Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

SIMULATION-SUPPORTED ENGINEERING OF SELF-ADAPTIVE SOFTWARE SYSTEMS

Tom Meyer
Andreas Ruscheinski
Pia Wilsdorf
Adelinde M. Uhrmacher

Institute for Visual and Analytic Computing
University of Rostock
Albert-Einstein-Straße 22
Rostock, 18059, GERMANY

ABSTRACT

Engineering a self-adaptive software system is challenging. During design- as well as run-time, assurance cases are central for ensuring reliable operation of the software. Simulation, in addition to software verification and testing, is a viable means to provide evidence for assurance cases. So far, little attention has been given to the development of underlying simulation models. Here, we argue that a systematic approach to develop simulation models will enhance the overall engineering process and will contribute to seamless integration of simulation and engineering processes. In our approach, we relate an explicit representation of the conceptual model and simulation experiments to artifacts of the engineering process. We will show first steps of applying our approach in a concrete ongoing software project for medical diagnosis, and discuss the role of components of the conceptual model in designing the software as a self-adaptive software system.

1 INTRODUCTION

Self-adaptive systems are "systems that are able to adjust their behavior in response to their perception of the environment and the system itself" (Cheng 2009). Central to self-adaptive systems is the *adaptation controller*, which changes the configuration or structure of the software system, i.e., the *adaptation target*, according to some goals and based on specific requirements. Thereby, the state of the software and its environment needs to be taken into account.

Designing *self-adaptive software systems* (SAS) is challenging. Not only is it difficult to build a reliable change mechanism, triggering exactly when a change should occur, but the resulting system also has to behave as expected as well. Especially in safety-critical areas, e.g., in Cyber-Physical Systems or in medicine where patient health and lives are at risk the engineering process needs to ensure that adaptations work reliably according to the defined requirements (Bolbot et al. 2019). Therefore, assurance cases can be used during the life cycle of SASs. They allow a continuous assurance referring to monitoring, analyzing, guaranteeing, and predicting system properties throughout the entire software life cycle (Calinescu et al. 2017).

For assurance evidence, they rely on formal models which are an essential ingredient in engineering SASs during design- and run-time (Blair et al. 2009; Cheng et al. 2014). All models that are used during design- and run-time should be valid representations of the system and the environment, as their validity and faithfulness are considered critical for the successful use and operation of the self-adaptive system (Aßmann et al. 2014). This also applies to simulation models which are not only applied during design-time but also form an efficient means for selecting adaptation options during run-time (Weyns and Iftikhar 2016) a method that is similar to symbiotic simulation (Fujimoto et al. 2002). While frameworks and approaches

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

emphasize the role of models, including simulation models, and their faithfulness in engineering SASs, little attention is given toward how these models are developed.

In the following, we will explore how a systematic development of simulation models, where different artifacts of the modeling and simulation life cycle are made explicit, not only contributes to the validity (or faithfulness) of simulation models but also unlocks synergistic effects in the overall engineering process of self-adaptive systems.

2 IDENTIFYING SYNERGIES BETWEEN ENGINEERING AND SIMULATION LIFE CYCLES

To identify possible synergies between a SAS engineering life cycle and the modeling and simulation life cycle, we will first shortly discuss both cycles separately, to identify shared concepts and interdependencies that can be exploited for more effective engineering of SAS.

2.1 Life Cycle of Modeling and Simulation

Simulation studies are intricate processes interweaving activities such as system analysis, conceptual modeling, simulation model development, and simulation experiment execution (Balci 1998; Sargent 2013). Often, the authors of these processes look at simulation studies from a certain perspective to highlight the importance of specific activities, such as model validation (Balci 1998), products, such as the conceptual model (Sargent 2013; Robinson 2008), or their relations (Ruscheinski et al. 2019b).

Various definitions of the conceptual model exist (Bock et al. 2017), here we adopt the interpretation of the conceptual model as a collection of various early-stage products of a simulation study, e.g., the research objective, requirements, qualitative model, (specification of) data and information sources, or assumptions (Wilsdorf et al. 2020b).

Other products of the modeling and simulation life cycle are the simulation model itself and simulation experiments, which are closely related to the conceptual model during the modeling and verification, experimentation, and result interpretation phases (see Figure 1). General modeling formalisms such as DEVS (Zeigler et al. 2018) or domain-specific languages allow specifying the simulation model (Helms et al. 2017; Loreti and Hillston 2016) unambiguously and in an executable manner. The semantics of languages and formalisms also guides the development of efficient execution algorithms (Helms et al. 2017). To enhance the reproducibility and reuse of simulation experiments, various approaches exist that support their specification, generation, and execution. To those belong, e.g., formats (Wilsdorf et al. 2019), and domain-specific languages (Ewald and Uhrmacher 2014). Making products of the simulation study explicit and specifying relations and constraints declaratively, e.g., based on an artifact-based workflow, supports guidance and documentation of simulation studies (Ruscheinski et al. 2019b; Ruscheinski et al. 2019a). For example, to validate a simulation model, all simulation experiments with the role validation, that are specified for the simulation model and its associated requirements are automatically executed (Ruscheinski et al. 2019b). Explicit requirements (or behavioral properties (Batt et al. 2006) or hypotheses (Lorig et al. 2017)) and explicitly specified simulation experiments make this automatic generation and execution of simulation experiments possible.

