BAMBERGER BEITRÄGE

ZUR WIRTSCHAFTSINFORMATIK UND ANGEWANDTEN INFORMATIK

ISSN 0937-3349

Nr. 93

Literature Survey of Performance Benchmarking Approaches of BPEL Engines

Cedric Röck, Simon Harrer

May 2014

FAKULTÄT WIRTSCHAFTSINFORMATIK UND ANGEWANDTE INFORMATIK

OTTO-FRIEDRICH-UNIVERSITÄT BAMBERG

Distributed Systems Group

Otto-Friedrich Universität Bamberg

An der Weberei 5, 96052 Bamberg, GERMANY

Prof. Dr. rer. nat. Guido Wirtz

Due to hardware developments, strong application needs and the overwhelming influence of the internet, distributed systems have become one of the most important topics for nowadays software industry. Owing to their ever increasing importance for everyday business, distributed systems have high requirements with respect to dependability, robustness and performance. Unfortunately, distribution adds its share to the problems of developing complex systems. Heterogeneity in both, hardware and software, frequent changes, concurrency, distribution of components and the need for inter-operability between systems complicate matters. Moreover, new technical aspects like resource management, load balancing and guaranteeing consistent operation in the presence of partial failures put an additional burden onto the developer. *Our long-term research goal is the development, implementation and evaluation of methods helpful for the realization of robust and easy-to-use software for complex systems in general while putting a focus on the problems and issues regarding distributed systems on all levels.* This includes design methods, visual languages and tools for distributed systems development as well as middleware, SOA and cloud computing issues. Our current research activities focus on different aspects centered around that theme:

- *Implementation of Business Processes and Business-to-Business-Integration (B2Bi)*: Starting from requirements for successful B2Bi development processes, languages and systems, we investigate the practicability and inter-operability of different approaches and platforms for the design and implementation of business processes.
- *Quality, esp. Robustness, Standard-conformance, Portability, Compatibility and Performance of Process-based Software and Service-oriented Systems*: In both, industry and academia, process languages have emerged, e.g. Windows Workflow (WF), Business Process Model and Notation (BPMN) and Web Services Business Process Execution Language (WS-BPEL). Although widely used in practice, current implementations of these languages and models are far from perfect. We work on metrics to compare such languages w.r.t. expressive power, conformance and portability as well as additional quality properties, such as installability, replaceability, adaptability and inter-operability. These metrics are developed and validated formally as well as evaluated practically. In the context of BPMN, we work on tools to check for and improve the standard compliance for human-centric process models on different layers of abstraction. Runtime environments for process languages with a focus on BPEL are investigated by means of a framework that eases the comparative test of different run-times and process engines.
- *Cloud Application Portability:* The hype surrounding the Cloud has lead to a variety of offerings that span the whole cloud stack. We examine important aspects of portability in cloud environments and enhance the portability of cloud applications by applying common standards between heterogeneous clouds. We make use of a holistic view of the cloud including important aspects like cloud specific restrictions, platform configurations, the deployment and life cycle of cloud applications.
- *Visual Programming- and Design-Languages*: The goal of this long-term effort is the utilization of visual metaphors and visualization techniques to make design- and programming languages more understandable and, hence, more easy-to-use. Currently, languages for designing and programming sensor networks are at the focus of this effort.

More information about our work can be found at `www.uni-bamberg.de/en/pi/`. If you have any questions or suggestions regarding this report or our work, don't hesitate to contact us.

Bamberg, June 2014 Guido Wirtz

Literature Survey of Performance Benchmarking Approaches of BPEL Engines

Cedric Röck, Simon Harrer

Abstract Despite the popularity of BPEL engines to orchestrate complex and executable processes, there are still only few approaches available to help find the most appropriate engine for individual requirements. One of the more crucial factors for such a middleware product in industry are the performance characteristics of a BPEL engine. There exist multiple studies in industry and academia testing the performance of BPEL engines, which differ in focus and method. We aim to compare the methods used in these approaches and provide guidance for further research in this area. Based on the related work in the field of performance testing, we created a process engine specific comparison framework, which we used to evaluate and classify nine different approaches that were found using the method of a systematical literature survey. With the results of the status quo analysis in mind, we derived directions for further research in this area.

Keywords SOA, BPEL, process engines, performance testing

Contact:

cedric.roeck@uni-bamberg.de, simon.harrer@uni-bamberg.de

Contents

List of Figures

List of Tables

1 Introduction

Over the past few years, the research concerning the Web Services Business Process Execution Language (BPEL) [27] made huge progress and focused on arising chances and challenges for businesses [7]. Based on service-oriented architectures (SOAs), one of the major trends in the development of business information systems, BPEL steadily became *the* standard for Web Service based business orchestration [21]. In addition, BPEL is also used to provision software in the cloud [6] and to implement partner specific views of a choreography, in the domain of business to business (B2B) integration [29]. Strongly connected to the growing popularity is the development of more complex systems, which also leads to an increased error-proneness of the developed software [3]. Therefore, the need to intensively test those SOAs also gains more importance. However, there still is a tremendous deficit in terms of proper testing tool support, which has been considered to be one of the major problems of SOAs [8].

A classic goal of performance testing has always been to compare competing platforms, allowing the selection of the best fitting product for particular needs [38]. Despite the grown acceptance of SOAs, performance testing in this area still lacks some major aspects. While the number of approaches steadily grows, the majority focuses solely on the evaluation of single Web services instead of middleware components or even complete systems [19]. However, "the middleware used to build a distributed application often determines the overall performance of the application" [11, p. 2] and should therefore be considered at least as carefully as the choice of partner services involved in a process. Looking at performance testing of BPEL processes and their engines, multiple studies have been conducted in industry and academia. However, there is no standardized benchmark, let alone a commonly agreed upon testing methodology [10]. As a consequence, the current view on the status quo and further research topics is clouded. In this work, we aim to provide a clear view by means of a selection of comparison criteria and a literature survey of the current approaches, as well as their classification into our framework.

Our contribution comprises three parts. First, a literature survey to find the available approaches that perform performance benchmarking of BPEL engines. Its method is outlined in Section 2 whereas the nine found approaches are described in Section 3. Second, a framework to compare performance benchmarking approaches of process engines. The framework and its criteria are detailed in Section 4 and applied onto the found approaches from Section 3, for which the results are given in Section 5. Third, in Section 6, directions on improving performance benchmarking of process engines in general and BPEL engines in particular based on an evaluation of the status quo comparison are given. Our work concludes in Section 7, which also outlines future work.

2 Method

To ensure that we find the relevant performance benchmarking approaches of BPEL engines, we conducted an extensive and systematical literature survey. At first, we searched in the archives of the ACM[1] as well as IEEE[2] and with the meta search engine Google scholar[3]. Additionally, we also leveraged the search engine Google itself for finding approaches in industry as well. For our search, we used the keyword combinations *bpel performance*, *bpel performance benchmark*, *bpel performance testing*, *middleware performance* and *web service performance* as our search queries. Next, we filtered down the search results in two phases: firstly, by reading the title, and secondly, by skimming the abstract. Thereafter, we performed forward and backward searches on the remaining references.

