Workflow Management Systems Benchmarking: Unfulfilled Expectations and Lessons Learned

Vincenzo Ferme[†], Jörg Lenhard[*], Simon Harrer[‡], Matthias Geiger[‡], and Cesare Pautasso[†]
[†]Faculty of Informatics, USI Lugano, Switzerland, {firstname.lastname}@usi.ch
[*]Department of Mathematics and Computer Science, Karlstad University, Sweden, joerg.lenhard@kau.se
[‡]Distributed Systems Group, University of Bamberg, Germany, {firstname.lastname}@uni-bamberg.de

***Abstract*—Workflow Management Systems (WfMSs) are a type of middleware that enables the execution of automated business processes. Users rely on WfMSs to construct flexible and easily maintainable software systems. Significant effort has been invested into standardising languages for business processes execution, with standards such as the Web Services Business Process Execution Language 2.0 or the Business Process Model and Notation 2.0. Standardisation aims at avoiding vendor lock-in and enabling WfMS users to compare different systems. The reality is that, despite standardisation efforts, different independent research initiatives show that objectively comparing WfMSs is still challenging. As a result, WfMS users are likely to discover unfulfilled expectations while evaluating and using these systems. In this work, we discuss the findings of two research initiatives dealing with WfMSs benchmarking, presenting unfulfilled expectations and lessons learned concerning WfMSs' usability, reliability, and portability. Our goal is to provide advice for practitioners implementing or planning to use WfMSs.**

***Index Terms*—Workflow Management Systems, Standards, Lessons Learned, Evaluation Research**

I. Introduction

Workflow Management Systems (WfMSs) automate the execution of business processes [1]. They are a widely used middleware technology with many open-source and commercial products available[1]. The complexity of WfMSs makes it difficult for users to compare and select the right WfMS for their use case. The most critical part is probably the language in which a user can model workflows. To this end, several organisations proposed standards, as for example the Organization for the Advancement of Structured Information Standards (OASIS) with the Web Services Business Process Execution Language (WS-BPEL) [2], or the Object Management Group (OMG) with the Business Process Model and Notation (BPMN) [3]. These standards define the language that can be used to implement workflows and the execution lifecycle of the workflow instances. They are meant to clarify the exact scope, building blocks, constraints and semantics of the language in a precise and unambiguous fashion. As a result, the users may freely select a WfMS with respect to the standard it supports.

Unfortunately, in many real-world WfMS, this assumption is flawed in many ways, e.g., inconsistencies [4], [5] and limited support of the standards [6], [7].

This work aims to provide guidance for practitioners (e.g., users and vendors of WfMSs) by highlighting key issues to consider during WfMSs evaluation and by presenting WfMSs that provide suitable approaches. The lessons learned summarised in this extended abstract are derived from the evidence reported in a number of studies that use specific WfMSs, e.g. [4], [6], [8]–[16], which are the result of two independent research initiatives (Betsy [17] and BenchFlow [18]). The BPMN 2.0 WfMSs we consider here are *Activiti*, *Camunda*, and *jBPM*, which according to the vendor websites, are widely used in the industry. Further products claiming to support BPMN 2.0 were also evaluated but could not be integrated in our evaluation approaches due to various reasons such as licencing issues or missing standard compliance [8]. For WS-BPEL, our lessons learned are based on the usage of *Apache ODE*, *OpenESB*, *bpel-g*, *Orchestra*, *Petals ESB*, and three commercial WfMSs whose names we cannot disclose. The data collected during our research are publicly available on an interactive dashboard[2].

II. Unfulfilled Expectations and Lessons Learned

During the analysis we recognised a number of items related to WfMSs usability, reliability, and portability [19] for which we found pitfalls. We also identified WfMSs that avoid the pitfalls by following what we consider a good approach (Table I). In the following, we discuss five major items that we identified.

A. Usability Findings

1) Correctness Checking during Deployment:
WS-BPEL [2] and BPMN 2.0 [3] define rules and constraints for determining whether modelled workflows are correct. WS-BPEL explicitly lists 94 rules named *static analysis rules*, describing issues that any standard compliant WfMS should detect at workflow deployment time. As reported in Table I (A1), many WfMSs do not reject invalid workflows on deployment [12]. Thus, undetected workflow errors may lead to failures of the WfMSs at runtime, which is costly. In some cases of erratic workflows, the execution did not crash observably, but instead completed with non-deterministic results.

