Towards Systematic Testing of Complex Interacting Systems

René Schumann, Caroline Taramarcaz

Smart Infrastructure Laboratory, Institute of Information Systems
University of Applied Sciences Western Switzerland (HES-SO) Valais/Wallis
Rue de Technopole 3, 3960 Sierre, Switzerland
rene.schumann | caroline.taramarcaz@hevs.ch

Abstract. In this paper we introduce the notion of adaptive systems, which can change their behavior during run-time. Thereby change is neither randomly nor directly pre-programmed. The change is triggered by a data processing procedure based on data acquired during the run-time of the system. In that sense the adaptation can be considered directed and unpredictable. While we have accepted, and mostly reacted, to systematic risk originating in our infrastructure, e.g. black-outs, or communication failures, we argue that adaptive systems create a new type of error-behavior that cannot be handled by classical techniques for testing systems. Here we introduce the idea of structured simulation as a means for testing the behavior of adaptive systems.

1 Introduction

We consider that in future interconnected systems the majority of IT systems will be adaptive. We consider a systems to be adaptive, when it has the ability to adjust its behavior during run-time in a not necessary predicted, but goal-oriented way. Thereby it is not relevant which technique is used for implementing the adaptive system. Recent techniques for implementing such types of systems are, for instance, Machine Learning based systems. Following the widely discussed rise of AI and its forms of applications, e.g. Industry 4.0, those adaptive systems are likely to be embedded in more and more IT systems or cyber-physical systems in our environment. In environments populated by such systems the systemic risk of interconnected, adaptive systems is actually unknown, and can hardly be estimated.

We argue that the behavior of these systems can expose a undesired behavior due to mis-adaptation, which represents a new type of error and also create a new type of systematic risk, which we discuss in more detail in Section 2. As we consider this type of risk not to be avoidable, if we want to benefits from the potentials of adaptive systems, we have to consider new types of mitigating and reducing this new type of risk. Therefore we discuss novel way of testing adaptive systems. This testing has to show on the one hand the system at hand provides the behavior patterns we are hoping for, while also validate that undesired behavior is not likely to appear in a future usage of the system, or show

under which conditions mis-adaptation could occur, to provide feedback to the developers to improve the system at hand. To do so, we introduce the idea of structured simulation in Section 3 and present a software framework enabling that can support the application of structured simulation. However, we will also show that not only for the testing of adaptive systems, but also *conventional* software system testing, e.g. a traffic management system, can benefit from using the structured simulation approach and framework. Finally we conclude on what has been achieved so far, and what remains a research challenge, for the moment.

2 Adaptive Systems as a systematic risk

In this section we are going to detail how adaptive systems can expose an error type that has been not in system design and testing so far. Therefore however, it is necessary to outline what adaptive systems are, and how they differ from *conventional* systems. *Conventional* systems can be considered as deterministic, the system behavior, its output, depends on a given input sequence. Also for most applications of e.g. Machine learning, a predictor/ classifier is trained, and afterwards but into use, it is typically expected that it will provide the same result, given the same inputs. From an abstract point of view, the system in that sense works as a finite state machine, processing the input. Given a certain initial state, with each input the system transients to a different internal state, and after one or several inputs along the sequence the system shows a certain, typically desired, behavior. For the processing of the next input sequence the machine is assumed to start processing again in the initial state. This situation can be considered as a tree, with a root as the initial state and leaf as those where the system shows the behavior. Due to the large variety the tree can have a rather high branching factor, and is potentially even infinite. In case of finite trees, the system behavior can be investigated either by exploring all leaves, or if these are too many by some type of Monte Carlo investigation. This situation is simplified in Figure 1.

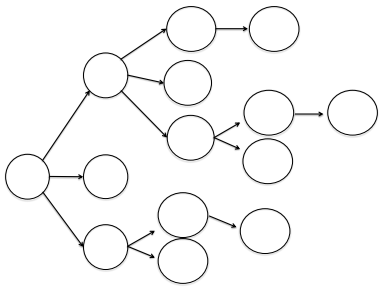

Fig. 1. Representation of the state space of possible worlds for a finite graph

Note, of course testing of those systems also today falls short, as not necessary all relevant parts of the input are within the scope of the mental model of the developer / tester. For instance, a time out of data base connection can cause an unforeseen error, if the option of the database not been available has not been considered during test and development of the system. This is, however it is an important issue of all types of testing and not related to the type of system. However, as for the aspect of the incompleteness of the mental-model is not specific for adaptive systems, but is more generic. we are not going to address this point in the following discussion.