2.2 Life Cycle for Engineering Self-Adaptive Software Systems

The engineering of SASs is a similarly complex process. Although the implementation of the adaptation controller can be implicit inside a SAS, an architecture that clearly distinguishes between the adaptation target and adaptation controller facilitates the design and testing of a SAS (Weyns 2019). Whereas the adaptation target implements the basic functionality, the controller is responsible for triggering at the right time, the right changes within the adaptation target. A classic approach for designing the controller involves: monitoring operation and environment, continuous re-evaluation of the situation, identification of suitable adaptations, and the execution of the adaptation. This approach of Monitor, Analyze, Plan, and Execute is

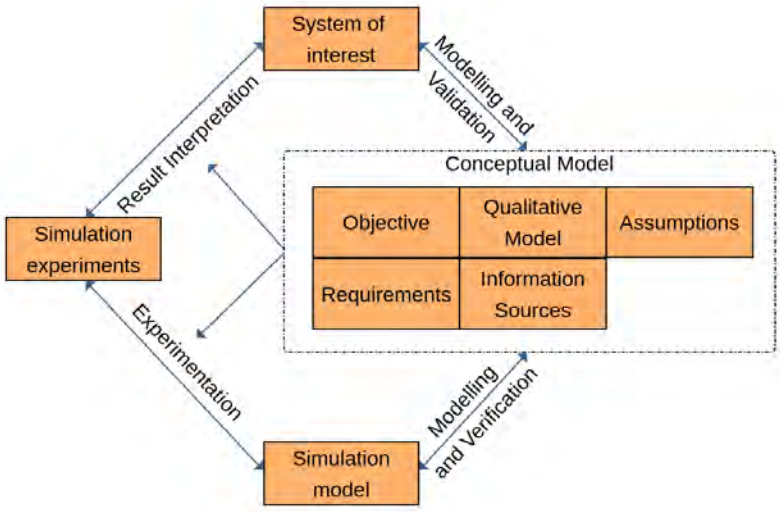

Figure 1: Overview of the modeling and simulation life cycle including central artifacts, such as conceptual model, simulation model, and simulation experiments, adapted from (Sargent 2013) and (Robinson 2008).

known as MAPE pattern, or MAPE-K pattern (in which a knowledge-base with adaptation requirements is shared among MAPE's components) (Kephart and Chess 2003).

An essential part of the software design process is the definition of requirements. They are defined based on the goals of the SAS, which capture the objectives that the SAS should achieve, despite or due to its adaptations. In general, goals form a core concept of requirement engineering (Yu and Mylopoulos 1998), as they are useful for specifying, analyzing, and maintaining requirements. Some requirements might specifically address individual components of the SAS. For example, they might refer to the adaptation controller, demanding the adaptation-loop to be deadlock-free (Khakpour et al. 2016), or they will define expectations and constraints to the MAPE stages. Requirements that refer to the adaptation target need to consider the possible effects of adaptations. Also, the dynamic environment might introduce uncertainties, which further influences requirement specification and adaptation (Cheng et al. 2009).

Similarly, like requirements, component models are significant in engineering SAS (Blair et al. 2009; Cheng et al. 2014). During run-time, the SAS can leverage such declarative specified knowledge about itself and its environment, to reason and plan adaptations (Bennaceur et al. 2014). Models "provide up-to-date and exact information about the system to drive subsequent adaptation decisions; and if the model is causally connected, then adaptations can be made at the model level rather than at the system level" (Blair et al. 2009). E.g. during run-time, simulation models can improve the analysis of adaptation options (Weyns and Iftikhar 2016). Thereby, different target-system and environment models may have to be maintained for different concerns, such as performance, reliability, and functional requirements, and purposes like verification and simulation. Consequently, there is plenty of research in modeling SASs both for adaptation process and target-systems (Villegas et al. 2013; Weyns et al. 2012; Zhang and Cheng 2006).

To reduce the likelihood that system requirements exist only on paper, assurance cases (Bishop et al. 2004) have been introduced (De Lemos et al. 2014). Assurance cases document the *assurance evidence* for *assurance claims* (e.g., our system requirements) with a connecting *assurance argument*, which should clarify why the evidence supports the claims. However, SASs present a special case for assurance cases. Since typically not all reachable configurations and states generated by the adaptation process are known, collecting assurance evidence is not limited to design and test phases. Instead, assurance evidence needs to be gathered during the entire life cycle of adaptive software, including its run time (Weyns et al. 2017).

Figure 2 shows a life cycle for SAS Development based on Calinescu et al. (2017) and Andersson et al. (2013): we can divide the life cycle in online and offline adaptation phases (Andersson et al. 2013). In the online phase, the system can autonomously adapt to changes in the environment, whereas the adaptations that go beyond the modeled adaptation capabilities, have to be done offline by the system maintainers.

The life cycle begins in an initial offline phase, where the first system iteration is developed. In this phase, the models for adaptation controller, target system, and environment are created according to system and adaptation requirements. For the controller models, existing reference models (Weyns et al. 2012; Iftikhar and Weyns 2014; Calinescu et al. 2017) can be adapted and verified formally to produce strong assurance evidence. The resulting evidence should then be connected properly to the required assurance claims.

If the models changed and require an adaptation in the SAS implementation, the new implementation can be deployed to an online phase. The online phase perpetually produces assurance evidence which may be required to develop complete assurance arguments.

