A BPMN Profile for Test Case Execution Visualization

Daniel Lübke[1,2,*]

[1]Digital Solution Architecture GmbH, Hannover, Germany
[2]Leibniz Universität Hannover, FG Software Engineering, Hannover, Germany

Abstract

Testing executable business processes, e.g., developed in BPMN 2.0 or WS-BPEL, is an important task within development projects. However, this task is labor-intensive due to the amount and scale of interactions of a test framework (e.g., BPELUnit) with the process under test. We want to help testers to debug test case failures by visualizing the test run information as a BPMN diagram. We developed a visual, BPMN-based notation for visualizing test runs, implemented a generation from BPELUnit test suites and execution logs to the newly defined format, and applied this in an industrial project. While no thorough validation has yet been performed, early results indicate better understandability of our notation compared to raw test logs.

Keywords

BPMN, Test Case, Visualization, Profile, BPELUnit

1. Motivation

Testing executable business processes is an important task in many digitization projects: errors in critical business processes are a huge risk for reputation and income of organizations. Therefore, unit testing, e.g., with BPELUnit [1], is often one activity to improve functional quality.

However, analyzing failing unit tests in executable business processes is often difficult, because they integrate many partners that potentially are performing activities in parallel. As such, test case logs containing successfully executed and failed activities alongside passed input and output messages become large and are hard to analyze. While test frameworks indicate which activity failed, further analysis is often required to pinpoint the underlying problem. This is especially true, when test cases are generated, e.g, by using combinatorial test design [2, 3] because errors in the test case configuration are more likely than with manually written tests.

We developed an approach to generate BPMN models, which help developers and testers to easier analyze failed test cases. It builds upon a test case subset for BPMN [4] and uses colors and documentation to make test case execution information better accessible.

This paper starts with a presentation of related work (section 2). Afterwards, the generation approach and underlying meta models are presented in section 3, which are implemented in a prototype application (section 4). At the end, a small validation in an industry project is

ZEUS'2024: 16th Central European Workshop on Services and their Composition, February 29 – March 01, 2024, Ulm, Germany

*Corresponding author.
daniel.luebke@digital-solution-architecture.com (D. Lübke)
https://www.digital-solution-architecture.com (D. Lübke)
0000-0002-1557-8804 (D. Lübke)

S. Böhm and D. Lübke (Eds.): 16th ZEUS Workshop, ZEUS 2024, Ulm, Germany, 29 February–1 March 2024, published at http://ceur-ws.org

presented in section 5. Finally, conclusions are drawn and outlook on future work is made (section 6).

2. Related Work

Visualizing test cases has been a research area of interest for quite some time. For example, Cornelissen et al. [5] visualized execution traces of unit tests with UML Sequence Diagrams to help developers better understand the inner workings of software systems.

But visualization is not only concerned with the test execution per se but also with its results. For example, Opmanis et al. [6] plotted test result data over time, and Dzidic [7] performed a case study visualizing metrics of software tests as dashboards in the financial sector.

However, more similar to the goals of this research, studies have been concerned with visualizations to better detect faults: Jones et al. [8, 9] used coverage information for visualizing which parts of the software have been covered by which test cases. Similar work has been done by Koochakzadeh & Garousi [10]. They visualized dependencies in the program under test on class dependencies and implemented this as an Eclipse plug-in. Wes et al. [11] plotted test executions as Multivariate Visualizations to cluster test cases and ease reasoning about successfully implemented scenarios.

Existing tooling can also be used to present more data to testers. For example, there is a tool for the Camunda BPMN engine [1] to visualize test coverage metrics as a graphical overlay over the process.

3. Definition, Meta Model and Generation

A BPMN Test Execution Visualization is a BPMN model that shows all activities executed or supposed to be executed during a test run, which conforms to the following constraints (derived from the constraints for model-driven testing of executable business processes [12]):

1. It shows test case execution in a BPMN model,
2. it uses uncollapsed pools for each logical party in the test case plus one collapsed pool for the process under test,
3. it only uses the following BPMN elements: tasks (user, service), events (catch/throw message, timer), and gateways (parallel),
4. its message flows must originate or terminate at the collapsed pool for the process under test,
5. it shows the status of each activity by color: red (error), yellow (interrupted), and green (okay),
6. it contains additional diagnostic information in the BPMN elements' documentations, e.g., received and sent message payload.