Adaptive systems however, cannot be considered as deterministic systems, i.e. their behavior is in a limited sense non-deterministic. Thus, executed twice with identical parameters, it cannot be guaranteed that the response of the system is identical. The idea of adaptation requires that the system has a constant feedback loop with its environment, i.e. its inputs- and outputs of the past influence its internal state, and therefore its reactions to to future inputs from its environment. This is for instance the case if we consider systems is creating its learned model dynamically, i.e. non-stationary learning, see e.g. [3].

Definition 2.1

Definition: An **adaptive system** is a system that is capable of adapting its behavior at runtime based on its previous reaction to its environment, and if this in beneficial to its internal goals, whereby this is considered by an internal reasoning process, and its internal state.

Thus, considering the difference in the way of Figure 1, we have to state that the resulting tree becomes infinite, as there is no restarting of processing at the intial state to process the second input. Actually, each possible sequence becomes infinite, as it is shown in Figure 2. Consequently, also validation via a complete enumeration of all leaves becomes impossible. Adaptation is performed

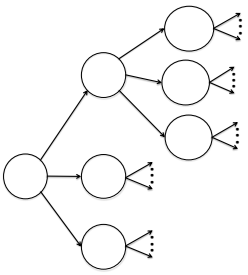

Fig. 2. Representation of the state space of possible worlds

by implementing a feedback loop in the system to increase its performance. Thus, adaptation is neither randomly nor directly pre-programmed. In that sense the

adaptation can be considered directed and unpredictable at design-time. So, while for a single decision, without a-priori knowledge of all previous in- and outputs, the system's behavior can be considered as non-deterministic. However, given the entire sequence of the system interactions with its environment up to a certain point that is subject to analysis, the system can be considered deterministic.

As it is typically not clear at design-time, which possible input sequence, out of the virtually infinite number of possible sequences, the system will face,system's behavior cannot be specified for all states (nodes in Figures 2). This situation has been actually the reason why the usage of adaptive systems has been, among others, has been promoted, e.g. applying Multi-Agent techniques in the field of service oriented-computing [5, 6].

However, while the adaptation mechanism might create more desirable system's behavior in a large number of situations, it cannot be guaranteed that in each possible world the system might operate in, its reaction is acceptable. Also testing the system is not sufficient. Tests can show that for a (potentially large) number of (finite) sequences a system shows desirable behavior. However, it cannot show the absence of a sequence in which the adaptation is performing in a non-acceptable way. In this sense the integration of adaptive systems imposes a potential risk to the overall system, compared to conventional systems.

Despite the integration of potential risks, adaptive systems are becoming more and more relevant, as for instance AI-based systems enable behavior adjustment are becoming more and more relevant for recent applications [3]. In fact, there exist a number of techniques allowing for the implementing of adaptive behavior. However, for the following discussion it is not considered relevant, which technique is used.

While at this point in time the idea of risks induced by adaptive systems seems to be rather theoretical, we actually can observe the first instances occur, even though they are, at the moment, not discussed in brought public, and maybe also not recognized as generic problem of adaptive systems. Among the most prominent fails of AI in2018[1] there is the case of a system developed for Amazon's HR department, to provide recommendations about suitable candidates to hire. The system was initially trained with a large database from former applicants of the company. So, while the Machine Learning algorithm used inside the system has been actually implemented correctly, the developers and users had to learn that the system developed a sexual-bias, as it preferred male applicants over female. Another popular example has been the chatbot from Microsoft, which had adapted its behavior during the chats, exposing more and more *racism behavior* [2]. Both examples show that system with adaptive behavior are in danger to show undesired adaptation. In both cases the bias was not in the initial implementation of the learning algorithms, but the bias was

[1] see `https://medium.com/syncedreview/2018-in-review-10-ai-failures-c18faadf5983` (latest accessed on the 05.04.2019)

[2] `https://techcrunch.com/2016/03/24/microsoft-silences-its-new-a-i-bot-tay-after-twitter-users-teach-it-racism/` (latest accessed on the 05.04.2019)

introduced within the input given to the system at runtime, to which the system adapted. While both have been isolated system, it already provides a showcase for potential problems that are introduced with adaptive system.

So while designers of systems have control on the adaptation mechanism, and also its initial training (stationary learning/adaptation) of the system. However, for the adaptation happening at runtime (non-stationary learning/adaptation) there seems to be so far no mechanism to avoid undesired adaptation, which is also hard, as it is non-deterministic, both in time and direction. However, for applications that do for instance monitoring networks this is an essential capability, as they learn at runtime to distinct *normal* network traffic from exceptions, which each time has network specific aspects.

