A Framework for Testing Self-organisation Algorithms

Benedikt Eberhardinger, Gerrit Anders, Hella Seebach, Florian Siefert, and Wolfgang Reif
Institute for Software & Systems Engineering, University of Augsburg, Germany
{ eberhardinger, anders, seebach, siefert, reif }@isse.de

Abstract. The characteristics of self-organising, adaptive systems (SOAS) lead to a significantly higher flexibility and robustness against an ever-changing environment. This flexibility makes it hard to test these systems adequately, which is, however, inevitable in order to assure their quality. As a part of our vision of testing SOAS, we present a framework for testing self-organisation (SO) algorithms. The framework addresses the following key challenges for testing SO algorithms: state space explosion, interleaved feedback loops, and failure overlapping.

Keywords. Self-organising Systems, Testing

1 Testing Self-organising Systems

Self-organisation and adaptation allow systems to fulfil their goals despite an ever-changing environment by enabling them to restructure at run-time. This new flexibility leads to challenges in engineering these systems. For example, the behaviour cannot be fully specified at design-time and adaptation decisions are moved to run-time. This makes it hard to assure that those systems fulfil their requirements, but makes it even more necessary to take appropriate measures. In this context, testing plays a decisive role. The crucial question is how to deal with the size of the state space of a self-organising, adaptive system (SOAS). In our vision of testing SOAS [3], we presented a layered approach to address the challenges of testing these kind of systems. As a part of this approach, we present a framework for testing self-organisation algorithms.

2 IsoTeSO: A Framework for Isolated Testing of SO Algorithms

Self-organising (SO) algorithms are characterised by their inherent indeterministic behaviour, an ever-changing environment, a high degree of interacting components, as well as their interleaving feedback loops. Consequently, testing is faced with the three following challenges, among others: (C1) In SOAS, different SO algorithms interact with and influence the system components, possibly resulting in interleaved feedback loops. These are challenging in testing, because it is hard to get dedicated results for a single SO algorithm. (C2) After a reorganisation, the result provided by an SO algorithm has to be applied to the existing system structure. Under certain circumstances, it is not possible to detect an invalid reorganisation result if only the resulting system structure is evaluated. Therefore, there is a chance of failure overlapping. (C3) A testing approach has to deal with a huge state space that results from the inherent indeterministic behaviour of SO algorithms and the ever-changing environment of SOAS.

2.1 Architecture of the Framework

Our Framework for Isolated Testing of Self-organisation Algorithms (IsoTeSO) is designed to meet the challenges (C1), (C2), and (C3). In the following, we show how we achieve this. An overview of the framework is given in Fig. 1.

Using IsoTeSO, SO algorithms are isolated from other parts of the system, including other SO algorithms. For this purpose, we provide a fully controlled test environment. We address (C1) by obtaining dedicated results for the SO algorithms under Test (SOuTs) by fully stubbing their environment. For this purpose, the execution component provides a system simulator and a grey-box interface for the SOuT. Within the system simulator, an environment simulator stresses the SOuT by changing the environmental conditions over time. These situations are derived from a functional, stochastic test model (details can be found at the end of this section) used to automatically generate the test runs.

As mentioned before, the SOuT is plugged in the test system via a grey-box interface that enables the evaluation of internal states of the SOuT, for example for checking interim results. Our evaluations showed that this principle allows to reveal overlapping failures with regard to SO algorithms (cf. (C2)). The information needed by the oracle to detect these overlapping failures are provided in the form of interim results by the SOuT. The evaluation of the validity of reconfigured system structures is performed by checking the violation of constraints that describe valid solutions. These constraints represent the requirements of the SOAS (cf. [4]). Thus, it is possible to automatise the evaluation conducted by the oracle, resulting in a fully automatised test process.

To derive appropriate test runs for execution and evaluation, we use a functional, stochastic test model. By using this test model, we are able to extract likely as well as critical traces from the state space (cf. (C3)).

The SOAS we consider are based on interactions within their ever-changing environment: The environment is influenced by the system, and the system also influences the environment itself. In case of stigmergic approaches, the environment can even be used by the system components as communication medium. We use a stochastic notation for modelling the behaviour of the environment with respect to a specific context.