[1]See https://en.wikipedia.org/wiki/List_of_BPMN_2.0_engines, last visited on March 6, 2017

[2]See http://peace-project.github.io, last visited on March 6, 2017

TABLE I
SUMMARY OF THE FINDINGS: + (GOOD APPROACH), − (PITFALL PRESENT), ∼ (PITFALL PARTIALLY PRESENT), N/A (NO OBSERVATIONS)

WfMS	A1	A2	A3	B	C
BPMN					
Activiti	∼ [10] ∼ [8]	∼ [10]	+ [8]	N/A	∼ [8] ∼ [16]
Camunda	∼ [10] ∼ [8]	+ [10]	+ [8]	N/A	∼ [8] ∼ [16]
jBPM	∼ [10] ∼ [8]	∼ [10]	∼ [8]	N/A	∼ [8] ∼ [16]
WS-BPEL					
Apache ODE	∼ [12]	N/A	+ [20]	− [11]	∼ [11]
OpenESB	− [12]	N/A	− [20]	+ [11]	∼ [11]
bpel-g, Orchestra, Petals ESB	∼ [12]	N/A	+ [20]	+ [11]	∼ [11]
3 Commercial	N/A	N/A	N/A	N/A	∼ [6]

2) Sensible Default Configuration:

Users expect a WfMS to be usable after a successful installation, with the provided default configuration. Thus the default configuration should enable a stable usage of the WfMS and enable the execution of workflow models utilising any of the features defined in the standard-based workflow language which are supported by the system [19]. This is not always the case, especially with BPMN WfMSs, as shown in Table I (A2). Some WfMSs neither provide a sensible default configuration nor clear guidelines to configure the system in the right way, resulting in the usage of systems with an unstable configuration. This might have a negative impact on the reputation of a vendor, caused by misleading evaluations by the users regarding the actual performance of a WfMS [21]. Furthermore, if the configuration limits the execution of specific language features, the users will consider the system unsuitable for specific use cases they might have.

3) Availability of Management APIs:

WfMSs are usually deployed in production as part of a more complex software ecosystem and typically interact with other diverse systems. When deploying the WfMSs to production, they are also usually integrated into a continuous integration and delivery lifecycle [22]. To successfully achieve the integration, users expect WfMS vendors to provide management APIs for phases like workflow deployment, enactment and analysis. Such management APIs are not present in the WS-BPEL WfMS OpenESB, which rely more on file handling or web interfaces and are provided, or partially provided, in a small subset of BPMN WfMSs, reported in Table I (A3). The rest of the evaluated systems, not reported in Table I, do not provide them at all [20], [23]. Given the limitations or lack of WfMS APIs, it is often hard to integrate the products in continuous integration. E.g., it is not possible to automate testing of the deployed workflows to quickly detect errors.

B. Reliability Findings - Isolation of Instance Execution

WfMSs usually execute many different workflow instances concurrently and users expect that the instances do not influence each other during the execution. Moreover, the stability and integrity of a WfMS should not be impacted in any way by the execution of workflow instances. The WfMSs should restrict hostile instances from breaking out of their runtime environment or crashing the WfMS [15], [24]. The WfMS should also minimise performance influences of and among different workflow instances [15], [24]. Missing isolation during workflow instance execution is similar to a single process crashing a complete operating system. This should not happen and the WfMS should protect itself and the other running instances, for example by detecting excessive resource usage (e.g., RAM, CPU, I/O, Network) and suspending critical workflow instances. Table I (B) reports the findings about WS-BPEL WfMSs, showing that Apache ODE did not isolate hostile instances in certain cases [11].

C. Portability Findings - Standard-based Portability

One of the goals of standardising workflow languages is to establish a commonly agreed set of functionality and a serialisation format for specifying the workflows [2], [3]. Users of WfMSs supporting standard workflow languages expect to be able to move workflows between any of the WfMSs supporting the standard keeping the same execution semantics, protecting themselves from vendor lock-in. Table I (C) shows that all of the evaluated WfMSs are restricted in terms of standard-based portability, mainly because of their limited and non-compliant support of different language features [6], [8], [11], [16]. This limits the advantage of relying on a standard and results in WfMS-dependent workflows, introducing vendor lock-in.

III. CONCLUSION AND FUTURE WORK

In this work, we presented five lessons learned regarding the usage of WfMSs with the aim of helping practitioners who need to select a WfMS. The lessons learned are also relevant for WfMS vendors, because they provide insights on how their systems are perceived and can be improved. The limitations we highlighted are related to usability: the lack of correctness checking of workflow at deployment time, the lack of a sensible default configuration and missing management APIs. We then reported two additional limitations: the non-isolated execution of workflow instances impacting the reliability of WfMSs and the lack of support in standard-based portability of workflows. We plan to extend the presented set to cover more quality aspects and to provide a more comprehensive set of lessons learned to help future practitioners.

ACKNOWLEDGMENT

This work is partially funded by the Swiss National Science Foundation with the BenchFlow - A Benchmark for Workflow Management Systems (Grant Nr. 145062).

REFERENCES

[1] F. Leymann and D. Roller, *Production Workflow: Concepts and Techniques*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[2] OASIS, *Web Services Business Process Execution Language*, 2007, v2.0.

[3] ISO/IEC, *ISO/IEC 19510:2013 – Information technology - Object Management Group Business Process Model and Notation*, 2013, v2.0.2.

