Ontology-Based Test Case Generation For Simulating Complex Production Automation Systems

Thomas Moser, Gregor Dürr and Stefan Biffl
Christian Doppler Laboratory for
Software Engineering Integration for Flexible Automation Systems
Vienna University of Technology, Austria
{thomas.moser, gregor.duerr, stefan.biffl}@tuwien.ac.at

Abstract—**The behavior of complex production automation systems is hard to predict, therefore simulation is used to study the likely system behavior. However, in a real-world system many parameter variants need to be tested with limited resources. Therefore, test cases need to be generated in a systematic way to find suitable scenarios efficiently. This paper investigates the effort of two approaches for providing test cases based on available testing knowledge. The traditional approach uses a static generator script based on implicit testing knowledge, which takes significant effort to add new parameters. The innovative approach uses a dynamic generic generator script based on an ontology data model of the testing knowledge. We empirically evaluate these approaches with a use case from the production automation domain. Major result is that the high-level test description of the ontology-based approach takes more initial effort for setup, but increases the usability and reduces the risk of errors during the test case generation process.**

Keywords - test case generation, ontology, production automation simulation, explicit testing knowledge.

I. INTRODUCTION

Production automation systems are often complex as the behavior of the overall system cannot easily be predicted from the behavior of the subsystems. Therefore, simulation is used to study the behavior of complex production automation systems. In addition to simulation accuracy the simulation system performance is an important issue, particularly if many parameters and value ranges for system behavior need to be evaluated systematically [6]. Example test parameters for assembly lines in production automation are scheduling strategy, failure handling strategy, and the number of products.

Software testing investigates the quality of the product or service under test. In order to achieve sufficient test coverage for the requirements of an application, there must be at least one test case for each requirement and relevant parameter setting, which can take considerable effort. A major goal is to generate test cases in a systematic way to efficiently find suitable scenarios for most of the requirements with limited testing resources. In this paper, the test cases define input data to simulate complex distributed assembly line systems with a simulation tool [7, 8, 13].

This paper presents an approach for extracting expert testing knowledge into an ontology and for using this explicit knowledge to efficiently generate test cases for a production automation simulator [7, 8, 13]. For this purpose, the test coverage combined with the cost to achieve this test coverage is used as performance metric. In our context, test coverage is the ratio between the number of generated test case scenarios and the number of all possible test case scenarios regarding a given set of parameters respectively their possible parameter values.

This paper investigates two approaches for providing test cases. The traditional approach implements a test as a static generator script, which is fairly simple and quick to start up, but takes significant effort to add new parameters, since all existing scripts need to be updated to accommodate the new parameter settings. In addition, the tester needs programming skills both for setting and modifying parameters.

The innovative approach models system and test know-how in an ontology-based data model and dynamically adapts a generic generator script according to the settings in the test data model. Test cases are generated and run with respect to the parameters chosen by the user. Important advantages of this approach are (a) available efficient tool support for modifying ontologies in case of changes to the parameters and/or structure of the system under test and (b) the fact that the generator script is not affected by the modification. Therefore, even testers without programming skills can add new parameters by modifying the underlying data model. Test cases are generated from the ontology and exported as XML file. The implemented generator script and the ontology are loosely coupled. Therefore, changes to the ontology do not necessarily lead to changes of the generation script. This design approach enables a flexible and high-level test description.

The evaluation part explores how the ontology-based approach can reduce the cost for test description, enables a more efficient and effective generation of test cases while aiming at an exact definition of test coverage to be achieved, and improves the changeability of the test case generation approach (e.g., the introduction of new test parameters) due to lower efforts to implement new test case parameters.

The remainder of this paper is structured as follows: Section 2 summarizes related work on software testing, production automation simulation, and on ontology-based test case generation. Section 3 identifies the research issues and summarizes the use case. Section 4 introduces the ontology-based test case generation approach, while section 5 presents the evaluation results. Section 6 discusses the evaluation results with regard to the research issues, and finally section 7 concludes the paper and identifies further work.

II. Related Work

This section presents related work on automated software testing. Furthermore, the production automation simulation systems MAST and SAW are shortly introduced. Finally, related work on ontologies and on ontology-based test case generation is summarized.

A. Automated Software Testing

Testing of software is an important part of every software life cycle to demonstrate the capability of the software and identify defects before operational use. The National Institute of Standards and Technology (NIST) reported in a 2002 study that software bugs cost the U.S. economy around 60 billion dollar per year [10]. The same study showed that these costs could be reduced by more than a third by improving testing. In the IEEE Standard on the Software Life Cycle Process, a process is defined as a set of activities; process groups represent a higher level of abstraction [1].