Using our method for the literature survey, we found nine BPEL middleware performance test projects, which are briefly described in Section 3. While the benchflow[4] project would be the perfect candidate at creating a benchmark for BPEL engines, it is still in its early stages. During our literature survey, we also came across a plethora of additional related studies which either focus on other middleware products, e.g., benchmarking the performance of enterprise service buses (ESBs) with ESBPerformance[5], or focus on performance modeling as well as prediction [33, 32, 16], and simulation [9]. However, these are out of scope.

[1]http://dl.acm.org
[2]http://www.computer.org/portal/web/search
[3]http://scholar.google.com
[4]http://www.iaas.uni-stuttgart.de/forschung/projects/benchflowE.php
[5]See http://esbperformance.org/.

3 Approaches

In this section, the nine BPEL performance benchmarking approaches found using the method from Section 2 are briefly described. Their full evaluation using the comparison framework from Section 4 is available in Section 5.

Multiple studies have originated from the engine vendors themselves. This includes Intel and Cape Clear with their *Cape Clear* BPEL Engine [18], Sun Microsystems with its *OpenESB* [35], and ActiveEndpoints with *ActiveVOS* [1]. In 2007, Intel and Cape Clear published a White Paper [18] in which they demonstrate their method to evaluate the performance and horizontal scaling capabilities of their BPEL engine *Cape Clear 7*. More specifically, they focus on the clustering capabilities of their engine, and among others on the ability to recover from single server crashes within a cluster. In contrast, ActiveEndpoints tested their BPEL engine *ActiveVOS* to verify the maximum load its engine is able to handle, whereas Sun Microsystems benchmarked their BPEL Engine within the *OpenESB* bundle [35] to spot performance degradation during development.

In academia, there are on the one hand approaches that benchmark the performance of BPEL engines to proof the validity of their contribution [30, 4, 13, 22] and on the other hand approaches that *solely* focus on benchmarking the performance of BPEL engines [23, 12]. Roller [30] worked on optimizing the performance of workflow engines in general and applied his techniques onto BPEL engines in a case study. To measure the success of his proposed improvements, he conducted performance tests before and after applying his optimizations. Hackmann et al. [13] developed a lightweight BPEL engine for mobile devices, named *Sliver*, and used performance benchmarking as a means to evaluate whether their engine has the expected characteristics to make it suitable for being executed on mobile devices in comparison to the typical BPEL engine that is targeted for powerful server systems. Liu et al. [22] propose a framework for fault-tolerant composition of transactional web services (*FACTS*) which can automatically introduce fault-handling logic into BPEL processes. To determine the performance impact of the additional fault-handling logic, they evaluated the performance costs corresponding to each fault-handling strategy on a single BPEL engine. Bianculli et al. [5, 4] developed *SOABench*, a testbed generation framework to benchmark the performance of arbitrary service-oriented middleware. To proof the applicability of their approach, they benchmarked the performance of BPEL engines in a case study. In contrast, Din et al. [12] created a configurable workload model to specifically benchmark the performance of different configurations of a BPEL engine including an evaluation, whereas Längerer et al. [23] solely conducted a performance comparison in their more practical study.

4 Comparison Framework and its Criteria

Performance testing refers to the usually technical examination of a system under test (SUT) with the aim to analyze and validate a product's characteristics. While it is known for allowing a measurement-based comparison of different products and platforms under similar conditions, hence offering valuable information for purchase decisions [38], performance testing can also refer to the model or simulation based evaluation, which can be realized at much earlier stages during the development cycle. Furthermore, performance testing is also used to identify bottlenecks and verify the requested quality of service (QoS) attributes in nearly finished software, i.e., whether the predefined nonfunctional performance requirements, which are often party of contractually binding service level agreements (SLA), are met [39].

According to Koziolek [20], the performance of a software component is influenced by five factors: its implementation, the usage profile, deployment platform, required services and resource contention. These influences are also illustrated in Figure 1. Focusing on the test of BPEL middleware, neither the usage profile, nor implementation details of the component are important for our work, as they should be identical for all systems under test. Instead, we focus mainly on the influence of the deployment platform, namely the BPEL engine, while adjusting and monitoring the behavior of required services, and available resources.

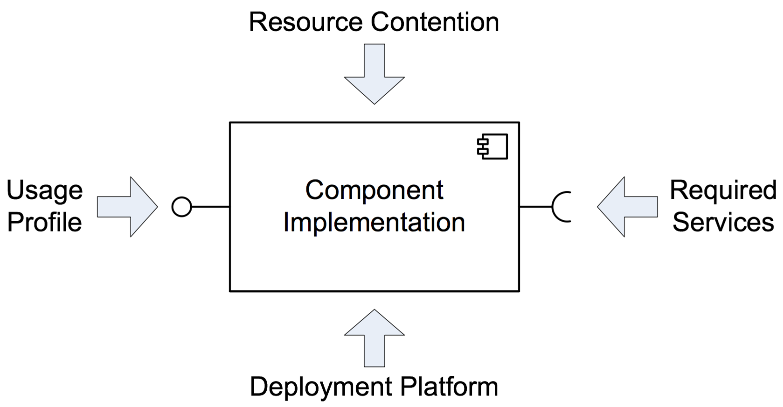

Figure 1: Influences on the performance of a software component [20]

Based on these influences, we created a framework to enable the classification of BPEL engine performance tests. In the following, we present four primary (see Section 4.1) and three secondary criteria (see Section 4.2), according to which BPEL performance test approaches can be analyzed and made comparable.

4.1 Primary Criteria

The primary criteria include the performance test type (Section 4.1.1), the workload along its categorization and injection strategies (Section 4.1.2), and the metrics according to which the workload execution on the system under test is measured (Section 4.1.3)

4.1.1 Types of Performance Tests

Measurement based performance tests can be executed in several ways depending on the intended purpose of the test. These different execution strategies establish the *types* of performance tests, namely *baseline*, *stress* and *load* tests [24].

Baseline tests measure the response time behavior of an application with only a single request (or at most a single concurrent user), and therefore represent the best-case scenario. Consequently, it can either be used for product comparison itself or as a benchmark baseline for other test types, e.g., for load tests [26, pp. 38-40].

A *load test* verifies the application behavior in typically expected situations, which also include load peaks with multiple concurrent users. It is often used to prove that an application meets performance related quality of service (QoS) attributes [24, 9] [25, p. 291].

Pushing an application beyond the expected level of load and peak conditions is called *stress testing*. Its goal lies in revealing bugs, as well as finding SLA violations and unexpected application behavior that only appear in extreme conditions [24]. The results are also referred to as worst-case scenario and hint at the application's capacity limits [26, pp. 38-40] [25, p. 291].

4.1.2 Workload: Definition, Categories and Injection

Workloads are defined as the sum of all inputs that are received by the system under test. They are considered to be one of the key aspects for assuring the validity of performance test results [34, 39, 38, 25]. In the context of BPEL engines, the workloads consists of the invoked processes and their dependent Web services. This also includes the requests starting the BPEL processes and the strategy for sending these requests.