3 Structured Simulation

In the following we discuss the implications on testing for adaptive systems. We do not consider here methods for the formal analysis, even though there have been attempts using model-checking for adaptive systems. Those techniques often depend on the actual implementation techniques, e.g. Multi-Agent Systems [7] , or Machine Learning methods [3] . Furthermore, formal aspects often have issues when the system to validate contains a model of its environment, as then the model of the environment needs to become part of the formal model, which is not only adding a barely handleable complexity, but also the correctness of the domain model becomes an issue. Furthermore we also do not discuss on how self-X abilities could be used, first discussions on how to approach this have been outlined in [4].

Here, we consider a systematic investigation of the system's behavior to analyze potential occurrence of undesired adaptation. We consider this is a first and important step, to have a systematic way to point developers to the relevant design decisions, i.e. in cases when an undesired adaption might happening.

As we have outlined in the previous section, and in particular with Figures 1 and 2, the testing needs to be take into account that it is not sufficient to test isolated states, but sequences of inputs. These sequences must be constructed step-wise as they are infinite. Considering the observation that a virtual infinite number of infinite long sequences of inputs should be tested, it becomes clear that what we present here is a heuristic, that tries to take advantage of the fact that, given a unique initial state of the system to test, the number of states is countable infinite, which enables us to provide a formal way to enumerate the states.

Independent on what adaptive system actually needs to be tested, it needs to be tested within a context, that allows for providing inputs to the system in a controlled way. This in fact can be considered a form of simulation environment in which the adaptive system is tested. Thus, in the following we consider that the systematic testing becomes equivalent to the execution of a systematic run of a simulation study, consisting of a larger number of simulation runs.

The idea is to start from a fixed initial state, for which in the best case the applied simulation environment has been calibrated. Starting from this specification of the system inputs, for which typically the system has already been tested, the inputs shall be gradually changed by applying *modifiers* leading to different potential future worlds. By applying multiple modifiers, or one several time, the overall space of possible inputs for the system can be generated systematically. This is shown in Figure 3. Furthermore we need to provide a method

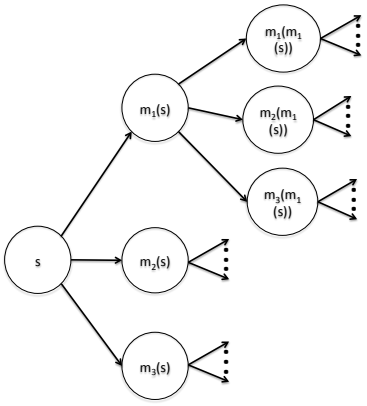

Fig. 3. Representation of the state space of possible worlds for a *base* scenario `s` and the three modifiers m_1, m_2, m_3, as presented in [4]

to systematically explore the states of the environment sketched in Figure 3. We do so by using the idea of probability a state might occur. We therefore assume that each state has a probability it will occur, and each *modifiers* has a probability of its occurrence. Thus, given the probability of a state and the probabilities of the occurrence of the different *modifiers*, the probability of occurrence of all sequential states can be computed. Note, a current limitation in this concept is that for the moment the occurrence of all *modifier* are considered to be mutual independent. Assuming the probability of the initial state as 1, as it has been ideally calibrated with real-world data, the probability of all states can be computed and is for each sequence is strictly declining. All states can now be systematically explored with declining probability of occurrence. Thus, a modified version of Dijkstra's Algorithm can be used to provide a procedure enumerating the states. The probability of a sub-sequence can be considered as equivalent to the probability of its last state.

3.1 The StructuredSim Framework

We have been started to implement the ideas in the *Structured Sim*[3] framework. The goal of the framework is to provide a way to systematically plan, execute and analyze simulation studies. The framework has been not been developed for any specific simulation software, thus some glue code needs to be written. The framework consists of three main parts following the overall flow is shown in Figure 4, while the more detailed interfacing idea of the simulator of the StructuredSim framework is depicted in Figure 5.