A test process can be seen as the systematic execution of a software program. Test management and running the test object with specific data are the important parts of the test process. The goal of the test management is to plan, execute, and analyze the test run. A test run consists of one or more test cases [11].

B. Production Automation Simulation

The Manufacturing Agent Simulation Tool (MAST) is an agent-based solution for some typical manufacturing tasks by using multi-agent technologies for manufacturing control [13]. Flexible distributed manufacturing systems can be modeled as intelligent, autonomous and cooperative agents [12], where each agent manages the local behavior to meet goals locally without centralized control [13]. In addition, the agents interact to achieve the system goal cooperatively.

The Simulation of Assembly Workshop (SAW)) [7, 8], which is an extension of the original MAST system, investigates processes, methods, and tool support for planning, coordination, simulation, and lab tests for work shifts in an assembly workshop. Generally, a workshop aims to effectively and efficiently carry out the work orders in a work shift. SAW helps to understand the impact of tactical decisions in the production automation environment. For instance, the user can optimize his assembly line by analyzing the effect of the different strategies. The SAW project provides a simulator that can execute the test cases for the production automation domain. In addition, the logged events during the simulation allow analyzing the simulation object under test. Thus, the results of the simulation enable checking the post condition of the test case. The explained life cycle of a test case is adopted for the simulation process of the SAW project during the elaboration of this work.

C. Ontologies for test case generation

In general, ontologies are a main part of the semantic web technology and facilitate the knowledge representation of real-world concepts. Ontologies are formal models of a specific application domain, and primarily used to facilitate the exchange and partitioning of knowledge. More precisely, an ontology is a data model that represents a set of concepts within a domain and their relationships. The word ontology has its origin from the Greek words *ontos* (=being) and *logos* (=word). From a philosophical point of view an ontology refers to the subject of existence, that is the study of being as such [3]. Gruber [4] defines an ontology as an explicit specification of a conceptualization. Where a conceptualization illustrates an abstract, simplified picture of the world used for representation and designation. Each knowledge representation follows a certain degree of conceptualization, either explicitly or implicitly. Moreover, ontologies can effectively support software development processes by providing a continuous data model [2].

Ontologies could help generate basic test cases since they encode domain knowledge in a machine processable format. A simple example for this would be regarding cardinality constraints. Since those constraints define restrictions on the association of certain classes, they can be used to derive equivalency classes for testing [5]. Ontologies may not be the first candidate for such a scenario, since there are formalisms like OCL that are specialized for such tasks. However, once domain knowledge is available in an ontology format, it might be feasible to reuse that knowledge. Nguyen et al. [9] describe a framework for automated test case generation in the context of multi-agent systems. They use agent interaction ontologies that define content semantics of agent interactions to generate test inputs, guide the exploration of the input space during test case generation, and verify messages exchanged between agents with respect to the agent interaction ontology.

III. Research Issues and Use Case

The generation of test cases based on available testing knowledge is an import factor for the simulation of complex systems, such as production automation systems. This paper investigates two approaches for providing test cases. A traditional approach based on manually derived static generator scripts, which takes significant effort to add new parameters. In addition, the users need programming skills for both setting and modifying parameters. The ontology-based approach uses a dynamic generic generator script based on an ontology as data model. Test cases are generated with respect to the chosen parameters by the user.

This work proposes an ontology as suitable data model to provide the necessary data to generate a suite of test cases with an ontology-based approach for a given set of parameters. This claim is also met by the static approach. In other words, the fact that the ontology-based approach is newer

and works as well as the old static one is not enough motivation for change. However, measurable benefits are essential to accept the new ontology-based approach and the change costs. As with every new engineering method an important question is what aspects of the new approach are comparable to or better than a best-practice traditional approach. Therefore, we derive the following research issues.

RI-1. Feasibility of the ontology-based test case generation approach. The ontology-based approach aims at overcoming limitations of the traditional static approach. Firstly, the new approach should provide a high-level test description to allow the target audience to generate test cases with less effort. Secondly, a validation check and consistency check of the parameter setting is essential to reduce the risk of making mistakes during the configuration phase of the test case generation process.

RI-2. Cost-benefit potential of ontology-based test case generation approach. The following two sub research issues define a metric how the cost-saving potential can be measured with respect to the effort for test description and the effort for setting up the solution. RI-2a assumes that both the number of parameters to choose and the number of supported data types are constant, while RI-2b addresses the cost for adding new parameters with different data types.

