B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 25

Application and Testing of Business Processes in the Energy Domain

Kristof Böhmer[1] Florian Stertz[1] Tobias Hildebrandt[1] Stefanie Rinderle-Ma[1]
Günther Eibl[2] Cornelia Ferner[2] Sebastian Burkhart[2] Dominik Engel[2]

Abstract:

The energy domain currently struggles with radical legal and technological changes, such as, smart meters. This results in new use cases which can be implemented based on business process technology. Understanding and automating business processes requires to model and test them. However, existing process testing approaches frequently struggle with the testing of process resources, such as ERP systems, and negative testing. Hence, this work presents a toolchain which tackles that limitations. The approach uses an open source process engine to generate event logs and applies process mining techniques in a novel way.

Keywords: Business Process, Process Testing, Energy, Process Analysis

1 Introduction

The protection of today's energy production and transmission organizations against fraud, misuse, and faults is crucial in order to ensure the stable, easy, and cheap access to electricity [Co09]. Until now the energy domain achieved the required level of protection based on an *isolation driven strategy*, cf. [We10]. However, driven by legislation changes, increased complexity, and new technologies the need for open and standardized approaches becomes obvious, cf. [AB07]. For example, *smart meters* have emerged in recent years. Smart meters provide fine-grained energy consumption measurement and bidirectional communication with various stakeholders, such as energy providers, cf. [DWD11]. Such novel technologies and developments provide a plethora of advantages – but also pose *novel challenges*, cf. [We10]. For example, smart meters do not only foster the stabilization of large complex energy transmission networks but also enable to remotely cut off end customers from their access to electricity, cf. [DWD11]. Hence, it is important to test smart meters and related technologies.

At the same time, the complexity increased in the energy domain. Hence, standardized use cases were defined and agreed upon by major Austrian energy producers and distributors, cf. [OE15]. These use cases describe the core business processes of energy producers and distributors (denoted as *energy processes* in the following) in textual form. For energy

[1] Universität Wien, Workflow Systems and Technology, Währingerstrasse 29, A-1090 Wien, firstname.lastname@univie.ac.at
[2] FH Salzburg, Josef-Ressel-Zentrum, Campus Urstein Süd 1, A-5412 Puch/Salzburg, firstname.lastname@fh-salzburg.ac.at

providers it is a crucial task to implement the energy processes in order to address upcoming energy related challenges.

The process model for the `prepayment` use case is depicted in Fig. 1. This and the following examples are defined using Business Process Model and Notation (BPMN) as the standard process modeling notation. Fig. 1 contains multiple related critical smart metering use cases (e.g, payment handling) that can, if a fault occurs, lead to an unexpected energy turnoff. For the sake of brevity this work will focus on the depicted use case and its connected process model. However other use cases and requirements were also successfully modeled.

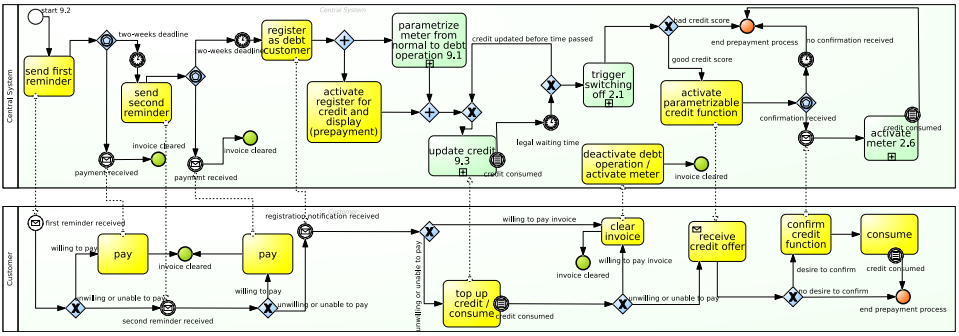

Fig. 1: BPMN model of prepayment process and use cases (modeled using Signavio)

In addition to implementing specifications/use cases it is also necessary to *test* the created processes and their integration of internal and external services. Hereby related faults can be identified early during the process design phase. Business process testing focuses on defining and executing test cases that specify expected behavior. Unfortunately current process testing approaches are somewhat limited. For example, existing approaches struggle with a flexible integration of real world and mocked services into the business processes. Hence existing work abstracts from incorporating external services into the tests, cf. [BR16].