[4] M. Geiger and G. Wirtz, "BPMN 2.0 Serialization - Standard Compliance Issues and Evaluation of Modeling Tools," in *5th International Workshop on Enterprise Modelling and Information Systems Architectures*, Sep. 2013.

[5] E. Börger, "Approaches to Modeling Business Processes. A Critical Analysis of BPMN, Workflow Patterns and YAWL," *Softw Syst Model*, vol. 11, no. 3, pp. 305–318, 2012.

[6] S. Harrer, J. Lenhard, and G. Wirtz, "Open Source versus Proprietary Software in Service-Orientation: The Case of BPEL Engines," in *11th International Conference on Service-Oriented Computing (ICSOC)*, Dec. 2013, pp. 99–113.

[7] M. Geiger, S. Harrer, J. Lenhard, M. Casar, A. Vorndran, and G. Wirtz, "BPMN Conformance in Open Source Engines," in *9th IEEE International Symposium of Service-Oriented System Engineering*, Mar. 2015.

[8] M. Geiger, S. Harrer, J. Lenhard, and G. Wirtz, "BPMN 2.0: The state of support and implementation," *Future Generation Computer Systems*, Jan. 2017.

[9] M. Skouradaki, D. H. Roller, F. Leymann, V. Ferme, and C. Pautasso, "On the Road to Benchmarking BPMN 2.0 Workflow Engines," in *6th ACM/SPEC International Conference on Performance Engineering*. ACM, 2015, pp. 301–304.

[10] M. Skouradaki, V. Ferme, C. Pautasso, F. Leymann, and A. van Hoorn, "Micro-Benchmarking BPMN 2.0 Workflow Management Systems with Workflow Patterns," in *28th International Conference on Advanced Information Systems Engineering*, Jun. 2016, pp. 67–82.

[11] S. Harrer, J. Lenhard, and G. Wirtz, "BPEL Conformance in Open Source Engines," in *5th IEEE International Conference on Service-Oriented Computing and Applications*, Dec. 2012, pp. 237–244.

[12] S. Harrer, C. Preißinger, and G. Wirtz, "BPEL Conformance in Open Source Engines: The Case of Static Analysis," in *7th IEEE International Conference on Service-Oriented Computing and Applications*, Nov. 2014, pp. 33–40.

[13] V. Ferme, A. Ivanchikj, C. Pautasso, M. Skouradaki, and F. Leymann, "A Container-centric Methodology for Benchmarking Workflow Management Systems," in *6th International Conference on Cloud Computing and Services Science*, Rome, Italy, 2016.

[14] V. Ferme, M. Skouradaki, A. Ivanchikj, C. Pautasso, and F. Leymann, "BenchFlow: Performance Benchmarking of BPMN 2.0 Open Source Workflow Management Systems," in *Submitted to 29th International Conference on Advanced Information Systems Engineering*, 2017.

[15] S. Harrer, F. Nizamic, G. Wirtz, and A. Lazovik, "Towards a Robustness Evaluation Framework for BPEL Engines," in *Proceedings of the 7th IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, Nov. 2014, pp. 199–206.

[16] J. Lenhard and G. Wirtz, "Portability of Executable Service-Oriented Processes: Metrics and Validation," *Service Oriented Computing and Applications*, vol. 10, no. 4, Dec. 2016.

[17] M. Geiger, S. Harrer, and J. Lenhard, "Process Engine Benchmarking with Betsy in the Context of ISO/IEC Quality Standards," *Softwaretechnik-Trends (STT)*, vol. 36, no. 2, pp. 57–60, 2016.

[18] V. Ferme, A. Ivanchikj, and C. Pautasso, "A framework for benchmarking BPMN 2.0 workflow management systems," in *13th International Conference on Business Process Management (BPM)*. Springer, 2015.

[19] ISO/IEC, *ISO/IEC 25010:2011; Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, 2011.

[20] S. Harrer and J. Lenhard, "Betsy–A BPEL Engine Test System," Otto-Friedrich Universität Bamberg, Tech. Rep. 90, Jul. 2012.

[21] V. Ferme, A. Ivanchikj, and C. Pautasso, "Estimating the Cost for Executing Business Processes in the Cloud," in *International Conference on Business Process Management (BPM)*, 2016, pp. 72–88.

[22] C. Thiemich and F. Puhlmann, "An agile BPM project methodology," in *11th International Conference on Business Process Management*, 2013, pp. 291–306.

[23] J. Lenhard, S. Harrer, and G. Wirtz, "Measuring the Installability of Service Orchestrations Using the SQuaRE Method," in *Proceedings of the 6th IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Kauai, Hawaii, USA: IEEE, Dec. 2013.

[24] F. Leymann, "BPEL vs. BPMN 2.0: Should You Care?" in *2nd Intl. Workshop on BPMN*, 2010, pp. 8–13.