IV. Ontology-Based Test Case Generation Approach

This section describes the ontology-based test case generation approach. In the first subsection, a short overview of the simulation systems underlying ontology data model is given, while the second subsection describes the test case generation suite.

A. Simulation System Data Model

The SAW system [7, 8] uses a 5-layer ontology as underlying data model. The five layers of the simulation system and their relationships among each other are illustrated in Figure 1. In the following, each layer is described shortly. The *business layer* prioritizes all incoming orders with respect to their due date. In the *shift layer* a capacity check of the resources needed for producing the ordered product takes place. In addition, the business orders get transformed to work orders. In the *job shop layer* the work orders get broken down to single tasks by the production strategies. Afterwards, the responsible production strategy orders the tasks with respect to their specific criteria. The load balancer in the *operation layer* is responsible for a well-balanced utilization of the available machines. Furthermore, the shortest path consisting of one or more conveyor belts is calculated to fulfill the different tasks. *The master data layer* contains all information, which remains rather constant during the simulation such as the arrangements of the conveyor belts, the arrangements of the machines, and the structure of the product trees. All other layers can refer to this information.

In order to provide the ontology-based test case generation functionality, we extended the original SAW ontology data model by adding the so-called *test case layer* to the ontology data model. The *test case layer* together with a suitable test case generator script provides high quality test cases in an automated and systematic way. As a consequence high test coverage can be achieved, which is essential for testing the performance of simulation systems. The user can decide if she wants to create test cases manually or with the help of a generator script. In case of using a generator script the user just has to configure the parameter setting which contains the test case parameters and the corresponding set of valid values. Afterwards, the generator script generates the test cases with respect to the chosen parameter setting.

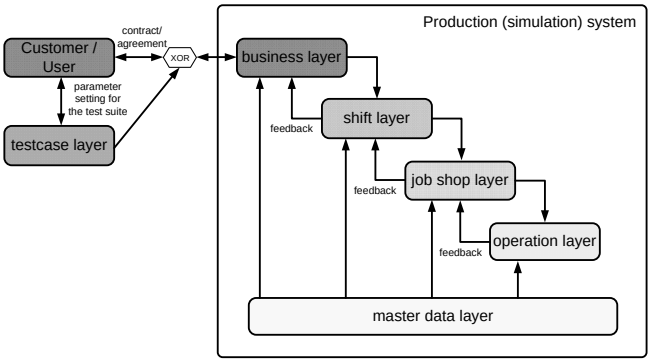

Figure 1. Layers of the SAW systems ontology data model.

B. Test Suite Generation

Initially, the ontology-based approach requires a GUI that can be dynamically adapted with regard to the available test case parameters. The ontology-based approach extracts the offered test case parameters, the allowed values for the parameters, and structural information from the underlying data model. Afterwards the identified information of the ontology is passed to the GUI. This data flow makes it possible to build a dynamic GUI which corresponds to the ontology at runtime. As a consequence, modifications to the ontology do not necessarily lead to manual changes neither to the GUI nor to the dynamic generic script.

On the one hand the ontology-based approach uses the underlying ontology's structure and restrictions to ensure the consistency of the test cases. On the other hand the data ranges of the offered parameters are used to ensure the validation of the test cases. Nevertheless, the user has to manage the underlying ontology. Therefore, the user needs skills for modifying the ontology with common graphical editor tools like Protégé[1]. Of course, for modifying an ontology users need some experience but it involves less effort than modifying a hard-coded script. Furthermore, if something goes wrong during the modification of the ontology the user will recognize it immediately in the parameter setting configuration process. On the contrary, the user might realize that something went wrong during the modification of the static script after the simulation run took place by analyzing the simulation results. This is a frustrating experience as users usually do not know what went wrong. In that case the user is captured in a trial and error loop. Surely, these circumstances negatively affect the acceptability of the static approach.

[1] http://protege.stanford.edu/

By using the ontology-based approach, the user can always choose from all test case parameters supported by the generator script to define the parameter setting. In addition, the ontology-based approach ensures that only valid and consistent parameter settings can be defined by the user. This circumstance can be achieved with the information of the ontology. Afterwards, the ontology-based script generates test cases based on the parameter setting. At the end of the path the dynamic generic script generates the XML file. A useful characteristic of the "3 Phases Process Model" shown in Figure 2 is the fact that the first phase and the second phase are totally independent of the domain. This is ensured as the domain specification is hidden in the ontology. The first phase communicates with the ontology and passes the element names, valid values for element instances, and information about the structure of the ontology to phase two. After this step the second phase uses the received information about the ontology to build a dynamic GUI at runtime. The user can configure the third phase based on the data model easily.