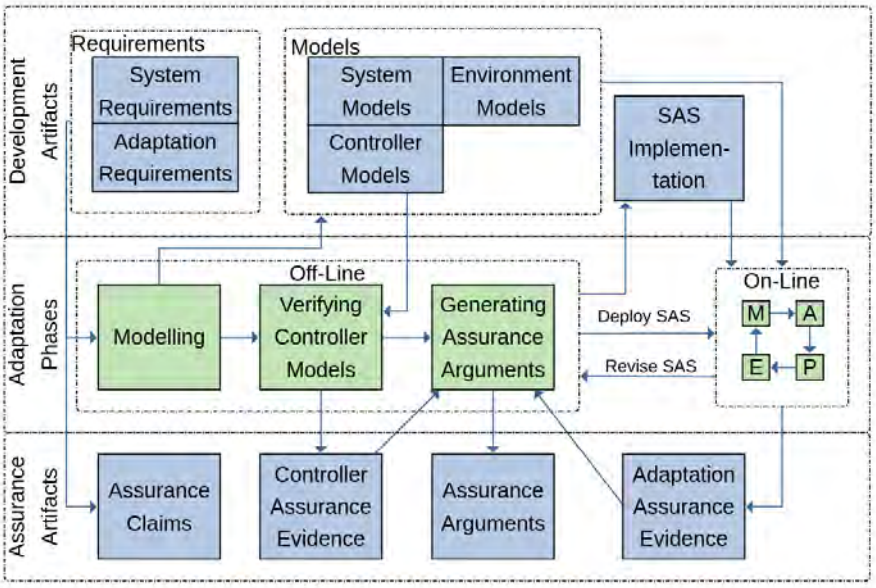

Figure 2: Development process for self-adaptive software based on (Calinescu et al. 2017; Andersson et al. 2013)

3 SIMULATION-SUPPORTED ENGINEERING OF SAS

Both modeling and simulation life cycle, as well as SAS engineering life cycle, provide important information for each other (see Figure 3). The goals and requirements of the SAS engineering life cycle can be translated into research questions and requirements of the simulation study. The conceptual model of the simulation study reveals valuable information for assessing the reliability of the SAS. For example, those parts that refer to the environmental model, indicate the degree and range of scenarios, for which the SAS is likely to work as intended (via assumptions, data used for calibration, etc.). Simulation experiments, particularly if made explicit, provide evidence and at the same time arguments for assurance cases. The qualitative model, which is part of the conceptual model of the simulation study, should be consistent with models in engineering SAS.

The requirements that are used as assurance claims are very similar to the requirements for the conceptual model of the simulation study. However, the requirements in engineering SAS refer only to the SAS, whereas

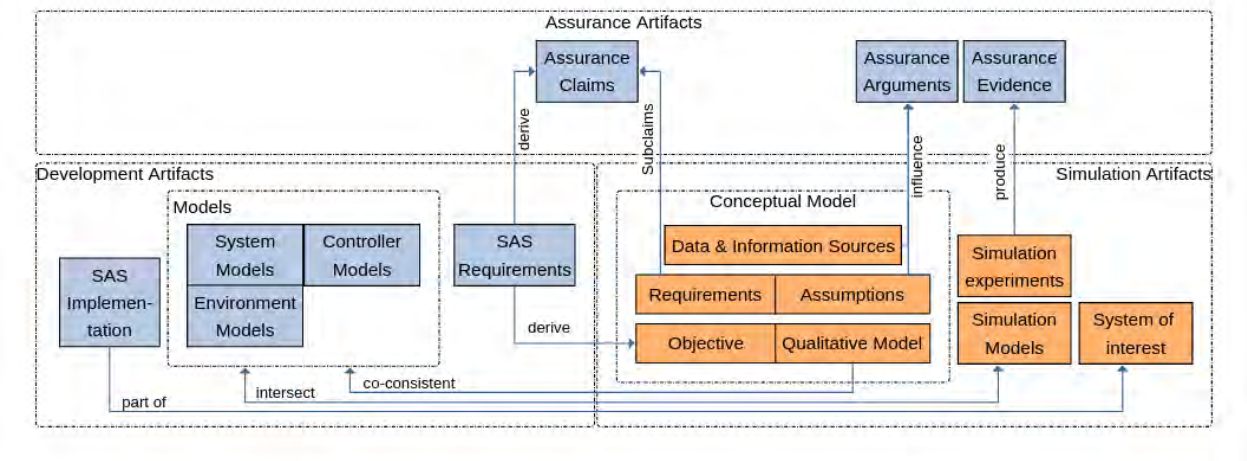

Figure 3: Relation between artifacts from simulation and engineering life cycle

requirements as for the conceptual model refer to all models: controller models, target-system models, environment models, as well as compositions from all of these. Also, the conceptual model might constrain how the models should behave. This is particularly relevant for the environment model, where the data used for validation and calibration allow to evaluate the range of environmental behavior in which the SAS has been analyzed. Assumptions and simplifications that refer to the environment model, the adaptation controller, and the target system, provide further valuable information. This information has so far no counterpart in the SAS engineering frameworks we have seen; nonetheless, it appears important to evaluate the models and the conclusions derived based on these models.

In engineering SAS, models are also used to know what and how to monitor, how to analyze, and how to plan the adaptations (Bennaceur et al. 2014). The qualitative model as part of the conceptual model of a simulation study, however, can be used to generate specific experiments (Wilsdorf et al. 2020a) which in turn can provide evidence for assurance cases. To ensure the consistency of the qualitative model, typically a list of variables and some causal relations, with other models, approaches from engineering such as mega models or provenance can be used. Mega models capture the different development models and their dependencies to address consistency (and traceability) in the development process (Bézivin et al. 2004). Recently the concept of provenance has also been introduced into the field of simulation studies. Exploiting provenance standards the different artifacts of simulation studies and their relations can be defined, conveniently stored, and accessed e.g., by querying a graph database, beyond individual simulation studies (Ruscheinski and Uhrmacher 2017; Budde et al. 2021). The approach has successfully been used to define families of simulation models consisting of more than 19 simulation studies, each with its own research questions, requirements, assumptions, and data and information sources (Budde et al. 2021).

Thus, we find multiple interdependencies between the processes of conducting simulation studies and the SAS engineering process as well as between artifacts produced on the way. However, a crucial precondition for exploiting these interdependencies for a more effective (simulation-supported) engineering of SAS, is to make these various artifacts and their relation explicit. As a first step, we will look at how a conceptual model within simulation-based engineering of SAS can look like.