According to Menascé [25, pp. 265-266], the workload of software performance benchmarks can be categorized into four groups, which are illustrated in Figure 2. Basic operations, the innermost circle, refer to the smallest offered operations, or all supported and standardized activities in the context of a BPEL engine and supply more fine-grained performance characteristics. Toy-benchmarks, usually implementing classic puzzles or being proof-of-work concepts, are of not much use for performance tests. Kernels, which are shown in the third circle, are core parts of real programs. They represent the most important or time consuming parts of applications. In our context, they can be seen as processes implementing patterns, e.g., the workflow patterns [36] that were extracted from a large corpus of real world processes. They provide an intermediate level in terms of granularity and realism. Real programs, or real processes in our case, are often seen as the most important category of workloads because their results are most accurate for estimating the performance of production programs or processes [2, 34]. Because of this, real workloads are also used in other domains, e.g., for benchmarking SQL databases with TPC-C[6].

The previously specified workloads also have to be injected into the system under test. These injection strategies can be distinguished by their time to fully inject a complete workload and

[6]See `www.tpc.org/tpcc/`.

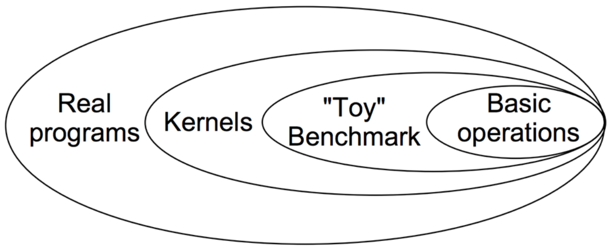

Figure 2: Benchmark hierarchy [25, p. 266]

to measure the execution at different points in time during the injection.

The workload injection strategies can be subdivided into a) *continuously* injecting the workload and b) injecting the workload with an *arrival rate*. *Continuously* injected workloads test the engine with a constant load, once reaching the plateau phase. This plateau, however, can either be reached with a *big bang* (default approach), or *stepwise*, by slowly increasing the number of concurrent users over a specific period [26, p. 41]. The *arrival rate* can be either fixed, e.g., 20 requests are sent every minute, or dynamic to simulate even more realistic peaks [34].

If the system under test and load generating clients are co-located, one has to closely observe the system for overloads. Therefore, they are often installed on separate systems, i.e., *distributed*. Moreover, a separation allows the load to be created from distributed clients, providing more realistic situations and assuring the saturation of the system under test.

4.1.3 Performance Metrics

The evaluation of software performance usually considers multiple metrics[7]. In this study, we use three main performance metrics: *latency*, *throughput* and *utilization*.

The *latency* or response time [11] defines the time span between sending a request and having received the full response. As shown in Figure 3, the response time is influenced by multiple factors. The execution time of partner processes t_e, the network delay t_l and the wrapping and parsing of XML messages t_w [31]. For BPEL processes, the classic response time must further be split-up into three distinguishable measures, the *execution* and the *creation* as well as the *update* response times. The *execution response time* refers to the classic response time, instantiating the process upon receiving the request as well as terminating it upon sending the response. In contrast, when having long running interactions, the *creation response time* measures the duration from sending a request instantiating the process until receiving the corresponding response, whereas the *update response time* measures the duration from correlating a sent request with an existing process instance until receiving the corresponding response [18].

Throughput is the most commonly used metric for software performance, defining the number of transactions that are completed within a specific time. In the context of BPEL engines, the term transaction can refer either to the completion of requests or process instances.

The *utilization* reflects the degree to which a capacity is used and refers to many parts of the

[7]In this context, metrics are also referred to as criteria, indicators or targets in literature.

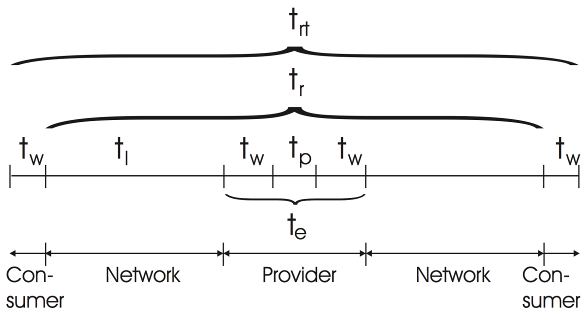

Figure 3: Relevant timings of a synchronous request-response transaction [31]

testbed, e.g., the network, database, server, or load-generating clients [26, p. 29].

As metrics differ in semantics and focus, they depend upon the test types. For example, a baseline test using any throughput metric is meaningless as there is only one active request at a time. Moreover, the throughput metric provides accurate results only for short-running or stateless transactions.

4.2 Secondary Criteria

Beyond the previous primary criteria, there are far more classification possibilities. We chose to further distinguish between three additional criteria. First, the license and the number of the system under test, being either open source or proprietary. Second, whether the setup of the test bed is automated or done manually, as this has an effect on the reproducibility of the experiment, which has previously been identified as one of the key aspects for a successful and widely accepted benchmark [17]. And third, as BPEL engines provide plenty of configuration options, we focus on whether the processes are executed in-memory or not, as this is a configuration option available within most BPEL engines, which we denote as *persistence* in our framework.

5 Results

The approaches in Section 3 are evaluated according to the criteria of our comparison framework in Section 4. The results are listed in Table 1 and Table 5, in which each approach is detailed in a separate row. The characteristics are grouped by their criteria of our comparison framework and shown in the header of the table in the first row. The cells represent the findings, being either filled or empty. Empty cells denote the absence of a criterion, whereas cells marked *n/a* denote that the approach did not provide any information regarding these criteria. We state the versions of the BPEL engines used in each approach in the following paragraphs, however, in some cases, the version is unknown.

Criteria	**Test Type**			**Workload**			**Workload Injection**				**Metrics**		
Approaches	Baseline	Load	Stress	Type	# of Processes	Ext. Service	Distributed	Continuously	Stepwise	Arrival Rate	Throughput	Latency	Utlization
SOABench [4]		x	x	basic	4	x	x	x		x	x	x	
OpenESB [35]		x		toy	1	n/a			x		x		
ActiveVOS [1]		x		real	2		n/a	n/a	n/a	n/a	x		
Sliver [13]	x			kernel	12	x		x				x	x
Workload model [12]		x		toy	1					x		x	
Intel & Cape Clear [18]		x		real	2	x	x	x			x	x	x
Roller [30]		x		real	1	x		x			x		
SWoM [23]		x		basic	2	x		x			x	x	
FACTS [22]		x		real	1	x	n/a	n/a	n/a	n/a		x	

Table 1: Literature analysis with primary criteria

The benchmark in *SOABench* [4] comprises two open source engines, *jBPM* and *Apache ODE* in the two versions v.1.3.3 and v.2.0-beta2, as well as the proprietary *ActiveVOS* v.5.0.2 Server Edition. However, at the time of the benchmark, the open source BPEL engine *ActiveBPEL* v5.0.2 has been part of the corresponding proprietary product, so the essential part of the software was open source. *SOABench* performs load tests using four BPEL processes. Two processes are built with the `flow` activity, the other two use only the `sequence` and the `while` activity. In addition, all four processes invoke mocked-up external services, thus, use the `invoke` activity as well. The performance is measured by means of response-time behavior (latency) and throughput metrics. Being based on the tool *Weevil* [37], *SOABench* features the automated generation, execution and analysis of testbeds. Hence, it allows repeating the test and reproduce the test results. Workloads are either injected based on a given arrival rate, or delayed by a *thinking time* between consecutive invocations.