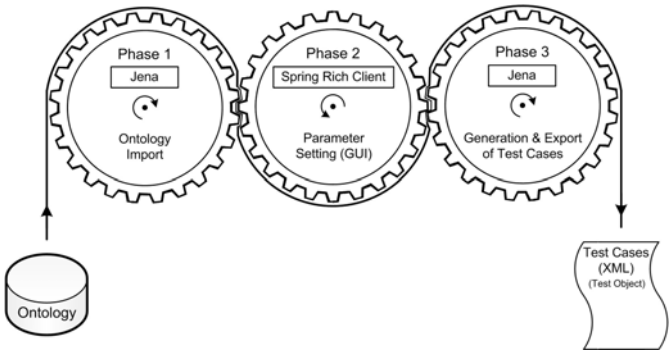

Figure 2. 3 Phases Process Model of the ontology-based approach.

V. EVALUATION

This section focuses on the evaluation concept and explains in detail the criteria for the different objectives. The objectives are the costs for the test description, the effort for implementing test case parameters, and whether the test coverage is definable by the user.

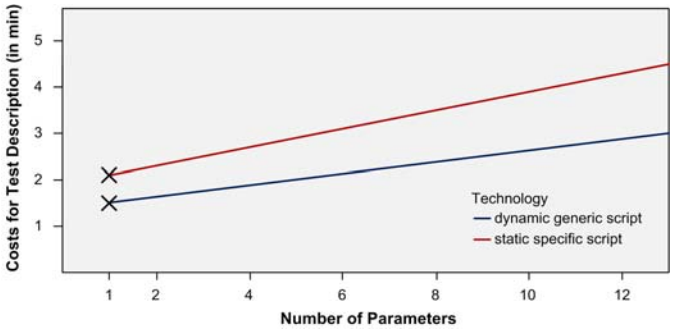

Figure 3. Duration of the test description.

The evaluation compares the traditional static approach and the novel ontology-based approach. Both approaches are test case generator scripts, which aim at providing test cases as input data for a simulation in an automated and structured way. Firstly, the evaluation concept determines the costs for the test description as configuration duration of the parameter setting. Secondly, the effort for implementing test case parameters is determined with respect to the experience level. Lastly, the Return on Investment (ROI) is calculated assuming experience in the use of the different technologies.

Figure 3 shows the costs for the test description for both generator scripts with respect to the implemented number of parameters. Therefore, the time to configure the test case generation process is measured. For the empirical evaluation, we measured the time needed for test description using 1, 4 and 10 parameters for each approach and extrapolated the results. In order to empirically evaluate the effort needed for additional parameters, we again measured the time needed to add 1, 3 or 5 parameters and then extrapolated the results to be able give evidences for larger use case scenarios. Figure 4 outlines the effort for implementing up to 195 test case parameters. As a result, the higher effort for the first time implementation of the dynamic generic script pays off after the implementation of the 38th test case parameter. Nevertheless, the secondary criteria such as a lower risk for mistakes during the configuration phase of the generation process and ensuring valid and consistent test cases as output of the generation process make the use of the dynamic generic script preferable even for a small number of test case parameters.

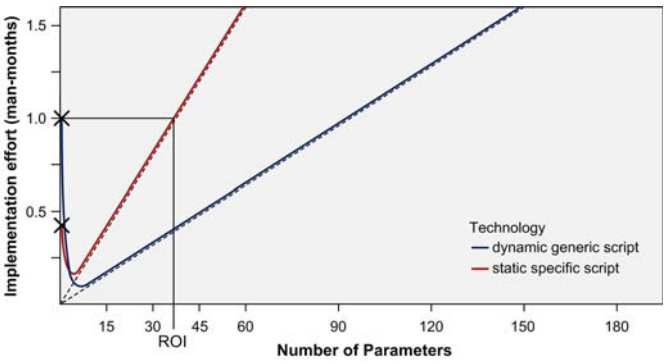

Figure 4. Effort for adding a high number of parameters.

VI. DISCUSSION

The results of the evaluation are listed in Table 1. As can be seen, the ontology-based approach performs better on most objectives. The ontology-based approach demands a higher effort for the first time implementation. In addition, the user needs to obtain the necessary skills for the used generator script. For using the static script the user needs programming skills for both generating test cases and adding new parameters. The ontology-based script assumes skills in using an ontology editor for adding new test case parameters to the ontology. The ontology-based approach requires no specific skills for generating test cases. The percentage values for the implementation effort are based on the effort interpolation presented in the evaluation section. For the constant parameters scenario, the effort needed for implementation of the static approach is take as base effort (100%) since it is lower than the effort needed for implementing the ontology-based approach; and vice versa for the expandable number of parameters scenario.