4 ENGINEERING AN ADAPTIVE SOFTWARE SYSTEM FOR PRIORITIZING GENETIC VARIANTS

We will put our methods to test in an ongoing software project and document our first steps. In this project, a SAS shall be developed that helps clinicians diagnosing rare diseases from genetic variants (mutations

in the DNA). First, we give a short overview of how these variants are identified and highlight the need for a SAS.

4.1 Identifying Disease-Causing Genetic Variants

Identifying disease-causing/pathogenic genetic variants plays a crucial role in diagnosing rare diseases as it is assumed that $\approx 80\%$ of such diseases have a genetic origin (Gülbakan et al. 2016). However, diagnosing rare disease cases is challenging. After sequencing and comparing the patient's DNA with the human reference genome, $25,000$ to $75,000$ genetic variants are found and have to be narrowed down to a handful of relevant variants (Cooper and Shendure 2011).

Variant prioritization methods (VPM) aim to support this process by calculating numerical scores predicting the functional impact of the variants (Bosio et al. 2019) and allow to rank the variants according to their relevance for the given case. Based on this *ranking* the clinical geneticist can focus their diagnosis first on the few highest-ranking variants for a given case. The methods rely on assumptions about how the pathogenicity of a variant can be recognized (Eilbeck et al. 2017), e.g., variants in highly conserved regions of the DNA are likely to be more damaging than variants in less conserved regions or common variants in a large population are likely to be benign. The variant prioritization methods are typically based on machine-learning algorithms. Curated gene-phenotype and gene-disease associations from public databases, e.g., ClinVar (Landrum et al. 2014) or HPO (Köhler et al. 2021) are used to train decision trees (do Nascimento et al. 2020) or Hidden Markov models (Shihab et al. 2013).

However, from the perspective of the clinician, the current approach to develop prioritization methods has several shortcomings. First, the resulting rankings of the methods do not encompass recent updates of curated databases since the prioritization methods are typically trained once or are at least updated irregularly (Rentzsch et al. 2019; Shihab et al. 2013) while other databases are updated regularly with new data, e.g., ClinVAR is updated every month. Second, the method does not reuse the knowledge from already diagnosed cases as this data is typically not available and thus can not be taken into account.

Therefore, a SAS shall be developed, which exploits changes in databases and already diagnosed cases to improve the variant prioritization over time.

4.2 A Conceptual Model for Supporting Engineering a SAS

In the following, we outline a conceptual model for the self-adaptive medical software following the definition of the conceptual model as proposed in (Wilsdorf et al. 2020b), i.e., as a collection of early-phase products of the modeling and simulation cycle. In particular comprising research questions, requirements, assumptions, data and information sources, and a qualitative description of the simulation model(s).

Research Question: The first step in each simulation study is the definition of questions that shall be answered by applying simulation techniques. We discovered three main areas of questions in our project:

1. How does the interaction with the system influences the system itself?
2. How does the input data influence the learning process and the resulting rankings?
3. How do the rankings influence the laboratory process and their future input?

These questions map roughly on three possible abstraction levels for the environment model; The most abstract view would suffice for scalability questions, i.e., questions asking how the system acts under high load: handling a lot of requests, saving a lot of diagnoses, or handling a lot of learning runs.

The second set of questions focuses on the system results (the computed rankings for a request). System results implicitly influence all diagnoses and consequently patient treatment and health. Here, the simulation results are vital as assurance evidence, before the final product can be used productively. Examples would be the influence of diagnoses on learned rankings (and the learning process), the correctness of rankings, and how they improve with accumulating knowledge. In this case, the input data cannot be random and

has to satisfy some constraints. Simultaneously, we have to monitor and evaluate the resulting outputs. Nonetheless, the model of the laboratories could be relatively simple, since the diagnosis creation process is independent of learning and ranking creation.

The last set of research questions has the most potential to improve the development process of the adaptive system using simulation. These questions are concerned with the system's impact on the environment, e.g.: how rankings can be integrated best in the laboratory workflow (for consensus negotiation), what is the potential to accelerate diagnoses, or how well are they trusted (and thus affect patient treatment). While the previous questions do not need a very sophisticated model of the environment, these questions concern entities and details in the environment, which therefore have to be modeled. As a result, our laboratory model has to be relatively complex.

One big benefit a simulation can introduce here is the explicit modeling of properties that in reality are implicit or hard to observe. For example, the diagnoses of laboratories can have errors, which are hard to spot, or simply are not known to be errors at the time of diagnosis. We can make such errors explicit in a simulation model and track the propagation through our system into other laboratories. As this aspect is interesting we select the following research question to focus on and base further steps on it: *How do erroneous rankings propagate through laboratories and back into the system?*

However, to specify concrete requirements we need to know a bit more about the variables to monitor and analyze in the simulation models. Therefore, our next step is to define the qualitative model which will present a bird's eye view of the simulation model, its components, and their interactions.

Qualitative Model: First, we can distinguish between the SAS and the environment as shown in Figure 4. The SAS comprises the adaptive controller and the target system (the latter are all SAS components that are not the adaptation controller). The environment consists of laboratories that produce case data.