Figure 1 shows the relevant excerpts of the BPELUnit, BPELUnit log, and BPMN meta models required for generation of the BPMN models: A test case execution is mapped to one BPMN

[1] https://github.com/camunda-community-hub/camunda-process-test-coverage

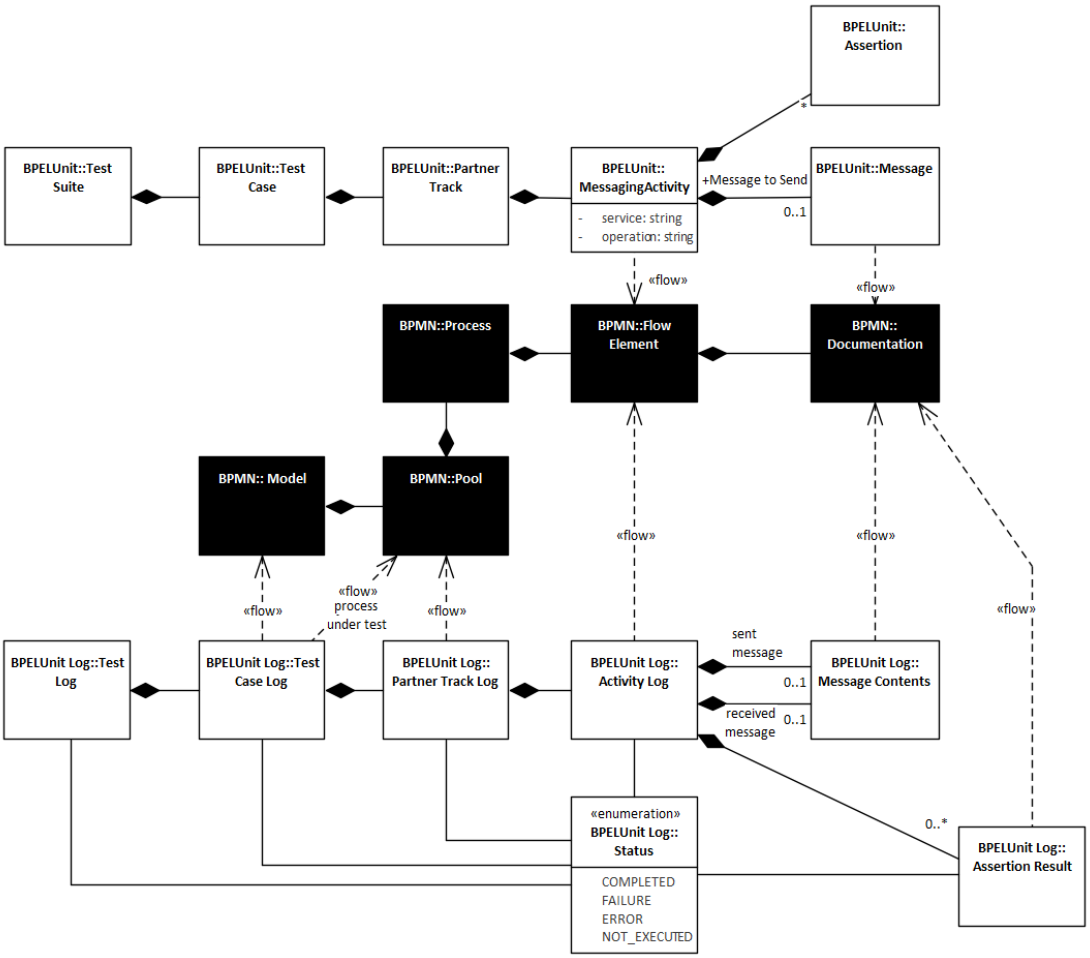

Figure 1: Meta Model and Generational Data Flows for Test Suite Information

collaboration diagram. Every mocked service, i.e., partner is represented by a pool. The process under test itself is represented by a collapsed pool. This means that this visualization technique will treat the process under test as a black box in contrast to other visualizations, e.g., those for test coverage.

If a mock sends a one-way message, a message throw event is added to its pool, if it receives a one-way message, a message catch event is added. If it initiates a two-way message exchange, a service task is added, and if it waits for a two-way message exchange a message catch event is added followed by a message throw event. If the mock waits for a specified duration, a timer event is added, and parallel branches within a partner is represented with parallel gateways. Because BPELUnit is block-structured, it is easy to also add the merging parallel gateway.

4. Prototype Implementation

Within this section we shortly describe some design considerations for the prototype application, which contains as much functionality as required for trying it in the industry project (see section 5). The prototype is available at https://github.com/dluebke/bpelunit-viz.

