Opportunities for Simulation in Software Engineering

Breno Bernard Nicolau de França[1], **Valdemar Vicente Graciano Neto**[2]

[1]Universidade Estadual de Campinas

[2]Universidade Federal de Goiás

breno@ic.unicamp.br, valdemarneto@ufg.br

Abstract. *Simulation has been successfully used in several domains, for research and practical purposes. Systematic approaches for simulation arose and a myriad of simulation models were proposed in the context of Software Engineering over the past decades. Despite the lack of rigor and industrial relevance on many of these, we discuss the existing synergies and consolidated knowledge to foster new opportunities between these areas.*

1. Introduction

Interdisciplinary research involving the fields of Simulation and Software Engineering (SE) have been explored since the 1970s. Until today, several proposals emerged trying to solve problems in both areas. Mainly, two types of contributions appear in the literature: (i) initiatives using SE techniques and methods to support the development of simulation models and programs; and (ii) simulation as a research method, the so-called Simulation-Based Studies (SBS), to support SE investigations and decision-making. Although we have reached an understanding that several SBS lack rigor and practical relevance (de França and Ali 2020), we claim the SE field and knowledge evolved, along with guidelines for conducting that sort of study. It brings new perspectives, in which research can improve methods for SBS and open up opportunities in face of new SE challenges.

This position paper recaptures the discussion about these topics to foster new and groundbreaking research using SBS to investigate contemporary challenges in SE. For that, we explore a little bit of history between those areas in Section 2. Section 3 presents the synergies involving simulation and SE. Section 4 presents some domains in which there are opportunities to conduct SBS to support SE investigations. Finally, Section 5 presents the final remarks with a positive message towards a more interdisciplinary work.

2. Simulation and Software Engineering

Both Simulation and SE prescribe the use of models at different levels of abstraction and formalism. While simulation requires an underlying model that is executed over time to materialize the system behavior, SE adopts models for capturing systems (static and dynamic) properties to support portions of the software development life cycle.

The history of adopting SE methods to develop simulation models dates from 70s, with proposals like the use of structured design techniques for abstracting simulation models (Ryan 1979). And, it evolved towards the use of software process maturity models to define simulation modeling life cycle. On the opposite direction, applying simulation models to solve or investigate software engineering problems started in the 80s with attempts of simulating the dynamics of software processes (Abdel-Hamid and Madnick 1983).

After that, we observe decades of contributions in conducting SBS in the context of software engineering (de França and Travassos 2013). de França and Travassos show that, until 2011, most of the research concentrated on simulating software processes/projects. In this sense, Ali and Petersen provided later detailed information on software process simulation in industrial contexts (Ali et al. 2014). Besides, both reviews conclude that SBS captured in their contexts mostly lack rigor and practical relevance.

These findings culminated in a set of guidelines to support planning and reporting of SBS in the context of SE. Later, de França and Ali consolidated this knowledge in a process for conducting this sort of studies (de França and Ali 2020).

3. Synergies between Software Engineering and Simulation

We present, in Figure 1, the main idea fostering the synergy between these areas. Under this perspective, SE can provide methods and techniques to support the development of simulation models. On the other hand, simulation could enable the prediction of software properties still at design-time, or support the analysis of modern software development methods and practices to increase their understanding and to anticipate their limitations. Additionally, simulation can provide observation at scale, where empirical or experimental studies are unfeasible (see Section 4).

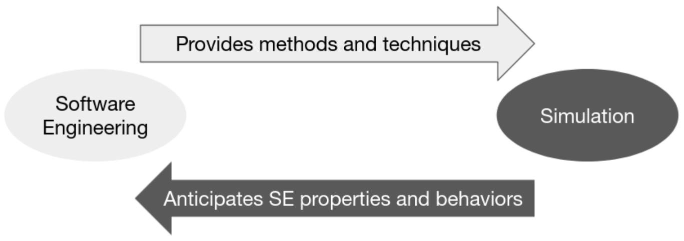
Provides methods and techniques
Software Engineering
Simulation
Anticipates SE properties and behaviors

Figure 1. Mutual benefits involving simulation and software engineering

Even SE and SBS having different purposes, being SE more problem-solving oriented and SBS concentrating more on investigation and knowledge acquisition, the synergy arises by understanding that: (i) software and simulation models are abstractions; (ii) simulators are software developed under particular simulation approaches; and (iii) software processes share similarities with SBS life cycle, which we explore next.