The performance test of *OpenESB* [35] does not include other engines and has been executed as a load test focusing only on the throughput. The workload is a single process, using a mixture of all supported BPEL activities, i.e., a toy benchmark, and is injected stepwise.

Criteria / Approaches	SUT: Open Source	SUT: Proprietary	Config: Persistence	Testbed: Automated
SOABench [4]	2	1		x
OpenESB [35]	1			
ActiveVOS [1]		1	x	
Sliver [13]	2			
Workload model [12]	1			
Intel & Cape Clear [18]		1	x	
Roller [30]		1		
SWoM [23]		3		
FACTS [22]	1			

Table 2: Literature analysis with secondary criteria

The performance evaluation of *ActiveVOS* [1] does not include any other engines as well. The load test in this study monitors the engine's throughput, uses two functional processes, with one process being analyzed with persistence enabled or disabled.

Hackmann et al. [13] have compared *Sliver* with *ActiveBPEL* v2.0.1.1, both being open source engines. The baseline test measures the request latency as well as the memory utilization of the engines. The workload consists of twelve of the original workflow patterns [36], i.e., kernels, which have been realized as BPEL processes that utilize external services and are invoked in consecutive requests.

Validating their workload model, Din et al. [12] have performed a load test that focuses only on the response time behavior (latency) of the *ActiveBPEL* engine, distinguishing between the creation and update response times. The tested process uses correlations but does not call external services. The workload is injected by several virtual users over a total duration of two minutes, injecting 20 new processes per second.

The performance test of Intel and Cape Clear [18] has been executed as a load test and focuses on latency, throughput and utilization measures of the tested *Cape Clear 7.0 beta 2* engine. The two tested processes invoke external services and implement a functionality which is typical for industry processes. Additionally, the test also focuses on the effect of persistence on the performance metrics. The workload is continuously injected from a set of distributed clients.

Verifying Roller's [30] proposals, his performance test has been executed as a load test on the *IBM WebSphere 7.0 64bit* engine. His approach measures only throughput metrics. The tested workload is a single realistic BPEL process, including the invocation of external services, which has been continuously called by the testing clients.

Benchmarking three proprietary BPEL engines, the *Stuttgarter Workflow Maschine (SWoM) 1.0*, the *IBM WebSphere Process Server 6.0.2.3* and the *Oracle BPEL Process Manager*, Längerer et al. [23] have conducted a load test focusing on throughput and latency metrics. The workload, which is continuously injected, consists of two processes. One uses the `assign` activity, whereas the other one calls an external Web service with the `invoke` activity.

The load test of Liu et al. [22] tests the response time behavior (latency) of a single realistic process, which includes external services and has been deployed on the *ActiveBPEL* engine. It is not mentioned how the workload has been injected.

6 Discussion and Further Suggestions

This section analyses the overall findings of our evaluation and discusses which parts of BPEL performance testing should be strengthened in future work to increase the quality of their results. As Table 1 and Table 5 show, the approaches differ in many aspects, thus, they follow no common schema. Because of this, we focus on patterns per criteria, i.e., compare the results column-wise.

The performance test type of the approaches is mainly load testing, whereas, in addition, Bianculli et al. [4] also applies stress testing. Only Hackmann et al. [13] solely performs baseline tests. This shows, that there is room for further evaluations, by means of baseline and stress tests.

The workload categories differ for all approaches, with four using real workloads, one using kernels, and two using basic and toy each. Furthermore, the number of processes for each workload also varies, with four approaches using only a single process, three using only two, and one using four processes. In contrast, the remaining approach uses twelve processes. The majority (six out of nine) of the approaches use processes that invoke external services, thus the test always includes the performance characteristics of the `invoke` activity. For the remaining three approaches, two do not use external services, while it is unknown for the last one. Because of the known number of processes and the fact that external services are not always included, there seems to be lots of space for further work. For all workload categories, a larger corpus of processes would help to improve the meaningfulness of the test results. Regarding the basic category, we propose to have a process per feature to be able to compare their performance characteristics. This can reveal the strengths and weaknesses of a specific engine as well as make the engines and the features comparable. For kernel processes, we suggest to cover more pattern catalogs, whereas for real processes, we propose to use distinct real processes from various use cases in different domains. As services are a crucial part of BPEL processes, they should not be neglected in further studies as well.

Concerning the workload injection, two approaches did not mention it at all. However, for many approaches, the injection strategy is not explicitly stated, but can be deducted for some, but not all cases. As the workload injection is a crucial part of a performance evaluation, we strongly advise to explicitly state the selected strategy.

Regarding the metrics, latency and throughput are used by six approaches each, whereas only two measure utilization. In this context, Intel and Cape Clear [18] provide the most complete approach as they measure all three metrics, whereas one and two metrics are measured by five and three approaches, respectively. Therefore, we propose to focus more on the utilization as it has been neglected by many approaches, but can reveal interesting characteristics of the engines as well as ensure that there are no system overloads falsifying any results.

The number of engines under test range from one up to at most three per approach, limiting the meaningfulness of their results for buying decisions. Only three approaches [4, 13, 23] really compare the performance of multiple engines, while the remaining six evaluate the performance of a single engine. Furthermore, there exists only one comparison between the performance of open source and proprietary engines. However, Bianculli et al. [4] benchmark ActiveVOS v5.0.2 which incorporates the open source ActiveBPEL v5.0.2 engine. Therefore, this is not a comparison of open source with proprietary engines as basically only open source ones are compared, leaving room for further work in this area.

Regarding the configuration opportunities of the BPEL engines, only two approaches [1, 18] tested their engines in different configurations, namely either execute their processes in-memory or not. As engines have more than this single configuration option, it shows that this has been neglected in research, despite its importance. The two approaches that test this option solely evaluate a single engine. Therefore, when comparing more than one engine, it has to be ensured that all engines support this capability.

With only Bianculli et al. [4] allowing to automatically setup the testbed and trigger the test execution, it is very hard and tedious to redo the experiment for all other approaches. Moreover, only [4, 30, 23, 18] even published their detailed test setup, processes and tools, which are essential for the repeatability of these tests.

None of the nine approaches allow analyzing the influence of environmental aspects, for instance the system's hardware, the database or influences of long-running transactions on the engines' performance. However, modern multi-core systems and solid-state drives provide new challenges and opportunities for differentiation among different middleware products.