Moreover, existing approaches focus on testing that specified behavior is possible (i.e., positive testing) neglecting negative testing. This is testing that a specific kind of behavior is not possible, e.g., to ensure that an anticipated fault is not present.

Hence, existing work isn't suitable to answer the following research questions:

RQ1 How can business process testing be applied in the energy domain?

RQ2 How to flexibly transition from simulation/test environments into production?

RQ3 How to conduct negative testing in the proposed process testing toolchain?

This work presents a toolchain for modeling and testing real world use cases and processes from the energy domain. Hereby this work especially focuses on automatically testing if all resources (e.g., applications or web-services), which are integrated in the processes, behave "correctly" based on their specification. Moreover it will be discussed how fine granular negative and positive test cases can be defined using conformance rules.

The toolchain is discussed in Section 2. Evaluation results are discussed in Section 3. Section 4 discusses related work. Conclusions and future work is given in Section 5.

2 Toolchain for Flexible Process Testing

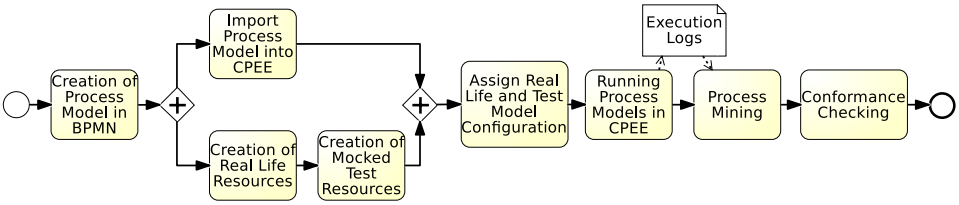

Fig. 2: Toolchain

Existing process testing approaches typically struggle with the integration and testing of resources, cf. [BR15]. Hence, we propose a resource focused testing approach, cf. Fig. 2. The proposed approach assumes that following information is given: a process model that should be tested, a set of resources that are utilized by the model, and test cases. Each test case (short test) holds resource specifications and the data flow variable values required to instantiate/execute the model. Moreover, the presented approach utilizes the Cloud Process Execution Engine (CPEE). It is applied to execute processes under real world conditions but also to simulate executions for test purposes.

The testing starts with importing the model into the CPEE two times with different configurations. The first configuration wires all resources – referenced by the model – to their real life resource implementations (e.g., real web services which would enable to check credit card data). Secondly, a testing configuration is created that utilizes abstracted mocked versions of the real life resources. We refer to the first configuration as the "real life model" and to the second one as the "testing model".

The mocked resources are generated individually, based on resource specifications which are hold by each test. Hence, each test case likely will bring a slightly varying specification. These specifications must fit the use cases and scenarios to be tested. For example, specifications and use cases can be extracted from public documentations, e.g., the use cases published by the Austrian energy domain [OE15]. Note, that the CPEE enables to flexibly configure if a mocked or real life resource should be utilized. Moreover, it enables to flexibly implement the mocked resource behavior programmability with the tool of choice. For example, as a process using a chosen modeling notation or as a service using a programming language such as PHP or Java.

Finally, all the preparatory test artifacts (e.g., mocked resources) are available. Hence, the testing can continue based on a two pronged approach. First, the real life model is executed based on the instantiation/test data defined in the test case. Secondly, the testing model is executed based on the same test data than the real life model and the mocked resources. During the execution of both model configurations the CPEE logging component, cf. [St16], stores all the executed behavior (i.e., execution events) in an execution log. Events and execution logs hold, for example, which steps (e.g., activities) were executed during the

execution of the models. Moreover, they enable to deduce the order of the steps and the data used/exchanged during the execution.

This enables to conduct a wide range of an analysis based on existing conformance checking and process mining techniques, cf. [Aa11]. Conformance checking enables to determine if recorded execution logs conform to expected behavior. For example, conformance checking enables us to determine if each recorded step fits to behavior which is defined in a process model. In comparison, process discovery approaches enable to automatically "identify" the structure/behavior of a process model based on recorded execution logs. In short, recorded execution logs can be converted into a model that is capable of producing the analyzed log – based on existing process mining approaches.

