and applying Spartan to perform temporal sensitivity analyses for predicting changes in influential biological pathways in computational models. *IEEE Transactions on Computational Biology*, *14*(2), 431–442.
3. Alden, K., Andrews, P. S., Polack, F. A., Veiga-Fernandes, H., Coles, M. C., & Timmis, J. (2015). Using argument notation to engineer biological simulations with increased confidence. *Journal of the Royal Society Interface*, *12*(104).
4. Bentley, K., Gerhardt, H., & Bates, P. (2008). Agent-based simulation of notch-mediated tip cell selection in angiogenic sprout initialization. *Journal of Theoretical Biology*, *250*, 25–36.
5. Bentley, K., Philippides, A., & Ravasz, R. E. (2014). Do endothelial cells dream of eclectic shape? *Developmental Cell*, *29*(2), 146–158.
6. Evans, S., Alden, K., Cucurull-Sanchez, L., Larminie, C., Coles, M. C., Kulberg, M. C., & Timmis, J. (2017). ASPASIA: A toolkit for evaluating the effects of biological interventions on SBML model behaviour. *PLoS Computational Biology*, *13*(2). https://doi.org/10.1371/journal.pcbi.1005351.
7. Greaves, R. B., Dietmann, S., Smith, A., Stepney, S., & Halley, J. D. (2017). A conceptual and computational framework for modelling and understanding the non-equilibrium gene regulatory networks of mouse embryonic stem cells. *PLoS Computational Biology*, *13*(9). https://doi.org/10.1371/journal.pcbi.1005713.
8. Greaves, R. B., Polack, F. A. C., & Forrester, J. (2012). CoSMoS in the context of social-ecological systems research. In S. Stepney, P. S. Andrews, & M. N. Read (Eds.), *Proceedings of the 2012 Workshop on Complex Systems Modelling and Simulation* (pp. 47–76). Orleans, France: Luniver Press.
9. Herd, B., & Miles, S. (2019). Detecting causal relationships in simulation models using intervention-based counterfactual analysis. *ACM Transactions on Intelligent Systems and Technology*, *10*(5).
10. Herd, B., Miles, S., McBurney, P., & Luck, M. (2018). Quantitative analysis of multi-agent systems through statistical verification of simulation traces. *International Journal of Agent-Oriented Software Engineering*, *6*(2), 156–186.
11. Leonov, G. (2016). *An integrated molecular cell biology and agent-based simulation approach to dissecting micro*RNA *regulatory networks*. Ph.D. thesis, University of York. http://etheses.whiterose.ac.uk/12032/.
12. Mısırlı, G., Taylor, R., Goñi-Moreno, A., McLaughlin, J. A., Myers, C., Gennari, J. H., Lord, P., & Wipat, A. (2019). SBOL-OWL: An ontological approach for formal and semantic representation of synthetic biology information. *ACS Synthetic Biology*, *8*(7), 1498–1514.
13. Moyo, D. (2014). *Investigating the dynamics of hepatic inflammation through simulation*. Ph.D. thesis, University of York. http://etheses.whiterose.ac.uk/6967/.
14. Polack, F. A. C., Andrews, P. S., Ghetiu, T., Read, M., Stepney, S., Timmis, J., & Sampson, A. T. (2010). Reflections on the simulation of complex systems for science. In R. Calinescu & R. Paige (Eds.), *15th IEEE International Conference on Engineering of Complex Computer Systems* (pp. 276–285). Oxford, UK: IEEE Press.

15. Read, M. N. (2011). *Statistical and modelling techniques to build confidence in the investigation of immunology through agent-based simulation*. Ph.D. thesis, University of York.

16. Stepney, S., & Polack, F. A. C. (2018). *Engineering simulations as scientific instruments: A pattern language*. Cham: Springer.

17. Williams, R. A. (2014). *An agent-based model of the IL-1 stimulated nuclear factor-kappa B signalling pathway*. Ph.D. thesis, University of York. http://etheses.whiterose.ac.uk/7808/.