One additional problem is that none of the approaches take into account that BPEL engines greatly vary in their degree of support of the BPEL specification [14, 15], i.e., they implement different subsets of the BPEL features. We propose to take these results into account, creating and selecting appropriate workloads for the engines to be compared.

7 Conclusion and Future Work

In our work, we created a comparison framework with which existing performance benchmarking approaches of process engines, and BPEL engines in particular, can be classified. We applied our comparison framework to nine approaches, revealing their differences and similarities. Based on the findings, we derived guidance for further research in the areas which have been neglected so far.

In future work, we want to apply our comparison framework and method onto other process languages. For instance, the nowadays very popular Business Process Modeling and Notation (BPMN) [28] 2.0, which includes execution semantics [28, pp. 425-444], thus, is similar to BPEL in this regard. However, it is also interesting how to take human based tasks for performance approaches into account.

References

[1] Active Endpoints Inc. Assessing ActiveVOS Performance. `http://www.activevos.com/content/developers/technical_notes/assessing_activevos_performance.pdf`. 2014-01-30.

[2] A. Avritzer, J. Kondek, D. Liu, and E. J. Weyuker. Software Performance Testing Based on Workload Characterization. In *WOSP*, 2002.

[3] V. R. Basili and B. T. Perricone. Software Errors and Complexity: An Empirical Investigation. *CACM*, 27(1):42–52, 1984.

[4] D. Bianculli, W. Binder, and M. L. Drago. Automated Performance Assessment for Service-oriented Middleware: A Case Study on BPEL Engines. In *WWW*, 2010.

[5] D. Bianculli, W. Binder, and M. L. Drago. SOABench: Performance Evaluation of Service-oriented Middleware Made Easy. In *ICSE*, 2010.

[6] T. Binz, G. Breiter, F. Leymann, and T. Spatzier. Portable Cloud Services Using TOSCA. *IEEE Internet Computing*, 16(03):80–85, 2012.

[7] M. Bozkurt, M. Harman, and Y. Hassoun. Testing and Verification in Service-Oriented Architecture: A Survey. *Software Testing, Verification and Reliability*, 23(4):261–313, 2012.

[8] G. Canfora and M. D. Penta. Testing Services and Service-Centric Systems: Challenges and Opportunities. *IT Professional*, 8(2):10–17, 2006.

[9] S. Chandrasekaran, J. Miller, G. Silver, I. B. Arpinar, and A. Sheth. Composition, Performance Analysis and Simulation of Web Services. Technical report, LSDIS Lab, Computer Science Department, University of Georgia, Athens GA, USA, 2002.

[10] S. Chen, L. Bao, and P. Chen. OptBPEL: A Tool for Performance Optimization of BPEL Process. In C. Pautasso and E. Tanter, editors, *Software Composition*, volume 4954 of *LNCS*, pages 141–148. Springer Berlin Heidelberg, 2008.

[11] G. Denaro, A. Polini, and W. Emmerich. Early Performance Testing of Distributed Software Applications. In *WOSP*, 2004.

[12] G. Din, K.-P. Eckert, and I. Schieferdecker. A Workload Model for Benchmarking BPEL Engines. In *ICSTW*, 2008.

[13] G. Hackmann, M. Haitjema, C. Gill, and G.-C. Roman. Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. In *Service-Oriented Computing–ICSOC 2006*, pages 503–508. Springer, 2006.

[14] S. Harrer, J. Lenhard, and G. Wirtz. BPEL Conformance in Open Source Engines. In *SOCA*, 2012.

[15] S. Harrer, J. Lenhard, and G. Wirtz. Open Source versus Proprietary Software in Service-Orientation: The Case of BPEL Engines. In *ICSOC*, 2013.

[16] H. J. A. Holanda, G. C. Barroso, and A. d. B. Serra. SPEWS: A Framework for the Performance Analysis of Web Services Orchestrated with BPEL4WS. In *ICIW*, 2009.

[17] K. Huppler. The art of building a good benchmark. In R. Nambiar and M. Poess, editors, *Performance Evaluation and Benchmarking*, volume 5895, pages 18–30. Springer Berlin Heidelberg, 2009.

[18] Intel and Cape Clear. BPEL scalability and performance testing. White paper, Intel and Cape Clear, 2007.

[19] L. Juszczyk and S. Dustdar. Script-Based Generation of Dynamic Testbeds for SOA. In S. Dustdar, D. Schall, F. Skopik, L. Juszczyk, and H. Psaier, editors, *Socially Enhanced Services Computing*, pages 77–94. Springer Vienna, 2011.

[20] H. Koziolek. Performance Evaluation of Component-based Software Systems: A Survey. *Performance Evaluation*, 67:634–658, 2010. Special Issue on Software and Perf.

[21] T. v. Lessen, D. Lübke, and J. Nitzsche. *Geschäftsprozesse automatisieren mit BPEL [Automating Business Processes with BPEL]*. dpunkt.verlag, 2011.

[22] A. Liu, Q. Li, L. Huang, and M. Xiao. Facts: A framework for fault-tolerant composition of transactional web services. *Services Computing, IEEE Transactions on*, 3:46–59, 2010.

[23] C. Längerer, J. Rutschmann, and F. Schmitt. Performance-Vergleich von BPEL-Engines [Performance Comparison of BPEL Engines]. Fachstudie [technical report], University of Stuttgart, Germany, 2006.

[24] J. Meier, C. Farre, P. Bansode, S. Barber, and D. Rea. *Performance Testing Guidance for Web Applications: Patterns & Practices.* Microsoft Press, Redmond, WA, USA, 2007.

[25] D. A. Menascé. *Capacity Planning for Web Services: Metrics, Models, and Methods.* Prentice Hall, 2002.

[26] I. Molyneaux. *The Art of Application Performance Testing.* Theory in practice. O'Reilly Media, 1st ed edition, 2009.

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

[28] OMG. *Business Process Model and Notation*, January 2011. v2.0.

[29] C. Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, October 2003.

[30] D. Roller. *Throughput Improvements for BPEL Engines: Implementation Techniques and Measurements Applied to SWoM.* PhD thesis, IAAS, Stuttgart, Germany, 2013.

[31] F. Rosenberg, C. Platzer, and S. Dustdar. Bootstrapping Performance and Dependability Attributes of Web Services. In *ICWS*, 2006.

[32] D. Rud, M. Kunz, A. Schmietendorf, and R. Dumke. Performance Analysis in WS-BPEL-based Infrastructures. In *UKPEW*, 2007.

[33] D. Rud, A. Schmietendorf, and R. Dumke. Performance Modeling of WS-BPEL-Based Web Service Compositions. In *SCW*, 2006.

[34] A. J. Smith. Workloads (Creation and Use). *Commun. ACM*, 50(11):45–50, November 2007.

[35] Sun Microsystems. Benchmarking BPEL Service Engine. `http://wiki.open-esb.java.net/Wiki.jsp?page=BpelPerformance.html`. 2014-01-30.

[36] W. van der Aalst, A. ter Hofstede, B. Kiepuszewski, and A. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, July 2003.