This approach proposes to exploit conformance checking and process mining to ensure the correct implementation of resources which are integrated into process models. Tool support for conformance checking and process mining is provided by, for example, ProM (http://www.promtools.org/). It is proposed to apply process mining on the execution log generated by the model under test. The hereby generated process model is later compared to the execution log generated by the real life model using existing conformance checking techniques, cf. [Aa11]. It is assumed that identified deviations likely indicate either faults at the test/mocked resources or, more likely, at the real life resource implementations.

For example, assume, that a resource has to decide if Eve is allowed to order a product based on her previously determined creditworthiness during a process model execution. Hence, the models' execution differentiates if Eve is creditworthy or not (e.g., an order product step is executed/logged during the models' execution or not). Imagine that the test case defines Eve as *not* creditworthy and also instruments the mocked service accordingly. However, if in the logged real life model events the order product step can be found then the conformance checking will fail and point out a potential fault.

The generated results can be utilized to improve the tested processes in an iterative manner. Hence, the proposed testing approach can be applied multiple times in a row to determine if all identified deviations from the test specification were found/fixed. During each iteration identified deviations can be addressed, either by fixing the real live resource implementation or by adapting a faulty test. Note, we assume that, most likely, multiple test cases will be available. In such a case the previous steps are executed once for each test case.

Finally, we want to point out two additional advantages of the proposed approach. Models can be tested and executed with a mixture of real life and mocked resources. This allows to test the implementation of the resources as soon as they become available. Moreover, this shortens the time to market. This is because the real life process model and execution engine can be used during all tests. Hence, the tested real life model configuration can, after the testing has finished, simply be pushed to the production environment. In addition, negative tests can also be conducted by altering the mocked resources so that they behave "incorrectly". This enables to test/ensure that the specified incorrect behavior is not occurring in the real life process model and its resources.

3 Evaluation

For evaluating the toolchain, we focused on the prepayment process, discussed in Section 1 and 2. We focused on this use case, because the end state of this use case is severe, since a customer would be cut off of his access to power (i.e., electricity), cf. [OE15]. As the first step, we created the BPMN model for this use case. Fig. 1 shows the result of the first step. This process starts with sending a reminder to the customer for paying his bill. A second reminder will be sent, if no payment has been received for two weeks. If the customer does not pay after two weeks the central system of the energy utility will register the customer as a debt customer. The smart meter of the customer will be put in a debt mode and a credit option will be activated for the customer, as a prepayment option. The customer can then top up his credit and consume it afterwards. After his credit is depleted, his power will be cut off. If he is financially powerful, the energy utility, can offer him a credit function for his power supply. If the customer accepts, his power supply will be turned on again and he can consume as much power as his credit covers. Again his power will be cut off, if his credit is depleted. A customer does always have the option of paying his bills to be cleared and registered as a normal customer again.

The next step involves two parts. The first part is building a process model in the CPEE. One approach would be creating an XML file directly for the CPEE, which is tedious and prone to errors. So the CPEE front end (http://cpee.org/~demo/cpee-cockpit) also allows to create process models easily in two ways. It is possible to either create a process model through a graphical editor or to directly importing a model from a BPMN file. The process model can be seen in Fig. 3. Every step of this process sends data to a resource.