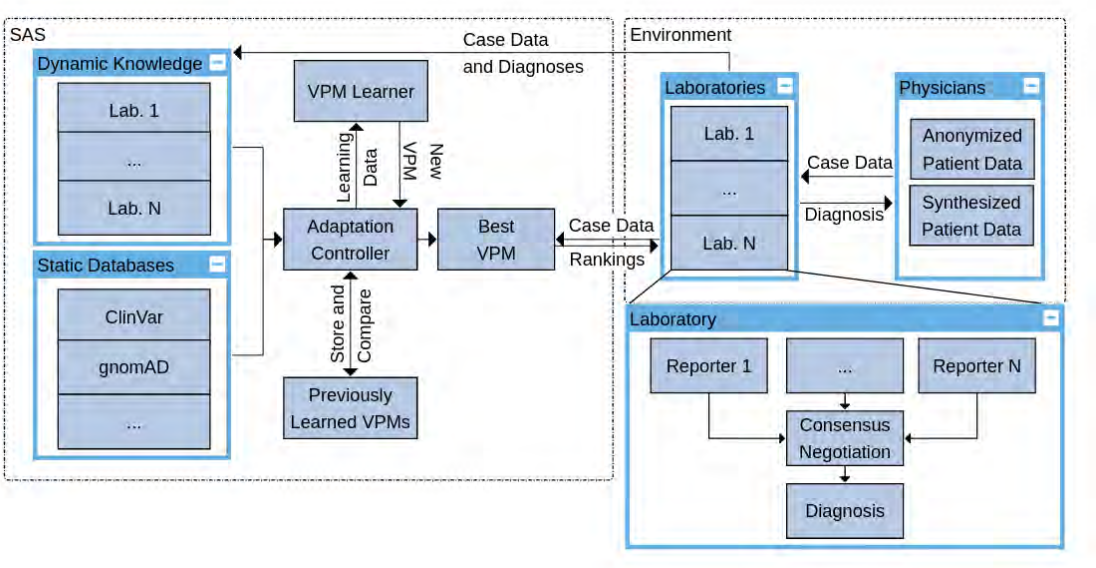

Figure 4: Qualitative model of the environment and its interactions with the SAS

A typical workflow begins with a patient who visits a physician (in a hospital or medical practice) with an illness. The detailed circumstances of patients are not of interest for our adaptive system, consequently, we represent patients as a set of genetic variants which has to be acquired or be generated somehow. Here, we can distinguish between two possible sources: real data that has been anonymized, or synthesized data that shares important features with the real data. To increase the credibility of the simulation, it would be wise to use both acquisition sources and test if they produce similar results.

The diagnosis process for rare genetic diseases is not done by the physician. It usually involves multiple medical reporters in a separate laboratory, who will settle on a diagnosis together. Together, physicians and laboratories form the environment of the SAS. They interact with it, by requesting rankings and providing their resulting diagnoses. The level of abstraction in this part is highly dynamic and depends on the research questions we want to answer. As argued earlier, depending on the modeling concern (be it reliability, scalability, etc.) different abstractions should be defined as part of the qualitative model. In Figure 4 we can already see two possible levels of abstraction for our laboratories: the most abstract view of a laboratory just forwards case data and turns the requested rankings into diagnoses—maybe even randomly generated. This corresponds to the first research question area and its abstraction level.

If we want to evaluate properties like scalability or throughput of the system, this can be sufficient. However, for an analysis of e.g. data integrity or feedback influence, the simulation model has to be more sophisticated (as mentioned in the third research question area). In such a scenario, we would model an actual laboratory workflow in more detail, as shown on the bottom right in Figure 4. This level of abstraction shows the laboratory process with several reporters who collectively diagnose a case, including an agreement on a final diagnosis. Both levels of analysis can be valid, depending on the objectives we want to solve.

The laboratories from the environment interact with our adaptive system. Its center is the adaptation controller which executes the MAPE-K loop (Kephart and Chess 2003). It monitors two knowledge sources: the *dynamic knowledge*, which is the accumulation of case and diagnosis data from the laboratories, and the *static databases*, which are the large curated medical databases designed for laboratory use. If the knowledge base changed sufficiently, the adaptation controller starts a new learning run. Our adaptation goal is to only use a better VPM than the current. Consequently, the newly learned VPM is analyzed and if it is better than the previous, the adaptation process can be planned and executed, resulting in the adaptation of the new VPM. Now, newly requested rankings are based on the improved VPM.

With the requested rankings, the laboratories can complete their diagnosis process and return the resulting diagnosis data to the requesting physician, who then can choose a treatment. Additionally, the diagnosis data is used in future learning iterations.

In addition to specifying the different model parts and their levels of abstraction, the qualitative model also contains information about the model parameters. Parameters are crucial when specifying a (conceptual) model, as these are the parts that can be tweaked during simulation experiments. In Table 1 we give some parameters for our simulation model. For example, we might make the number of simulated laboratories and reporters adjustable, for scalability questions, or introduce artificially simulated errors in diagnoses, to analyze error propagation.

Table 1: Parameter table comprised in the qualitative model.

Parameter	Domain
Reporter Expertise	Percentage
Reporter Negotiation Weight	Percentage
Laboratory Count	Integer
Reporter Count per Laboratory	Integer
Diagnosis Error Rate	Percentage

Data and Information Sources: To build a valid simulation model, it has to be based on various sources. These can be data obtained from measurements or questionnaires. For example, we can measure in a real-life laboratory, which reporter contributed which diagnoses in the consensus negotiation and where the diagnosis deviated from the consensus. Another source to model the laboratories are official guidelines (e.g. the ACMG Guidelines (Richards et al. 2015)). We can calibrate and validate our simulation with a comparison between these measurements and the results of our simulated laboratories. Furthermore, some

model parameters may be fixed based on other information sources such as scientific publications that might offer confirmed error rates.

So far, data and information sources are not explicitly reflected in the SAS development life cycle. However, the data or information sources can provide additional information for the assurance arguments. If simulation models are used to generate assurance evidence, and these models have been built, calibrated, and validated based on specific data and information sources, the assurance evidence produced might only be valid for exactly these sources.

Requirements: Having relevant research questions not only determines the abstraction level of the environment, but they are also helpful to formulate simulation requirements for modeling the system and its impact on the environment.