4.1. Targeted Tool Chain

Because we want to use colors for denoting tasks' error status, we were required to settle on one tool, because colors are not standardized in the diagram interchange format of the BPMN 2.0 specification. We settled on the Camunda Modeler because it allows us to show the documentation in a comparatively large text area in a side pane. Thus, developers can easily select BPMN elements of interest and directly see exchanged information.

4.2. XML Processing

We used Java's DOM and XPath API to read elements and attribute values from the BPELUnit log. We have planned to add the processing of the BPELUnit test suite itself as described above. However, the prototype does not yet resolve the original operation name as foreseen in the concept.

Internally, all metamodels are represented by model classes (BPELUnit, BPELUnit Log, BPMN 2.0). BPMN 2.0 can be serialized via custom serializers for the model classes.

4.3. Layouting

The resulting layout of a test case visualizaton is illustrated in Figure 2.

Because developers want to use the generated BPMN diagrams without further effort, we were required to layout the diagrams as well. We simply layouted all flow nodes in a pool horizontally. This approach works reasonably well because the diagrams contain no loops, few gateways, and are mostly sequential.

To reduce the average length of message-flows and number of line-crossings and thereby making the diagram easier to comprehend, we ordered the pools based on the number of contained flow nodes (events, tasks and gateways): The pool representing the process under test is placed in the middle. The pool containing the most flow nodes is placed on top of it, the pool with second most flow nodes is placed below it. The ordering progresses by adding the pools descendingly sorted by the number of contained flow nodes on top and below already layouted pools.

Some tuning was required due to the the large amount of message-flows and pools: Because nearly all task had straight vertical message flows to the process-under-test pool, most message flows overlapped. We resolved this problem by slightly offsetting the horizontal starting point within each pool.