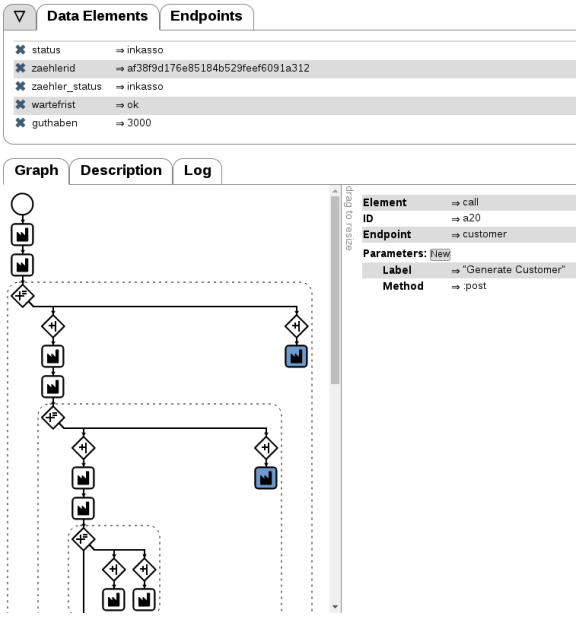

Fig. 3: Prepayment Process created using the CPEE front end

In this scenario, every resource is a web resource and the messages are sent via HTTP. It is important to note, that these resources can be created easily and do not require a big amount of code. For example, PHP based web-services are perfectly suited for this case. After the testing phase, each of these resources can be swapped for a real resource. List. 1 shows a very small entry of an event log. The event “Send first reminder” has been observed on the 5th of October. For this simple query, we wanted to know if a customer with the id of 163 has paid his bill, so we sent his id to the resource. The resource reported back to us and we can see, that the

customer with the id 163 paid his first reminder. The process should end here, since the customer should not be a debt customer now.

List. 1: Example of Log File

```
<event>
  <string key="concept:name" value="Send␣first␣reminder"/>
  <list key="data_send">
    <string key="knr" value="163"/>
  </list>
  <date key="time:timestamp" value="2016-10-05T19:38:26+02:00"/>
  <list key="data_received">
    <string key="sent" value="true"/>
  </list>
</event>
```

How the actions of a resource on the CPEE are logged is explained in more detail in [St16]. The next step describes executing this process model. The CPEE creates a process instance of a process model and executes it. It is possible to create many instances at once, so we can generate multiple event logs which cover the whole behavior of the model. Every process instance creates one event log. These event logs can also be put together to generate one large event log.

The next step of the toolchain is process mining. The basics of process mining are described in Section 2. We created an event log with 1001 processes which consists of 20349 events. With ProM we can mine a process model out of this log and see if it suits our designed process model and our conformance rules through conformance checking. It is important to note, that the implemented conformance checking at the moment only takes the control flow into account for the fault detection. Faults related to data, such as, an incorrect process data variable state, are not detectable at this time. An in depth description of conformance checking can be found in [RA08].

After the conformance checking, the test resources can be altered and new event logs can be created. These steps can be repeated until the conformance checking results fit the real process model and all the test resources were swapped out for the real resources.

4 Related Work

Related work can mainly be found in the *business process testing* domain. There a plethora of approaches, techniques, and concepts are available, cf. [BR15]. Surprisingly it was found that a *flexible integration* of external or internal services and applications (i.e., resources) into test executions is currently hardly provided. Hence, the majority of the existing process testing work abstracts from this challenge, and for example, ignores it, cf. [BR15; ML06].

Alternatively simple approaches are applied that typically simulate resources based on test cases that only hold predefined resource return values, cf. [Br08; BR15; LSD08]. Overall, the most flexible existing integration of resources were found in [LS06] and [LSD08]. In [LS06] the authors propose to transform the process into JAVA code (e.g., activities would become classes), which enables to flexibly mock external resources. For example, basic simulated resource behavior can be implemented in JAVA source code.

This results in a high manual programming effort and still does not enable to easily switch between mocked resources and their real world implementation. So in [LSD08] the authors propose to utilize the Business Process Execution Language (BPEL) to defined the behavior of mocked external resources. Unfortunately their proposed framework only focuses on testing BPEL processes. Overall we came to the conclusion that the integration of resources into test executions is currently limited and tackle this limitation with the proposed testing framework. A similar situation was found when the related work was checked for their *negative testing* capabilities. A majority of the identified existing work focuses on positive testing, cf. [BR15]. Hence, the expected behavior is defined at the test cases, for example, based on the expected variable states or control flow paths. During test execution the observed behavior is compared with the expected one.

However, while existing model checking based testing approaches are also mainly applied for positive testing they can also be utilized for negative testing, cf. [BR15; FBS04; Na06]. Unfortunately, such model testing based approaches frequently require an in depth knowledge of formal rule modeling techniques or custom rule definition languages. We assume that such knowledge can be lacking at typical IT professionals, cf. [BR15]. Overall, we concluded that existing process testing work does not put a strong focus on negative testing – a limitation addressed by this work. In addition we were not able to identify any existing work that focuses or reports on process testing in the energy domain.