[37] Y. Wang, M. J. Rutherford, A. Carzaniga, and A. L. Wolf. Automating Experimentation on Distributed Testbeds. In *International Conference on Automated Software Engineering (ASE)*, page 164. ACM Press, 2005.

[38] E. J. Weyuker and F. I. Vokolos. Experience with Performance Testing of Software Systems: Issues, an Approach, and Case Study. *IEEE Trans. Softw. Eng.*, 26:1147–1156, 2000.

[39] M. Woodside, G. Franks, and D. C. Petriu. The Future of Software Performance Engineering. In *FOSE*, 2007.

8 List of previous University of Bamberg reports

Bamberger Beiträge zur Wirtschaftsinformatik

Nr. 1 (1989)	Augsburger W., Bartmann D., Sinz E.J.: Das Bamberger Modell: Der Diplom-Studiengang Wirtschaftsinformatik an der Universität Bamberg (Nachdruck Dez. 1990)
Nr. 2 (1990)	Esswein W.: Definition, Implementierung und Einsatz einer kompatiblen Datenbankschnittstelle für PROLOG
Nr. 3 (1990)	Augsburger W., Rieder H., Schwab J.: Endbenutzerorientierte Informationsgewinnung aus numerischen Daten am Beispiel von Unternehmenskennzahlen
Nr. 4 (1990)	Ferstl O.K., Sinz E.J.: Objektmodellierung betrieblicher Informationsmodelle im Semantischen Objektmodell (SOM) (Nachdruck Nov. 1990)
Nr. 5 (1990)	Ferstl O.K., Sinz E.J.: Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)
Nr. 6 (1991)	Augsburger W., Rieder H., Schwab J.: Systemtheoretische Repräsentation von Strukturen und Bewertungsfunktionen über zeitabhängigen betrieblichen numerischen Daten
Nr. 7 (1991)	Augsburger W., Rieder H., Schwab J.: Wissensbasiertes, inhaltsorientiertes Retrieval statistischer Daten mit EISREVU / Ein Verarbeitungsmodell für eine modulare Bewertung von Kennzahlenwerten für den Endanwender
Nr. 8 (1991)	Schwab J.: Ein computergestütztes Modellierungssystem zur Kennzahlenbewertung
Nr. 9 (1992)	Gross H.-P.: Eine semantiktreue Transformation vom Entity-Relationship-Modell in das Strukturierte Entity-Relationship-Modell
Nr. 10 (1992)	Sinz E.J.: Datenmodellierung im Strukturierten Entity-Relationship-Modell (SERM)
Nr. 11 (1992)	Ferstl O.K., Sinz E. J.: Glossar zum Begriffsystem des Semantischen Objektmodells
Nr. 12 (1992)	Sinz E. J., Popp K.M.: Zur Ableitung der Grobstruktur des konzeptuellen Schemas aus dem Modell der betrieblichen Diskurswelt
Nr. 13 (1992)	Esswein W., Locarek H.: Objektorientierte Programmierung mit dem Objekt-Rollenmodell
Nr. 14 (1992)	Esswein W.: Das Rollenmodell der Organsiation: Die Berücksichtigung aufbauorganisatorische Regelungen in Unternehmensmodellen
Nr. 15 (1992)	Schwab H. J.: EISREVU-Modellierungssystem. Benutzerhandbuch
Nr. 16 (1992)	Schwab K.: Die Implementierung eines relationalen DBMS nach dem Client/Server-Prinzip
Nr. 17 (1993)	Schwab K.: Konzeption, Entwicklung und Implementierung eines computergestützten Bürovorgangssystems zur Modellierung von Vorgangsklassen und Abwicklung und Überwachung von Vorgängen. Dissertation

Nr. 18 (1993)	Ferstl O.K., Sinz E.J.: Der Modellierungsansatz des Semantischen Objektmodells
Nr. 19 (1994)	Ferstl O.K., Sinz E.J., Amberg M., Hagemann U., Malischewski C.: Tool-Based Business Process Modeling Using the SOM Approach
Nr. 20 (1994)	Ferstl O.K., Sinz E.J.: From Business Process Modeling to the Specification of Distributed Business Application Systems - An Object-Oriented Approach -. 1st edition, June 1994
	Ferstl O.K., Sinz E.J. : Multi-Layered Development of Business Process Models and Distributed Business Application Systems - An Object-Oriented Approach -. 2nd edition, November 1994
Nr. 21 (1994)	Ferstl O.K., Sinz E.J.: Der Ansatz des Semantischen Objektmodells zur Modellierung von Geschäftsprozessen
Nr. 22 (1994)	Augsburger W., Schwab K.: Using Formalism and Semi-Formal Constructs for Modeling Information Systems
Nr. 23 (1994)	Ferstl O.K., Hagemann U.: Simulation hierarischer objekt- und transaktionsorientierter Modelle
Nr. 24 (1994)	Sinz E.J.: Das Informationssystem der Universität als Instrument zur zielgerichteten Lenkung von Universitätsprozessen
Nr. 25 (1994)	Wittke M., Mekinic, G.: Kooperierende Informationsräume. Ein Ansatz für verteilte Führungsinformationssysteme
Nr. 26 (1995)	Ferstl O.K., Sinz E.J.: Re-Engineering von Geschäftsprozessen auf der Grundlage des SOM-Ansatzes
Nr. 27 (1995)	Ferstl, O.K., Mannmeusel, Th.: Dezentrale Produktionslenkung. Erscheint in CIM-Management 3/1995
Nr. 28 (1995)	Ludwig, H., Schwab, K.: Integrating cooperation systems: an event-based approach
Nr. 30 (1995)	Augsburger W., Ludwig H., Schwab K.: Koordinationsmethoden und -werkzeuge bei der computergestützten kooperativen Arbeit
Nr. 31 (1995)	Ferstl O.K., Mannmeusel T.: Gestaltung industrieller Geschäftsprozesse
Nr. 32 (1995)	Gunzenhäuser R., Duske A., Ferstl O.K., Ludwig H., Mekinic G., Rieder H., Schwab H.-J., Schwab K., Sinz E.J., Wittke M: Festschrift zum 60. Geburtstag von Walter Augsburger
Nr. 33 (1995)	Sinz, E.J.: Kann das Geschäftsprozeßmodell der Unternehmung das unternehmensweite Datenschema ablösen?
Nr. 34 (1995)	Sinz E.J.: Ansätze zur fachlichen Modellierung betrieblicher Informationssysteme - Entwicklung, aktueller Stand und Trends -
Nr. 35 (1995)	Sinz E.J.: Serviceorientierung der Hochschulverwaltung und ihre Unterstützung durch workflow-orientierte Anwendungssysteme
Nr. 36 (1996)	Ferstl O.K., Sinz, E.J., Amberg M.: Stichwörter zum Fachgebiet Wirtschaftsinformatik. Erscheint in: Broy M., Spaniol O. (Hrsg.): Lexikon Informatik und Kommunikationstechnik, 2. Auflage, VDI-Verlag, Düsseldorf 1996