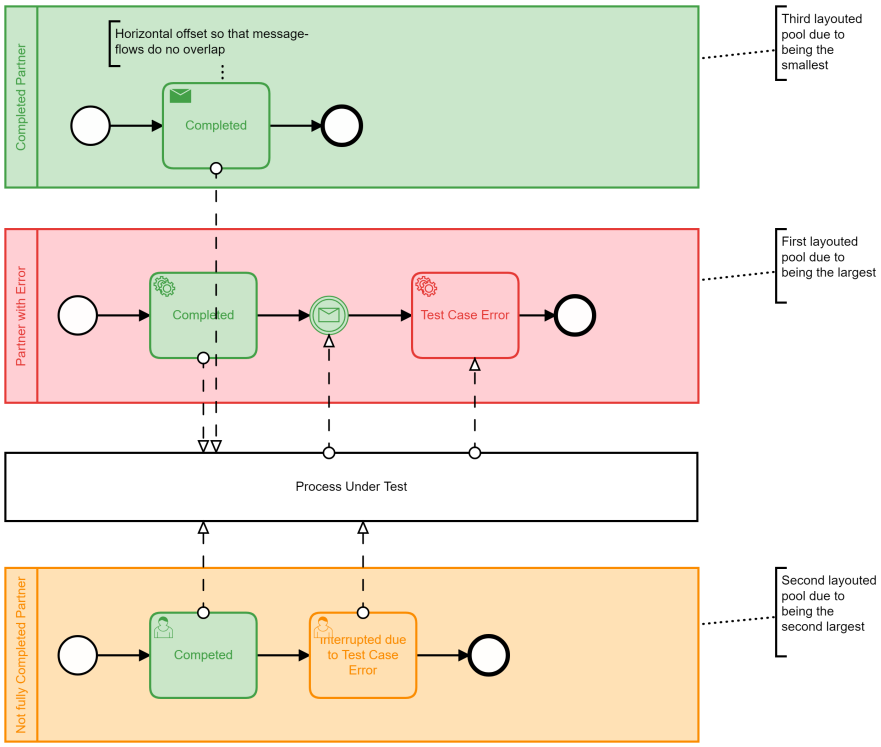

Figure 2: Example and Explanation of the Layout

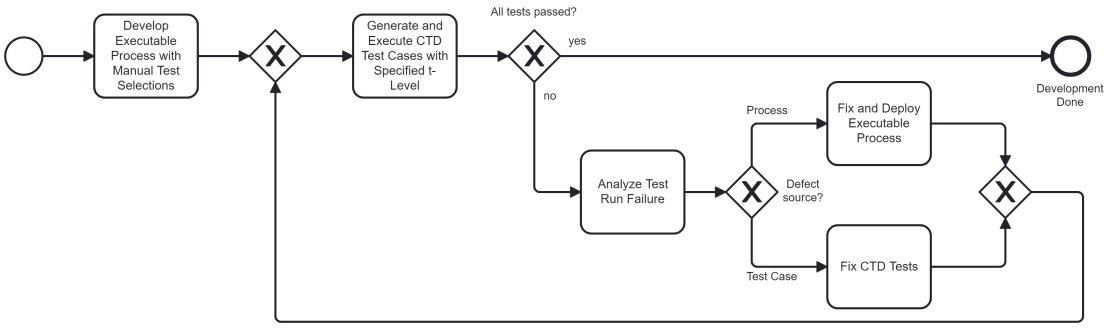

Figure 3: Development Process in the Industrial Project using CTD Test Generation

5. Evaluation: Application in Industry Project

For addressing quality concerns, Terravis [13] — a Swiss large-scale process integration platform — has invested much in efforts for improving testing. Due to many changes to its BPEL processes [14], extensive unit and integration tests based on BPELUnit [1, 15] have been developed. Especially, with the generation of such tests with Combinatorial Test Design (CTD) [2, 3] many

tests need to be debugged until the development of the generated test suites is completed. Similarily, after extending processes newly introduced bugs needed to be identified and tests adjusted.

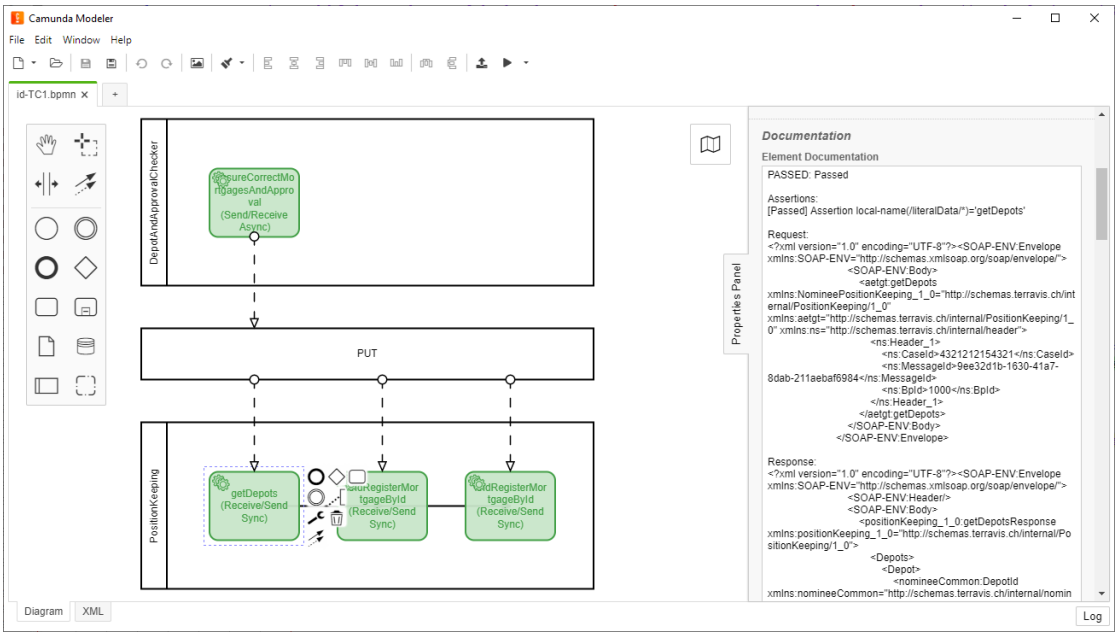

Figure 4: Generated BPMN test case visualization shown in Camunda Modeler with XML payload and assertion information in the right-hand property pane.

The project has so far used the visualization for 25 of its processes that are covered by CTD tests executed with BPELUnit. The largest BPELUnit log file is 48 MB large containing 328 test cases. The log file size is mainly influenced by the (large) XML messages being exchanged in the covered process. The visualization thus reduces the cognitive load required by developers because it filters out payload information initially although allowing users to access it via the elements' documentation as shown in Figure 4.

Feedback by developers was positive and they indicated that the visualization helped them to diagnose problems. However, further validation is required.

6. Conclusion and Outlook

In this paper we presented a way to visualize test cases for executable business processes and their executions as BPMN models. The proposed algorithm has been implemented in a tool and successfully applied in an industrial case. Consequently, BPMN can be used as a single language to model the business processes, model or visualize test cases (depending on the testing approaches), and visualize test execution results.