5 Conclusion

This paper proposes a process testing toolchain that takes a process model and resources as input and generates test log data as output. In addition existing process conformance and mining approaches are applied in a novel way to test the processes. Hereby, existing resource specifications are exploited to ensure that resources which are integrated into business processes are correctly implemented. In addition the presented toolchain enables, based on the process execution engine CPEE, a flexible integration and switches between real life implementations of resources and customized mocked resources (→ **RQ2**).

It was concluded that this not only enables to start early with resource/process testing (i.e., when the first real life resources become available) but also improves the time to marked for the process models under test. The presented approach also discusses the provision of negative testing capabilities (→ **RQ3**). In order to demonstrate the applicability and feasibility of the toolchain, it has been applied to a real-world uses case/specification from the energy domain [OE15], i.e., the *prepayment* process (→ **RQ1**).

The toolchain can be also applied independently from the energy domain. In future work, the negative testing capabilities could be used in the security domain to ensure that predicted attack vectors cannot be exploited.

Acknowledgement This research was partly funded by the Austrian Research Promotion Agency, project 849914.

References

[Aa11] van der Aalst, W. M.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, 2011.

[AB07] Armaroli, N.; Balzani, V.: The future of energy supply: challenges and opportunities. Angewandte Chemie International Edition 46/1-2, pp. 52–66, 2007.

[Br08] Breu, R.; Lechner, A.; Willburger, M.; Katt, B.: Workflow Testing. In: Leveraging Applications of Formal Methods, Verification and Validation. Springer, pp. 709–723, 2008.

[BR15] Böhmer, K.; Rinderle-Ma, S.: A systematic literature review on process model testing: Approaches, challenges, and research directions, Technical Report, 2015.

[BR16] Böhmer, K.; Rinderle-Ma, S.: A Testing Approach for Hidden Concurrencies based on Process Execution Logs. In: Service Oriented Computing. Oct. 2016.

[Co09] Cottrell, F.: Energy & society: the relation between energy, social change, and economic development. AuthorHouse, 2009.

[DWD11] Depuru, S. S. S. R.; Wang, L.; Devabhaktuni, V.: Smart meters for power grid: Challenges, issues, advantages and status. Renewable and sustainable energy reviews 15/6, pp. 2736–2742, 2011.

[FBS04] Fu, X.; Bultan, T.; Su, J.: Analysis of interacting BPEL web services. In: World Wide Web. ACM, pp. 621–630, 2004.

[LS06] Li, Z. J.; Sun, W.: Bpel-unit: Junit for bpel processes. In: Service Oriented Computing. Springer, pp. 415–426, 2006.

[LSD08] Li, Z. J.; Sun, W.; Du, B.: BPEL4WS unit testing: framework and implementation. Business Process Integration and Management 3/2, pp. 131–143, 2008.

[ML06] Mayer, P.; Lübke, D.: Towards a BPEL unit testing framework. In: Testing, analysis, and verification of web services and applications. ACM, pp. 33–42, 2006.

[Na06] Nakajima, S.: Model-checking behavioral specification of BPEL applications. Theoretical Computer Science 151/2, pp. 89–105, 2006.

[OE15] OE: Smart Metering Use-Cases für das Advanced Meter Communication System (AMCS), Version 1.0, tech. rep. 1/88, Oesterreichs Energie, 2015.

[RA08] Rozinat, A.; van der Aalst, W. M.: Conformance checking of processes based on monitoring real behavior. Information Systems 33/1, pp. 64–95, 2008.

[St16] Stertz, F.; Rinderle-Ma, S.; Hildebrandt, T.; Mangler, J.: Testing Processes with Service Invocation: Advanced Logging in CPEE. In: Service Oriented Computing. Oct. 2016.

[We10] Wei, D.; Lu, Y.; Jafari, M.; Skare, P.; Rohde, K.: An integrated security system of protecting smart grid against cyber attacks. In: Innovative Smart Grid Technologies. IEEE, pp. 1–7, 2010.