Problem Definition and System Observation. SBS include system observation and data collection for modeling. In this sense, requirements elicitation techniques can provide relevant data for simulation modeling. Conversely, software processes can benefit from simulations in the very beginning, where simulations can provide insights and discard alternative solutions that do not perform well in simulated *what if* scenarios, providing early feedback and perception of the system. This last scenario represents the state-of-the-practice in Systems Engineering.

Design Principles. As we understand, simulation models and software are abstractions of an intended system. Hence, depending on the scale of the simulation model, maintainability, reusability, and other quality attributes rises as concern. Hence, techniques and methods for modularization, separation of concerns, reuse, and model composition are also demanded for simulation models.

Software Testing vs. Simulation V&V Procedures. Generally, simulation models have an underlying formal basis abstracted in the selected approach. Hence, the application of V&V procedures is feasible, potentially facilitating the identification and mitigation of defects and failures (de França and Ali 2020). These procedures could be improved with software testing techniques and criteria already established in the SE literature. At the other perspective, performance, robustness, and security testing can benefit from simulation models for generating loads of requests, simulating malicious inputs and attacks to support software verification.

Production Real-Time Monitoring. Data collection at the production environment (including logs) can be an accurate data source to collect calibration data and create statistical distributions to support simulation model development and input data generation. Such an approach allows to conduct SBS in order to validate candidate software architectures or improvement opportunities.

Existing tools. The tool support provided on simulation models specification and execution, such as visual and animation mechanisms, could aid software architects, designers, and engineers to better understand the impact of design decisions and how they can affect the entire system. In critical systems, the software driving the system operation should be free of failures, and simulations can support evaluation activities by enabling the identification of latent problems, anticipating failures, enabling monitoring system operation, and predicting, at design-time, the behavior to be achieved at runtime. Errors can then be reduced earlier, increasing the level of trustworthiness provided by the system.

Clean simulation code. Simulation models are often coded (or at least can be translated) into simulation programming languages. In case of large-scale models, it hinders readability and comprehensibility, which ultimately impacts on code maintainability. Simulation models can also benefit from practices on internal quality, such as clean code, to improve reading, analisability, and modifiability.

Model-Driven Engineering (MDE). Several contributions appeared on the automated transformation of software models (e.g., UML models) into simulation models and vice-versa. Recently, the MDE community has invested on platforms to interpret models at runtime, materializing the software system itself. For instance, the idea of *models@runtime* builds upon the premise that models are also software (Blair et al. 2009), allowing to define animations and operational semantics. This can offer a visual perception as a metaphor for the system operation, evidencing messages exchange among systems components and allowing visualization of emergent behaviors. Such abstraction can be applied along with simulation techniques to provide accurate behavioral models.

4. Opportunities and Challenges

In this section, we discuss some opportunities and challenges, in which both simulation and SE can contribute together.

Systems-of-Systems. Systems-of-Systems (SoS) are sets of independent systems formed by independent software-intensive systems that work collaboratively to achieve a common goal. Smart cities are a remarkable instance of SoS targeting goals as fluid traffic, as a result of interoperability between autonomous vehicles and a smart traffic system. Besides, smart cities have the potential to reach hundreds of constituents systems working together. Hence, it is prominent to develop techniques to represent those large-scale systems and predict their properties still at design-time. However, to reliably simulate so many constituents, parallel and distributed simulation can also be required, which demands techniques to be investigated.

Software Ecosystems. Software Ecosystems (SECO) are conceived as a set of players and business centered on a common technological platform. Apple Store is an example of SECO, founded on a commercial platform that enable commercialization of several products, while co-habiting several different clients and suppliers. Simulation allows the assessment of strategical decision-making in SECO Management, as well as the prediction on how human, business, and technological factors influencing the SECOs may impact on its results, evolution and potential to generate value.

Trustworthiness and Critical Systems. Critical systems are those whose failures could cause injuries or financial losses to their users. Given such hard requirements, simulations can certainly help on the modeling and prediction of properties of these types of systems (along with assurance cases), anticipating and avoiding failures that could cause environmental disasters or other types of damage. In this sense, simulation models for that should include the notion of trustworthiness, which is a composite software quality attribute involving several other properties that need to be addressed, including tradeoffs.