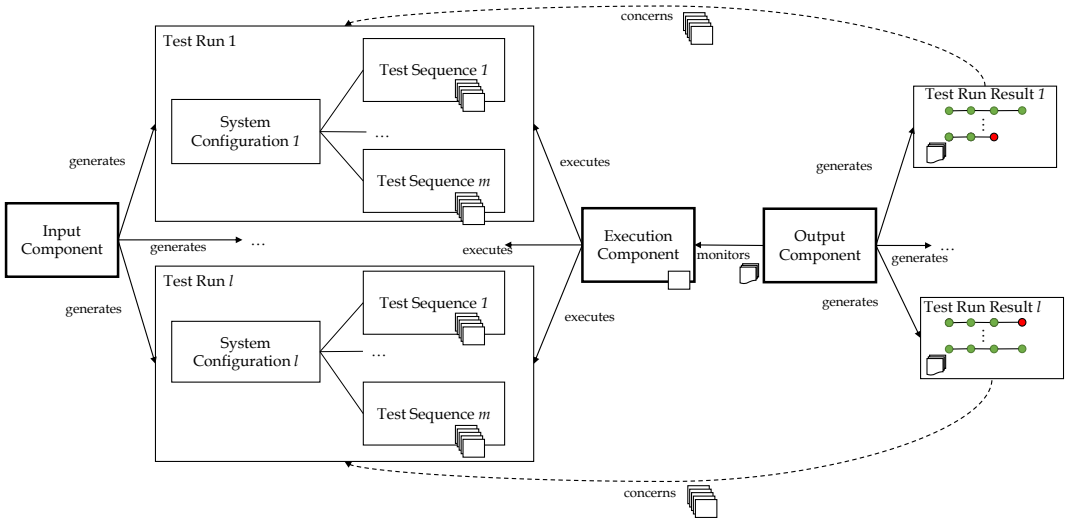

Figure 1: IsoTeSO consists of: (1) the input generation component, (2) the execution component, and (3) the output component. (1) generates l test runs, each consisting of an initial system configuration and m test sequences. Each test sequence contains n test cases. Test runs are performed individually by (2). (3) monitors the execution and samples the reorganisation results as well as the system structure for each test case. For each test run, the results of the test cases are evaluated (indicated in red and green). The results of all test sequences of a specific test run form IsoTeSO's output.

This concept is related to the theory of operational test profiles highlighted in [5]. For designing these Environment Profiles (EPs), we use Markov chains that describe different context-based states of the environment. The context is defined by a group of components, i.e., agents, as well as their location in the environment. For modelling EPs in a smart grid application, an exemplary context could be defined for a group of solar power plants in a particular solar farm. Decisive states for solar power plants mainly depend on the current weather condition.

To create the system configuration of a test run information about the static properties of the SOuT's environment as well as a specification of the SOuT itself is needed. This information and the information about the environment's dynamic behaviour encoded in the EPs form the test model of IsoTeSO.

3 Application of IsoTeSO

We applied the IsoTeSO framework to two partitioning-based SO algorithms: (1) a Particle Swarm Optimizer [1] and (2) a decentralized SO approach called SPADA [2]. These algorithms are deployed in a smart grid application, which is the foundation for designing the test models. For testing, we implemented an architecture according to the concepts of the IsoTeSO framework discussed in Sect. 2. By injecting specific faults, we were able to show that the framework can find different kinds of failures, e.g., faulty adaptation of a computed solution or invalid partitionings.

4 Conclusion

We outlined a framework for testing SO algorithms that copes with the complexity arising from the flexibility of SOAS. Furthermore, we showed how to meet challenges like (C1) interleaved feedback loops, (C2) failure overlapping, and (C3) state space explosion within IsoTeSO. To master these challenges, we isolated the SO algorithms to separate the feedback loops, applied a grey-box approach to detect overlapping failures, and used our test model to extract likely as well as critical traces out of the state space.

References

[1] G. Anders, F. Siefert, and W. Reif. A particle swarm optimizer for solving the set partitioning problem in the presence of partitioning constraints. In *Proc. 7th Int. Conf. Agents & AI (ICAART)*, 2015.

[2] G. Anders, F. Siefert, J.-P. Steghöfer, and W. Reif. A decentralized multi-agent algorithm for the set partitioning problem. In I. Rahwan et al., editor, *PRIMA 2012: Principles and Practice of Multi-Agent Systems*, volume 7455 of *LNCS*, pages 107–121. Springer, 2012.

[3] B. Eberhardinger, H. Seebach, A. Knapp, and W. Reif. Towards testing self-organizing, adaptive systems. In M. Merayo et al., editor, *Testing Software and Systems*, volume 8763 of *LNCS*, pages 180–185. Springer, 2014.

[4] B. Eberhardinger, J.-P. Steghöfer, F. Nafz, and W. Reif. Model-driven synthesis of monitoring infrastructure for reliable adaptive multi-agent systems. In *Proc. 24th IEEE Int. Symp. Software Reliability Engineering (ISSRE'13)*, pages 21–30. IEEE, 2013.

[5] C. Smidts, C. Mutha, M. Rodríguez, and M. J. Gerber. Software testing with an operational profile: Op definition. *ACM Comput. Surv.*, 46(3):39:1–39:39, 2014.