An assurance claim for the SAS, which also relates to our example research question is that *erroneous diagnosis do only propagate between laboratories if the SAS was repeatedly trained with sufficient erroneous data*: the SAS should only propagate errors from *wrong input* instead of errors from *bad learning*. Of course, we would like to see no propagation at all, but in reality, we cannot be sure that laboratories always produce correct diagnoses.

We can test this claim in a simulation experiment and define a concrete requirement that the SAS should be able to handle. The experiment could look like this: 30% of the simulated laboratories in the environment generate consistently wrong diagnoses for specific variants. These erroneous variants get checked into the knowledge base of our SAS and influence our learning runs. During the experiment, we collect all diagnoses on the same variant from the remaining 70% laboratories, which are the diagnoses that loop back into the SAS. If the diagnosis is also erroneous, the error has propagated and reinforces the wrong learning, if not, the error was called by the reporters of this laboratory, and error-correcting knowledge gets checked into the knowledge base. So, the error propagation rate is:

$$R_{prop} = \frac{WrongDiagnoses}{RightDiagnoses}$$

which we can measure for each simulation run. Now we can formulate a requirement for this scenario: $R_{prop} \leq 0.1$. We can evaluate this property with statistical model checking (Agha and Palmskog 2018) for all simulation runs.

To check this requirement the simulation model needs to be rather detailed referring to learning components and laboratory workflows. However, it is a good strategy to develop a sophisticated learning mechanism and tune things like adaptation behavior, credibility levels for laboratories, and other things that can influence the feedback loop between SAS and the environment.

If our experiment results satisfy the requirements, we still have no proof for the correctness of our SAS, but certainly, we have additional evidence. Here, we see the connection to assurance cases. Our experiment results map visibly on *assurance evidence*, whereas *assurance claims* are connected to requirements.

With that in mind we can see how a simulation study with an explicit conceptional model can improve the development process: not only does a structured notation help to analyze and understand the SAS better, the resulting simulation study can also be used to produce assurance evidence for an assurance claim.

Assumptions: To constrain our simulation model we can define simplifications. These are assumptions that restrict the level of detail of our processes or the interactions between them. A reasonable simplification in our model concerns the communication between different laboratories: In reality, members of a laboratory might interact and interchange knowledge due to personal or professional relations. However, the main diagnostic operation is done within the workflow of a laboratory. So, an articulated simplification could be: *Laboratories do not interact with each other directly.* Such a simplification can keep the model understandable and efficient. Assumptions indicate limitations of applying the models and influence how models are interpreted. They therefore should be considered in the assurance arguments of the combined SAS+Simulation development workflow (as shown in Figure 3). Assurance arguments provide explanations for the assurance claims based on evidence.

5 CONCLUSION AND FUTURE WORK

Developing self-adaptive software is a complex engineering task and testing the adaptation cycle requires runtime assurance of the complete system. One way to gather runtime assurance is by exploiting simulation models. We showed that the modeling and simulation life cycle (and in particular conceptual modeling) can be integrated into the adaptive-system life cycle from the start: it allows a more systematic development of the system and environmental models, and makes valuable context information accessible for SAS engineering by explicit early-stage products like the conceptual model and simulation experiments. The systems requirements of the engineering life cycle are mapped to the research objectives of the simulation studies. Assurance claims and sub-claims are mapped to simulation requirements, simulation experiments provide evidence for these assurance cases. The arguments of these assurance cases explain how the evidence, here the simulation experiment, gives substance to the assurance claim. In addition to the specification of a simulation experiment, which allows its replication, they are able to consider assumptions, data and information sources as possible constraints that apply to the simulation model.

We discussed these connections based on a self-adaptive medical software system for diagnosing genetic variants. The software is currently under development and as a medical product, it has to comply with high standards of quality. Assurance cases are central in its design. The generation of assurance evidence at runtime, before deploying the system productively, is a highly valuable contribution simulation provides, and a good reason to integrate this approach in the development process for self-adaptive systems seamlessly. Particularly, given the complexity of the endeavor, we hope that synergies between both life cycles will contribute to an effective and reliable design of the self-adaptive software system, and that part of this even transfers to the production process.

ACKNOWLEDGMENTS

This research was funded by research grant DFG UH-66/15 'MoSiLLDe' and the state of Mecklenburg-Vorpommern, Germany via the European Regional Development Fund (ERDF) of the European Union as part of the project 'IDEA-PRIO-U' (TBI-V-1-354-VBW-122).

REFERENCES

Agha, G., and K. Palmskog. 2018, January. "A Survey of Statistical Model Checking". *ACM Trans. Model. Comput. Simul.* 28(1):1–39.

Andersson, J., L. Baresi, N. Bencomo, R. De Lemos, A. Gorla, P. Inverardi, and T. Vogel. 2013. "Software Engineering Processes for Self-Adaptive Systems". In *Software Engineering for Self-Adaptive Systems II*, 51–75. Springer.

Aßmann, U., S. Götz, J.-M. Jézéquel, B. Morin, and M. Trapp. 2014. "A Reference Architecture and Roadmap for Models@ Run. Time Systems". In *Models@ run. time*, 1–18. Springer.

Balci, O. 1998. "Verification, Validation, and Accreditation". In *Proceedings of the Winter Simulation Conference (WSC)*, 41–48.

Batt, G., J. T. Bradley, R. Ewald, F. Fages, H. Hermans, J. Hillston, P. Kemper, A. Martens, P. Mosterman, F. Nielson, O. Sokolsky, and A. M. Uhrmacher. 2006. "06161 Working Groups' Report: The Challlenge of Combining Simulation and Verification". In *Simulation and Verification of Dynamic Systems*, edited by D. M. Nicol, C. Priami, H. R. Nielson, and A. M. Uhrmacher, Number 06161 in Dagstuhl Seminar Proceedings. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Bennaceur, A., R. France, G. Tamburrelli, T. Vogel, P. J. Mosterman, W. Cazzola, F. M. Costa, A. Pierantonio, M. Tichy, M. Akşit et al. 2014. "Mechanisms for Leveraging Models at Runtime in Self-Adaptive Software". In *Models run. time*, 19–46. Springer.