In future work we like to further validate the approach by conducting surveys and experiments for comparing log-based analysis with analysis performed using our models with regard to

effectiveness of finding and fixing problems in test cases and executable business processes.

We also made the prototype implementation publicly available so that everyone can use it and perform such studies with us or independently. We are happy to collaborate with others concerning research into optimizing the testing of executable business processes!

Acknowledgments

We like to thank all team members of the industrial case project Terravis for their participation.

References

[1] P. Mayer, D. Lübke, Towards a BPEL Unit Testing Framework, in: TAV-WEB '06: Proceedings of the 2006 Workshop on Testing, Analysis, and Verification of Web Services and Applications, Portland, USA, ACM Press, New York, NY, USA, 2006, pp. 33–42. URL: http://portal.acm.org/affiliated/citation.cfm?id=1145718.1145723&coll=ACM&dl=ACM&type=series&idx=1145718&part=Proceedings&WantType=Proceedings&title=International%20Symposium%20on%20Software%20Testing%20and%20Analysis&CFID=1483183&CFTOKEN=32880799#. doi:http://doi.acm.org/10.1145/1145718.1145723.

[2] T. Schnelle, D. Lübke, Towards the generation of test cases for executable business processes from classification trees, in: Proceedings of the 9th Central European Workshop on Services and their Composition (ZEUS) 2017, 2017, pp. 15–22.

[3] D. Lübke, J. Greenyer, D. Vatlin, Effectiveness of Combinatorial Test Design with Executable Business Processes, in: D. Lübke, C. Pautasso (Eds.), Empirical Studies on the Development of Executable Business Processes, Springer, 2019, pp. 187–207.

[4] D. Lübke, T. van Lessen, Modeling Test Cases in BPMN for Behavior-Driven Development (Extended Abstract), in: Proceedings of the EMISA Workshop 2016, 2016.

[5] B. Cornelissen, A. van Deursen, L. Moonen, A. Zaidman, Visualizing testsuites to aid in software understanding, in: 11th European Conference on Software Maintenance and Reengineering (CSMR'07), 2007, pp. 213–222. doi:10.1109/CSMR.2007.54.

[6] R. Opmanis, P. Kikusts, M. Opmanis, Visualization of large-scale application testing results, Baltic Journal of Modern Computing 4 (2016) 34.

[7] E. Dzidic, Data Visualization of Software Test Results : A Financial Technology Case Study, Master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2023.

[8] J. A. Jones, M. J. Harrold, J. Stasko, Visualization of test information to assist fault localization, in: Proceedings of the 24th International Conference on Software Engineering, ICSE '02, Association for Computing Machinery, New York, NY, USA, 2002, p. 467–477. URL: https://doi.org/10.1145/581339.581397. doi:10.1145/581339.581397.

[9] J. Jones, Fault localization using visualization of test information, in: Proceedings. 26th International Conference on Software Engineering, 2004, pp. 54–56. doi:10.1109/ICSE.2004.1317420.

[10] N. Koochakzadeh, V. Garousi, Tecrevis: A tool for test coverage and test redundancy

visualization, in: L. Bottaci, G. Fraser (Eds.), Testing – Practice and Research Techniques, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 129–136.

[11] W. Masri, R. A. Assi, F. Zaraket, N. Fatairi, Enhancing fault localization via multivariate visualization, in: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 2012, pp. 737–741. doi:10.1109/ICST.2012.166.

[12] D. Lübke, T. van Lessen, BPMN-based Model-Driven Testing of Service-based Processes, in: Business Process Modeling, Development, and Support 2017, 2017.

[13] W. Berli, D. Lübke, W. Möckli, Terravis – Large Scale Business Process Integration between Public and Private Partners, in: E. Plödereder, L. Grunske, E. Schneider, D. Ull (Eds.), Lecture Notes in Informatics (LNI), Proceedings INFORMATIK 2014, volume P-232, Gesellschaft für Informatik e.V., Gesellschaft für Informatik e.V., 2014, pp. 1075–1090.

[14] D. Lübke, Using Metric Time Lines for Identifying Architecture Shortcomings in Process Execution Architectures, in: Software Architecture and Metrics (SAM), 2015 IEEE/ACM 2nd International Workshop on, IEEE, 2015, pp. 55–58.

[15] D. Lübke, Test and Analysis of Service-Oriented Systems, Springer, 2007, pp. 149–171.