Cyber-physical Systems. The growing demands for integration software and hardware solutions in fields as embedded sytems, robotics, IoT, and high-performance networks makes simulation a natural candidate to evaluate initial assumptions before experimenting and testing with expensive hardware or complex heterogeneous devices. Digital Twins is also a trend that could be benefited from the association between SE and simulation.

Continuous Software Engineering. This emergent software engineering perspective refers to the *continuous flow* of the software lifecycle, from the business analysis (BizDev) to its full operation and use (DevOps) (Fitzgerald and Stol 2017), providing fast, flexible, and frequent work into small cycles, incrementaly deliverying functional software. Therefore, collecting data from practices as continuous deployment, real-time user and production monitoring enable to develop simulation models to forecast future scenarios regarding the software process and product, identifying wastes and bottlenecks, as well as the improvement of the delivery cycle. Additionally, software product can benefit also from the execution of automated V&V to build modern reliability models. Release and product stability could also be modelled using data from multiple executions of deployment procedures. Finally, pipeline executions can provide maintainability metrics, providing inputs for developing evolution models.

Although it is possible to envision several opportunities, there are also some challenges along the way. We highlight some general concerns, as follows.

In general, when developing simulation models, researchers need to observe and collect data from the real system or phenomenon in most of simulation studies (i.e., *in virtuo* and *in silico*), so they can understand what is to be simulated. For that, software engineering research often conduct *in vivo* and *in vitro* studies, having a closer contact with the object of study. That cycle sets a challenge: to conduct faster and more studies in a computational environment, we need to conduct more real-world and risky studies to gather knowledge to develop the computational models. It is particularly challenging in large-scale systems engineering, where we may have accurate model for physical subsystems, but software systems may introduce more uncertainty and making simulation models more complex, threating the study validity.

Other challenge concerns the application of simulation in Systems Engineering, which has been a *de-facto* standard for practice. However, taking simulation as software, the Systems Engineering domain lacks practices and guidelines on how to apply well-established SE techniques to improve the quality of the simulation code, such as modularization, refactoring, reuse, and other techniques to improve maintainability. Hence, we need to introduce SE disciplines into the knowledge areas with higher demands for simulation, which would encompass revisiting graduation courses and professional education.

In this sense, simulation education is also an important topic. We are not aware of many simulation courses being taught in Brazil, mainly relating it with Software Engineering. We hope the creation of MSSiS forum establishes and disseminates knowledge, and motivate the investigation of these topics in association. Finally, the level of accuracy of a simulation model depends on the study purpose, i.e., a model should exhibit as much details as the granularity needed to support inferences about a given system of interest. Therefore, a multi-resolution approach should be adopted to evaluate it, i.e., the model should be validated according to multiple granularities levels until someone confirms the model can provide results in a reliable way.

5. Final Remarks

This paper discussed some opportunities and challenges for simulation in software engineering, based on existing synergies between these areas. We discussed how these areas could foster each other, what emerging and contemporary domains in software engineering could be benefited from simulation approaches, and also externalized a set of challenges that should be solved in the forthcoming years to promote an appropriate use of simulation in the context software engineering.

References

[Abdel-Hamid and Madnick 1983] Abdel-Hamid, T. K. and Madnick, S. E. (1983). The dynamics of software project scheduling. *Communications of the ACM*, 26(5):340–346.

[Ali et al. 2014] Ali, N. B., Petersen, K., and Wohlin, C. (2014). A systematic literature review on the industrial use of software process simulation. *J Syst Softw*, 97:65–85.

[Blair et al. 2009] Blair, G., Bencomo, N., and France, R. B. (2009). Models@ run.time. *Computer*, 42(10):22–27.

[de França and Ali 2020] de França, B. B. N. and Ali, N. B. (2020). The role of simulation-based studies in software engineering research. In *Contemporary Empirical Methods in Software Engineering*, pages 263–287. Springer.

[de França and Travassos 2013] de França, B. B. N. and Travassos, G. H. (2013). Are we prepared for simulation based studies in software engineering yet? *CLEI Electron. J.*, 16(1).

[Fitzgerald and Stol 2017] Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *J Syst Softw*, 123:176–189.

[Ryan 1979] Ryan, K. T. (1979). Software engineering and simulation. Technical report, Institute of Electrical and Electronics Engineers (IEEE).