Bézivin, J., F. Jouault, and P. Valduriez. 2004. "On the Need for Megamodels". In *Proceedings of the OOPSLA/GPCE: Best Practices for Model-Driven Software Development workshop, 19th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 1–9. Citeseer.

Bishop, P., R. Bloomfield, and S. Guerra. 2004. "The Future of Goal-Based Assurance Cases". In *Proc. Workshop on Assurance Cases*, 390–395. Citeseer.

Blair, G., N. Bencomo, and R. B. France. 2009. "Models@ Run. Time". *Computer* 42(10):22–27.

Bock, C., F. Dandashi, S. Friedenthal, N. Harrison, S. Jenkins, L. McGinnis, J. Sztipanovits, A. Uhrmacher, E. Weisel, and L. Zhang. 2017. "Conceptual Modeling". In *Research Challenges in Modeling and Simulation for Engineering Complex Systems*, 23–44. Springer.

Bolbot, V., G. Theotokatos, L. M. Bujorianu, E. Boulougouris, and D. Vassalos. 2019. "Vulnerabilities and Safety Assurance Methods in Cyber-Physical Systems: A comprehensive Review". *Reliability Engineering & System Safety* 182:179–193.

Bosio, M., O. Drechsel, R. Rahman, F. Muyas, R. Rabionet, D. Bezdan, L. Domenech Salgado, H. Hor, J.-J. Schott, F. Munell et al. 2019. "eDiVA—Classification and Prioritization of Pathogenic Variants for Clinical Diagnostics". *Human mutation* 40(7):865–878.

Budde, K., J. Smith, P. Wilsdorf, F. Haack, and A. M. Uhrmacher. 2021. "Relating Simulation Studies by Provenance—Developing a Family of Wnt Signaling Models". *bioRxiv*.

Calinescu, R., D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly. 2017. "Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases". *IEEE Transactions on Software Engineering* 44(11):1039–1069.

Cheng, B. H. 2009. "RogÚrio de Lemos, Holger Giese, Paola Inverardi, Jeff Magee, Jesper Andersson, Basil Becker, Nelly Bencomo, Yuriy Brun, Bojan Cukic, et al. 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap". *Software engineering for self-adaptive systems. Springer* 7475:1–26.

Cheng, B. H., K. I. Eder, M. Gogolla, L. Grunske, M. Litoiu, H. A. Müller, P. Pelliccione, A. Perini, N. A. Qureshi, B. Rumpe et al. 2014. "Using Models at Runtime to Address Assurance for Self-Adaptive Systems". In *Models run. time*, 101–136. Springer.

Cheng, B. H., P. Sawyer, N. Bencomo, and J. Whittle. 2009. "A Goal-Based Modeling Approach to develop Requirements of an Adaptive System with Environmental Uncertainty". In *International Conference on Model Driven Engineering Languages and Systems*, 468–483. Springer.

Cooper, G. M., and J. Shendure. 2011. "Needles in Stacks of Needles: finding Disease-Causal Variants in a Wealth of Genomic Data". *Nature Reviews Genetics* 12(9):628–640.

De Lemos, R., D. Garlan, C. Ghezzi, and H. Giese. 2014. "Software Engineering for Self-Adaptive Systems: Assurances (Dagstuhl Seminar 13511)". In *Dagstuhl Reports*, Volume 3. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

do Nascimento, P. M., I. G. Medeiros, R. M. Falcão, B. Stransky, and J. E. S. de Souza. 2020. "A Decision Tree to Improve Identification of Pathogenic Mutations in Clinical Practice". *BMC medical informatics and decision making* 20(1):1–11.

Eilbeck, K., A. Quinlan, and M. Yandell. 2017. "Settling the Score: Variant Prioritization and Mendelian Disease". *Nature Reviews Genetics* 18(10):599–612.

Ewald, R., and A. M. Uhrmacher. 2014. "SESSL: A Domain-Specific Language for Simulation Experiments". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 24(2):1–25.

Fujimoto, R., D. Lunceford, E. Page, and A. M. Uhrmacher. 2002. "Grand Challenges for Modeling and Simulation". *Schloss Dagstuhl* 350.

Gülbakan, B., R. K. Özgül, A. Yüzbaşıoğlu, M. Kohl, H.-P. Deigner, and M. Özgüç. 2016. "Discovery of Biomarkers in Rare Diseases: Innovative Approaches by Predictive and Personalized Medicine". *EPMA Journal* 7(1):1–6.

Helms, T., T. Warnke, C. Maus, and A. M. Uhrmacher. 2017. "Semantics and Efficient Simulation Algorithms of an Expressive Multilevel Modeling Language". *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 27(2):1–25.

Iftikhar, M. U., and D. Weyns. 2014. "Activforms: Active Formal Models for Self-Adaptation". In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 125–134.

Kephart, J. O., and D. M. Chess. 2003. "The Vision of Autonomic Computing". *Computer* 36(1):41–50.

Khakpour, N., F. Arbab, and E. Rutten. 2016. "Synthesizing Structural and Behavioral Control for Reconfigurations in Component-Based Systems". *Formal Aspects of Computing* 28(1):21–43.

Köhler, S., M. Gargano, N. Matentzoglu, L. C. Carmody, D. Lewis-Smith, N. A. Vasilevsky, D. Danis, G. Balagura, G. Baynam, A. M. Brower, and et al.. 2021, Jan. "The Human Phenotype Ontology in 2021". *Nucleic Acids Research* 49(D1):D1207–D1217.