Nr. 37 (1996)	Ferstl O.K., Sinz E.J.: Flexible Organizations Through Object-oriented and Transaction-oriented Information Systems, July 1996
Nr. 38 (1996)	Ferstl O.K., Schäfer R.: Eine Lernumgebung für die betriebliche Aus- und Weiterbildung on demand, Juli 1996
Nr. 39 (1996)	Hazebrouck J.-P.: Einsatzpotentiale von Fuzzy-Logic im Strategischen Management dargestellt an Fuzzy-System-Konzepten für Portfolio-Ansätze
Nr. 40 (1997)	Sinz E.J.: Architektur betrieblicher Informationssysteme. In: Rechenberg P., Pomberger G. (Hrsg.): Handbuch der Informatik, Hanser-Verlag, München 1997
Nr. 41 (1997)	Sinz E.J.: Analyse und Gestaltung universitärer Geschäftsprozesse und Anwendungssysteme. Angenommen für: Informatik '97. Informatik als Innovationsmotor. 27. Jahrestagung der Gesellschaft für Informatik, Aachen 24.-26.9.1997
Nr. 42 (1997)	Ferstl O.K., Sinz E.J., Hammel C., Schlitt M., Wolf S.: Application Objects – fachliche Bausteine für die Entwicklung komponentenbasierter Anwendungssysteme. Angenommen für: HMD – Theorie und Praxis der Wirtschaftsinformatik. Schwerpunkheft ComponentWare, 1997
Nr. 43 (1997):	Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using the Semantic Object Model (SOM) – A Methodological Framework - . Accepted for: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1997
	Ferstl O.K., Sinz E.J.: Modeling of Business Systems Using (SOM), 2nd Edition. Appears in: P. Bernus, K. Mertins, and G. Schmidt (ed.): Handbook on Architectures of Information Systems. International Handbook on Information Systems, edited by Bernus P., Blazewicz J., Schmidt G., and Shaw M., Volume I, Springer 1998
Nr. 44 (1997)	Ferstl O.K., Schmitz K.: Zur Nutzung von Hypertextkonzepten in Lernumgebungen. In: Conradi H., Kreutz R., Spitzer K. (Hrsg.): CBT in der Medizin – Methoden, Techniken, Anwendungen -. Proceedings zum Workshop in Aachen 6. – 7. Juni 1997. 1. Auflage Aachen: Verlag der Augustinus Buchhandlung
Nr. 45 (1998)	Ferstl O.K.: Datenkommunikation. In. Schulte Ch. (Hrsg.): Lexikon der Logistik, Oldenbourg-Verlag, München 1998
Nr. 46 (1998)	Sinz E.J.: Prozeßgestaltung und Prozeßunterstützung im Prüfungswesen. Erschienen in: Proceedings Workshop „Informationssysteme für das Hochschulmanagement". Aachen, September 1997
Nr. 47 (1998)	Sinz, E.J.:, Wismans B.: Das „Elektronische Prüfungsamt". Erscheint in: Wirtschaftswissenschaftliches Studium WiSt, 1998
Nr. 48 (1998)	Haase, O., Henrich, A.: A Hybrid Respresentation of Vague Collections for Distributed Object Management Systems. Erscheint in: IEEE Transactions on Knowledge and Data Engineering
Nr. 49 (1998)	Henrich, A.: Applying Document Retrieval Techniques in Software Engineering Environments. In: Proc. International Conference on Database and Expert Systems

	Applications. (DEXA 98), Vienna, Austria, Aug. 98, pp. 240-249, Springer, Lecture Notes in Computer Sciences, No. 1460
Nr. 50 (1999)	Henrich, A., Jamin, S.: On the Optimization of Queries containing Regular Path Expressions. Erscheint in: Proceedings of the Fourth Workshop on Next Generation Information Technologies and Systems (NGITS'99), Zikhron-Yaakov, Israel, July, 1999 (Springer, Lecture Notes)
Nr. 51 (1999)	Haase O., Henrich, A.: A Closed Approach to Vague Collections in Partly Inaccessible Distributed Databases. Erscheint in: Proceedings of the Third East-European Conference on Advances in Databases and Information Systems – ADBIS'99, Maribor, Slovenia, September 1999 (Springer, Lecture Notes in Computer Science)
Nr. 52 (1999)	Sinz E.J., Böhnlein M., Ulbrich-vom Ende A.: Konzeption eines Data Warehouse-Systems für Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule“ im Rahmen der 29. Jahrestagung der Gesellschaft für Informatik, Paderborn, 6. Oktober 1999
Nr. 53 (1999)	Sinz E.J.: Konstruktion von Informationssystemen. Der Beitrag wurde in geringfügig modifizierter Fassung angenommen für: Rechenberg P., Pomberger G. (Hrsg.): Informatik-Handbuch. 2., aktualisierte und erweiterte Auflage, Hanser, München 1999
Nr. 54 (1999)	Herda N., Janson A., Reif M., Schindler T., Augsburger W.: Entwicklung des Intranets SPICE: Erfahrungsbericht einer Praxiskooperation.
Nr. 55 (2000)	Böhnlein M., Ulbrich-vom Ende A.: Grundlagen des Data Warehousing. Modellierung und Architektur
Nr. 56 (2000)	Freitag B, Sinz E.J., Wismans B.: Die informationstechnische Infrastruktur der Virtuellen Hochschule Bayern (vhb). Angenommen für Workshop "Unternehmen Hochschule 2000" im Rahmen der Jahrestagung der Gesellschaft f. Informatik, Berlin 19. - 22. September 2000
Nr. 57 (2000)	Böhnlein M., Ulbrich-vom Ende A.: Developing Data Warehouse Structures from Business Process Models.
Nr. 58 (2000)	Knobloch B.: Der Data-Mining-Ansatz zur Analyse betriebswirtschaftlicher Daten.
Nr. 59 (2001)	Sinz E.J., Böhnlein M., Plaha M., Ulbrich-vom Ende A.: Architekturkonzept eines verteilten Data-Warehouse-Systems für das Hochschulwesen. Angenommen für: WI-IF 2001, Augsburg, 19.-21. September 2001
Nr. 60 (2001)	Sinz E.J., Wismans B.: Anforderungen an die IV-Infrastruktur von Hochschulen. Angenommen für: Workshop „Unternehmen Hochschule 2001“ im Rahmen der Jahrestagung der Gesellschaft für Informatik, Wien 25. – 28. September 2001

Änderung des Titels der Schriftenreihe *Bamberger Beiträge zur Wirtschaftsinformatik* in *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* ab Nr. 61

Note: The title of our technical report series has been changed from *Bamberger Beiträge zur Wirtschaftsinformatik* to *Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik* starting with TR No. 61

Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik

Nr. 61 (2002)	Goré R., Mendler M., de Paiva V. (Hrsg.): Proceedings of the International Workshop on Intuitionistic Modal Logic and Applications (IMLA 2002), Copenhagen, July 2002.
Nr. 62 (2002)	Sinz E.J., Plaha M., Ulbrich-vom Ende A.: Datenschutz und Datensicherheit in einem landesweiten Data-Warehouse-System für das Hochschulwesen. Erscheint in: Beiträge zur Hochschulforschung, Heft 4-2002, Bayerisches Staatsinstitut für Hochschulforschung und Hochschulplanung, München 2002
Nr. 63 (2005)	Aguado, J., Mendler, M.: Constructive Semantics for Instantaneous Reactions
Nr. 64 (2005)	Ferstl, O.K.: Lebenslanges Lernen und virtuelle Lehre: globale und lokale Verbesserungspotenziale. Erschienen in: Kerres, Michael; Keil-Slawik, Reinhard (Hrsg.); Hochschulen im digitalen Zeitalter: Innovationspotenziale und Strukturwandel, S. 247 – 263; Reihe education quality forum, herausgegeben durch das Centrum für eCompetence in Hochschulen NRW, Band 2, Münster/New York/München/Berlin: Waxmann 2005
Nr. 65 (2006)	Schönberger, Andreas: Modelling and Validating Business Collaborations: A Case Study on RosettaNet
Nr. 66 (2006)	Markus Dorsch, Martin Grote, Knut Hildebrandt, Maximilian Röglinger, Matthias Sehr, Christian Wilms, Karsten Loesing, and Guido Wirtz: Concealing Presence Information in Instant Messaging Systems, April 2006
Nr. 67 (2006)	Marco Fischer, Andreas Grünert, Sebastian Hudert, Stefan König, Kira Lenskaya, Gregor Scheithauer, Sven Kaffille, and Guido Wirtz: Decentralized Reputation Management for Cooperating Software Agents in Open Multi-Agent Systems, April 2006
Nr. 68 (2006)	Michael Mendler, Thomas R. Shiple, Gérard Berry: Constructive Circuits and the Exactness of Ternary Simulation
Nr. 69 (2007)	Sebastian Hudert: A Proposal for a Web Services Agreement Negotiation Protocol Framework . February 2007
Nr. 70 (2007)	Thomas Meins: Integration eines allgemeinen Service-Centers für PC-und Medientechnik an der Universität Bamberg – Analyse und Realisierungs-Szenarien. February 2007 (out of print)
Nr. 71 (2007)	Andreas Grünert: Life-cycle assistance capabilities of cooperating Software Agents for Virtual Enterprises. März 2007
Nr. 72 (2007)	Michael Mendler, Gerald Lüttgen: Is Observational Congruence on μ-Expressions Axiomatisable in Equational Horn Logic?
Nr. 73 (2007)	Martin Schissler: out of print
Nr. 74 (2007)	Sven Kaffille, Karsten Loesing: Open chord version 1.0.4 User's Manual. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 74, Bamberg University, October 2007. ISSN 0937-3349.

Nr. 75 (2008)	Karsten Loesing (Hrsg.): Extended Abstracts of the Second *Privacy Enhancing Technologies Convention* (PET-CON 2008.1). Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 75, Bamberg University, April 2008. ISSN 0937-3349.
Nr. 76 (2008)	Gregor Scheithauer, Guido Wirtz: Applying Business Process Management Systems – A Case Study. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 76, Bamberg University, May 2008. ISSN 0937-3349.
Nr. 77 (2008)	Michael Mendler, Stephan Scheele: Towards Constructive Description Logics for Abstraction and Refinement. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 77, Bamberg University, September 2008. ISSN 0937-3349.
Nr. 78 (2008)	Gregor Scheithauer, Matthias Winkler: A Service Description Framework for Service Ecosystems. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 78, Bamberg University, October 2008. ISSN 0937-3349.
Nr. 79 (2008)	Christian Wilms: Improving the Tor Hidden Service Protocol Aiming at Better Performances. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 79, Bamberg University, November 2008. ISSN 0937-3349.
Nr. 80 (2009)	Thomas Benker, Stefan Fritzemeier, Matthias Geiger, Simon Harrer, Tristan Kessner, Johannes Schwalb, Andreas Schönberger, Guido Wirtz: QoS Enabled B2B Integration. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 80, Bamberg University, May 2009. ISSN 0937-3349.
Nr. 81 (2009)	Ute Schmid, Emanuel Kitzelmann, Rinus Plasmeijer (Eds.): Proceedings of the ACM SIGPLAN Workshop on *Approaches and Applications of Inductive Programming* (AAIP'09), affiliated with ICFP 2009, Edinburgh, Scotland, September 2009. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 81, Bamberg University, September 2009. ISSN 0937-3349.
Nr. 82 (2009)	Ute Schmid, Marco Ragni, Markus Knauff (Eds.): Proceedings of the KI 2009 Workshop *Complex Cognition*, Paderborn, Germany, September 15, 2009. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 82, Bamberg University, October 2009. ISSN 0937-3349.
Nr. 83 (2009)	Andreas Schönberger, Christian Wilms and Guido Wirtz: A Requirements Analysis of Business-to-Business Integration. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 83, Bamberg University, December 2009. ISSN 0937-3349.
Nr. 84 (2010)	Werner Zirkel, Guido Wirtz: A Process for Identifying Predictive Correlation Patterns in Service Management Systems. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 84, Bamberg University, February 2010. ISSN 0937-3349.
Nr. 85 (2010)	Jan Tobias Mühlberg und Gerald Lüttgen: Symbolic Object Code Analysis. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 85, Bamberg University, February 2010. ISSN 0937-3349.

Nr. 86 (2010) Werner Zirkel, Guido Wirtz: Proaktives Problem Management durch Eventkorrelation – ein *Best Practice* Ansatz. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 86, Bamberg University, August 2010. ISSN 0937-3349.

Nr. 87 (2010) Johannes Schwalb, Andreas Schönberger: Analyzing the Interoperability of WS-Security and WS-ReliableMessaging Implementations. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 87, Bamberg University, September 2010. ISSN 0937-3349.

Nr. 88 (2011) Jörg Lenhard: A Pattern-based Analysis of WS-BPEL and Windows Workflow. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 88, Bamberg University, March 2011. ISSN 0937-3349.

Nr. 89 (2011) Andreas Henrich, Christoph Schlieder, Ute Schmid [eds.]: Visibility in Information Spaces and in Geographic Environments – Post-Proceedings of the KI’11 Workshop. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 89, Bamberg University, December 2011. ISSN 0937-3349.

Nr. 90 (2012) Simon Harrer, Jörg Lenhard: Betsy - A BPEL Engine Test System. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 90, Bamberg University, July 2012. ISSN 0937-3349.

Nr. 91 (2013) Michael Mendler, Stephan Scheele: On the Computational Interpretation of CKn for Contextual Information Processing - Ancillary Material. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 91, Bamberg University, May 2013. ISSN 0937-3349.

Nr. 92 (2013) Matthias Geiger: BPMN 2.0 Process Model Serialization Constraints. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 92, Bamberg University, May 2013. ISSN 0937-3349.

Nr. 93 (2014) Cedric Röck, Simon Harrer: Literature Survey of Performance Benchmarking Approaches of BPEL Engines. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 93, Bamberg University, May 2014. ISSN 0937-3349.