Table 1. Results of the Evaluation.

	Ontology-based approach		Static approach	
	Constant Parameters	Expandability of parameters	Constant Parameters	Expandability of parameters
Test Description	high-level	high-level	low-level	low-level
Implementation	260%	100%	100%	250%
Test Coverage	is defineable	is defineable	is not defineable	is not defineable

VII. Conclusion and Further Work

High test coverage is essential for testing the performance of simulation systems not only in the production automation domain. Test case generators provide such test cases as input data for the simulation in an automatic and structured way. Many solutions for test case generators use a static approach which is difficult to expand. Mostly, these solutions offer an unsatisfying usability since only a low-level test description is supported. In this work, we discussed the test case generation process and developed an ontology-based approach based on an available static one to achieve high-level test description, lower effort for implementing additional test case parameters, and a definable test coverage.

This paper identified weaknesses of traditional approaches and proposed and evaluated a solution to address these weaknesses. After the implementation of the new approach the effectiveness of both approaches was compared. The ontology-based approach performs very well on most objectives as the result of the evaluation shows". Once the dynamic generic script is running it does not have to be maintained anymore even if the number of test case parameters varies over time.

Future Work. Future research will include the feedback of the simulation results into the ontology. An important fact is that the feedback is not limited to the simulation results since the results depend on the test cases and the assembly line. Another future research topic will be the more excessive usage of ontology-based reasoning to support the deduction of test cases. At the moment the ontology is used for building a dynamic GUI at runtime to ensure that the parameter setting is valid and consistent. In addition, the structure of the XML file corresponds to the structure of the ontology. However, the deduction of test cases is probably more efficient than generating test cases as combinatorial possibilities of the parameter setting especially since the ontology is a knowledge-based system and therefore enables to infer from the stored fact base.

Acknowledgment

This work has been supported by the Christian Doppler Forschungsgesellschaft and the BMWFJ, Austria. This work has been partially funded by the Vienna University of Technology, in the Complex Systems Design and Engineering Lab.

References

[1] B. Bruegge, and A.H. Dutoit, *Object oriented software engineering*, Prentice Hall, 2009.

[2] C. Calero, F. Ruiz, and M. Piattini, *Ontologies for Software Engineering and Software Technology*, Springer-Verlag New York Inc, 2006.

[3] D. Gašević, D. Djurić, V. Devedžić, and B. Selić, *Model driven architecture and ontology development*, Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2006.

[4] T.R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, no. 2, 1993, pp. 199-220.

[5] H. Knublauch, D. Oberle, P. Tetlow, and E. Wallace, "A semantic web primer for object-oriented software developers," *W3C Working Group Note http://www.w3.org/TR/sw-oosdprimer*, 2006.

[6] A. Lüder, J. Peschke, T. Sauter, S. Deter, and D. Diep, "Distributed intelligence for plant automation based on multi-agent systems: the PABADIS approach," *Production Planning and Control*, vol. 15, no. 2, 2004, pp. 201-212.

[7] M. Merdan, T. Moser, D. Wahyudin, and S. Biffl, "Performance Evaluation of Workflow Scheduling Strategies Considering Transportation Times and Conveyor Failures," *International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2008, pp. 389-394.

[8] M. Merdan, T. Moser, D. Wahyudin, and S. Biffl, "Simulation of Workflow Scheduling Strategies Using the MAST Test Management System," *10th International Conference on Control, Automation, Robotics and Vision (ICARCV 2008)*, 2008, pp. 1172-1177.

[9] C.D. Nguyen, A. Perini, and P. Tonella, "Ontology-based test generation for multiagent systems," *7th international joint conference on Autonomous agents and multiagent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 1315-1320.

[10] A. Press, "Software disasters are often people problems," *http://www.msnbc.msn.com/id/6174622*, 2004.

[11] A. Spillner, and T. Linz, *Basiswissen Softwaretest*, dpunkt-Verl., 2004.

[12] E.H. Van Leeuwen, and D. Norrie, "Holons and holarchies," *Manufacturing Engineer*, vol. 76, no. 2, 1997, pp. 86-88.

[13] P. Vrba, "MAST: manufacturing agent simulation tool," *IEEE Conference on Emerging Technologies and Factory Automation (ETFA '03)* 2003, pp. 282-287.