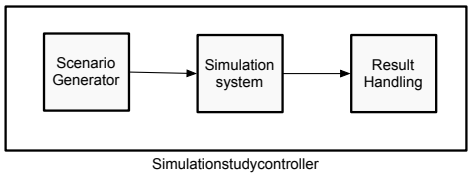

Fig. 4. Overall idea of the components required for the structured simulation, as presented in [4]

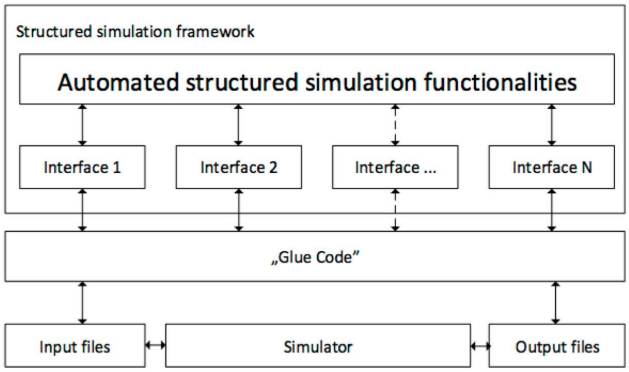

Fig. 5. Conceptual sketch for the interfaces between a simulator and the StructuredSim framework

During the development of the framework, it has been used with two different simulation environments, to ensure its general structure. Together with colleagues from the University of Zagreb we have implemented a link to the com-

[3] `http://silab.hevs.ch/structSim/structsim.html`
(latest accessed on the 05.04.2019)

mercial traffic simulation tool VISSIM[4], which is used by the traffic researchers to validate traffic management approaches, taking advantage of Machine Learning approaches [1]. The second simulator was a *Game of life* simulation [2].

In both cases the evaluation has been done for each state in isolation. This can be considered as an alternative to currently established Monte-Carlo simulations. Monte-Carlo studies require typically a particular large number of executions. As the environment of the system to test is simulated, this requires a large number of simulation runs, which makes this type of analysis prohibitive costly in time of computation times, as often a single simulation run can take several minutes computing time.

The structured simulation approach can offers here an alternative. As the different states of the environment are investigated in descending order of their probability enables for the *any-time* property. This means the exploration can be stopped at any time, and given the available amount of computation time, it is guaranteed that the most likely states have been explored up to this point. Alternatively one could specify a certain threshold up to which all states with a higher probability should be investigated. This way it is possible to specify up to which level of confidence the states should be investigated. Compared the current rather blind search (Monte-Carlo) this compares more to a directed search approach.

The current implementation of the StructuredSim Framework however has a further significant limitations. As pointed out states are systematically explored, but the resulting simulation control is still state-based. As we have argued above, we would need for the proper evaluation of adaptive systems. Thus, we have to consider sequences of states, not individual states. While we have integrated this into the conceptual framework, it still needs to be implemented. However, due to limitations in runtime, we still require to limit the total number of tests, and therefore our approach remains heuristics for testing, and further work will be required implementing techniques to avoid adaptive systems develop this undesired adaptation, however, to do so, we first need to further elaborate on the tooling for testing this type of systems, to approach the resulting risk of undesired adaptation in a systematic way.

References

1. Greguric, M., Ivanjko, E., Schumann, R., Taramarcaz, C.: Structured simulation: A framework for the automated analysis of adaptive systems (October 2015), presented at the TUD 1102 COST ARTS Final Conference, October 6th – 7th 2015., Bordeaux, France
2. Kraft, D.: Game of Life Simulation. Ph.D. thesis, Institute of Information Systems, University of Applied Sciences Western Switzerland. (2017), bachelor thesis
3. Sayed-Mouchaweh, M., Lughofer, E.: Learning in Non-Stationary Environments: Methods and Applications. Springer-Verlag, New York (2012)

[4] `https://www.ptvgroup.com/en/solutions/products/ptv-vissim/` (latest accessed on the 05.04.2019)

4. Schumann, R.: Performance maintenance of ARTS systems: Finding and managing performance deterioration by undesired adaptive behaviors, pp. 181– 195. Autonomic Systems (ASYS), Springer International Publishing (2016)
5. Singh, M.P., Huhns, M.N.: Service Oriented Computing. John Wiley & Sons Ltd., West Sussex (2005)
6. Timm, I.J., Hormann, A., Scholz, T.: Qualitätssicherung serviceorientierter Architekturen, pp. 115 – 139. Deutscher Universitäts-Verlag, Wiesbaden (2007)
7. Wolf, T.D., Holvoet, T.: Emergence and self-organisation: a statement of similarities and differences. In: Brueckner, S.A., Serugendo, G.D.M., Karageorgos, A., Nagpal, R. (eds.) Proceedings of the International Workshop on Engineering Self-Organising Applications. pp. 96–110. Springer (2004)