Landrum, M. J., J. M. Lee, G. R. Riley, W. Jang, W. S. Rubinstein, D. M. Church, and D. R. Maglott. 2014. "ClinVar: Public Archive of Relationships among Sequence Variation and Human Phenotype". *Nucleic acids research* 42(D1):D980–D985.

Loreti, M., and J. Hillston. 2016. "Modelling and Analysis of Collective Adaptive Systems with CARMA and its Tools". In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, 83–119. Springer.

Lorig, F., D. S. Lebherz, J. O. Berndt, and I. J. Timm. 2017. "Hypothesis-driven Experiment Design in Computer Simulation studies". In *2017 winter simulation conference (WSC)*, 1360–1371. Institute of Electrical and Electronics Engineers, Inc.

Rentzsch, P., D. Witten, G. M. Cooper, J. Shendure, and M. Kircher. 2019, Jan. "CADD: Predicting the Deleteriousness of Variants Throughout the Human Genome". *Nucleic Acids Research* 47(D1):D886–D894.

Richards, S., N. Aziz, S. Bale, D. Bick, S. Das, J. Gastier-Foster, W. W. Grody, M. Hegde, E. Lyon, E. Spector et al. 2015. "Standards and Guidelines for The Interpretation of Sequence Variants: a Joint Consensus Recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology". *Genetics in medicine* 17(5):405–423.

Robinson, S. 2008. "Conceptual Modelling for Simulation Part I: Definition and Requirements". *Journal of the operational research society* 59(3):278–290.

Ruscheinski, A., and A. M. Uhrmacher. 2017. "Provenance in Modeling and Simulation Studies - Bridging Gaps". In *Winter Simulation Conference (WSC 2017)*, 872–883. Electronic ISSN: 1558-4305: Institute of Electrical and Electronics Engineers, Inc.

Ruscheinski, A., T. Warnke, and A. M. Uhrmacher. 2019b. "Artifact-based Workflows for Supporting Simulation Studies". *IEEE Transactions on Knowledge and Data Engineering* 32:1064–1078.

Ruscheinski, A., P. Wilsdorf, M. Dombrowsky, and A. M. Uhrmacher. 2019a. "Capturing and Reporting Provenance Information of Simulation Studies Based on an Artifact-Based Workflow Approach". In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 185–196.

Sargent, R. G. 2013. "Verification and Validation of Simulation Models". *Journal of Simulation* 7(1):12–24.

Shihab, H. A., J. Gough, D. N. Cooper, P. D. Stenson, G. L. Barker, K. J. Edwards, I. N. Day, and T. R. Gaunt. 2013. "Predicting the Functional, Molecular, and Phenotypic Consequences of Amino Acid Substitutions using Hidden Markov Models". *Human mutation* 34(1):57–65.

Villegas, N. M., G. Tamura, H. A. Müller, L. Duchien, and R. Casallas. 2013. "DYNAMICO: A Reference Model for Governing Control Objectives and Context Relevance in Self-Adaptive Software Systems". In *Software engineering for self-adaptive systems II*, 265–293. Springer.

Weyns, D. 2019. "Software Engineering of Self-Adaptive Systems". In *Handbook of Software Engineering*, 399–443. Springer.

Weyns, D., N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek et al. 2017. "Perpetual Assurances for Self-Adaptive Systems". In *Software Engineering for Self-Adaptive Systems III. Assurances*, 31–63. Springer.

Weyns, D., and M. U. Iftikhar. 2016. "Model-Based Simulation at Runtime for Self-Adaptive Systems". In *2016 IEEE International Conference on Autonomic Computing (ICAC)*, 364–373.

Weyns, D., S. Malek, and J. Andersson. 2012. "Forms: Unifying Reference Model for Formal Specification of Distributed Self-Adaptive Systems". *ACM Transactions on Autonomous and Adaptive Systems (TAAS)* 7(1):1–61.

Wilsdorf, P., M. Dombrowsky, A. M. Uhrmacher, J. Zimmermann, and U. van Rienen. 2019. "Simulation Experiment Schemas–Beyond Tools and Simulation Approaches". In *2019 Winter Simulation Conference (WSC)*, 2783–2794. Institute of Electrical and Electronics Engineers, Inc.

Wilsdorf, P., F. Haack, K. Budde, A. Ruscheinski, and A. M. Uhrmacher. 2020a. "Conducting Systematic, Partly Automated Simulation Studies - Unde Venis et Quo Vadis". *AIP Conference Proceedings* 2293(1):020001.

Wilsdorf, P., F. Haack, and A. M. Uhrmacher. 2020b. "Conceptual Models in Simulation Studies: Making it Explicit". In *Proceedings of the 2020 Winter Simulation Conference*, edited by K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Yu, E., and J. Mylopoulos. 1998. "Why Goal-Oriented Requirements Engineering". In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, Volume 15, 15–22.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Academic press.

Zhang, J., and B. H. Cheng. 2006. "Model-Based Development of Dynamically Adaptive Software". In *Proceedings of the 28th international conference on Software engineering*, 371–380.

AUTHOR BIOGRAPHIES

TOM MEYER is a Ph.D. candidate in the Modeling and Simulation group at the University of Rostock. His e-mail address is tom.meyer@uni-rostock.de.

ANDREAS RUSCHEINSKI is a Ph.D. candidate in the Modeling and Simulation group at the University of Rostock. His e-mail address is andreas.ruscheinski@uni-rostock.de.

PIA WILSDORF is a Ph.D. candidate in the Modeling and Simulation group at the University of Rostock. Her e-mail address is pia.wilsdorf@uni-rostock.de.

ADELINDE M. UHRMACHER is a professor at the Institute of Visual and Analytic Computing, University of Rostock, and head of the Modeling and Simulation group. Her e-mail address is adelinde.uhrmacher@uni-rostock.de.