"A Decision Support System for the Performance Benchmarking of Workflow Management Systems"

Marigianna Skouradaki[1], Tayyaba Azad, Uwe Breitenbuecher[1], Oliver Kopp[2], Frank Leymann[1]

[1]Institute of Architecture of Application Systems, University of Stuttgart, Germany
[2]Institute of Parallel and Distributed Systems, University of Stuttgart, Germany

A Decision Support System for the Performance Benchmarking of Workflow Management Systems

Marigianna Skouradaki[1], Tayyaba Azad, Uwe Breitenbücher[1], Oliver Kopp[2], and Frank Leymann[1]

[1]IAAS, [2]IPVS, University of Stuttgart, Germany
{firstname.lastname}@iaas.uni-stuttgart.de

Abstract. Along with the growing popularity of the Workflow Management Systems, the performance and efficiency of their underlying technology becomes crucial for the business. The development of a representative benchmark for Workflow Management Systems is very challenging, as one needs to realistically stress the different underlying components. However, structured information on how to do so is generally missing. Thus, the users need to arbitrarily make crucial design decisions or to study complex standard benchmarks before designing a benchmark. In this work, we propose a Decision Support System to ease the decision making of the desigh of benchmarks for Workflow Management Systems. We present the conceptual models of the Decision Support System and provide a prototypical implementation of it. Finally, we validate the functionality of our implementation with representative use cases.

Keywords: Decision Support System, Benchmarking, Workflow Managament Systems

1 Introduction

The representativeness and reliability of a benchmark comprises its key characteristics, otherwise it may indicate misleading results [28]. As Moscato [11] characteristically says there are “Lies, Damned Lies and Benchmarks”. Therefore, standard benchmarks are generally developed and supported by corporations like the Standard Performance Evaluation Corporation (SPEC) [21] and the Transaction Processing Performance Council (TPC) [25]. In order to develop a new benchmark or to apply an existing benchmark on a middleware system, practitioners need to comprehend and analyze a set of standard benchmarks.

The BenchFlow project [1] comprises an academic effort to create the first standard benchmark for the performance of Business Process Model and Notation 2.0 (BPMN 2.0) compliant Workflow Management Systems (WfMS). Thus, in this case information otherwise acquired by the companies is currently

[1] http://www.iaas.uni-stuttgart.de/forschung/projects/benchflow.php

unavailable, as companies cannot share their data to maintain their corporate assets. In order to ensure the reliability and representativeness of the BenchFlow benchmark, we have conducted a literature review on standard benchmarks of related middleware technologies and custom benchmarks of WfMSs. The goal of this review is to filter the relevant information and answer research questions as the following: 1) What are the key decision points for the construction of new WfMS benchmarks? 2) What are the dependencies that affect the design of the participating artifacts in a benchmark? 3) How could one utilize historical data to take key decisions for the design of a WfMS benchmark?

To get an in-depth knowledge of the related existing benchmarks, we focused on standardized benchmarks that were published by industry-accepted consortia such as SPEC [21] and TPC [25], as well as state-of-the art custom benchmarks that target to measure the performance of WfMS. We have gathered the information in a knowledge base and offer it as a Decision Support System (DSS) [16], named as *DSS4MiddlewarePBenchmarking*to support stakeholders in future decisions concerning the construction of WfMS benchmarks.

Thus, the original scientific contributions of this work are to:

1. Investigate the related standard and custom benchmarks in order to outline the current trends on benchmarking and DSS in regards to benchmarking;
2. Analyze the requirements for the definition of the *DSS4MiddlewarePBenchmarking*;
3. Identify the artefacts that are relevant for the construction of new WfMS benchmarks and their underlying dependencies;
4. Provide a prototypical implementation of the *DSS4MiddlewarePBenchmarking*;
5. Validate the solution through use case scenarios.

The remainder of this paper is structured as follows: Sect. 2 provides background information on current standard benchmarks for related middleware and custom benchmarks and introduces the concept of Decision Support Systems; Sect. 3 specifies the functional and non-functional requirements, introduces the conceptual model on which our system relies, and the design dependencies that stem from the participating artifacts; Sect. 4 explains the technologies used for the implementation of the *DSS4MiddlewarePBenchmarking*; Sect. 5 validates the system through use cases; Sect. 6 overviews existing work for Decision Support Systems and Sect. 7 concludes and proposes our plans for future work.

2 Background

2.1 Standardised Middleware Benchmarks

There exist a number of organisations that provide standard benchmarks. Two of the most relevant ones that focus on performance benchmarking are the Standard Performance Evaluation Corporation (SPEC) [21] and Transaction Processing Performance Council (TPC) [25]. The following sections summarise the most relevant benchmarks published by these consortia.

SPEC ® JMS 2007 [22] provides the assessment of performance for Message Oriented Middleware (MOM) servers based on the Java Message Service (JMS). The main purpose of the SPEC ® JMS 2007 benchmark is to support a standard workload and metrics in order to provide an in depth performance analysis of all the individual components comprising the JMS-based MOM platforms. In order to avoid the scalability limitations within the workload of SPEC ® JMS 2007, users are able to increase the number of destinations (queues and topics). The number of messages per destination can be increased or users can scale the workload in a customised manner. The application scenario for SPEC ® JMS 2007 involves the model of a supply chain for a supermarket. The supermarket company, its stores, its distribution centers and its suppliers are the different participants that are involved in this scenario. The requirements discussed in the previous section are applied to this scenario. It allows a clear specification of interactions that stress defined features of the JMS Servers. For instance, publish/subscribe or peer-to-peer communication as well as diverse message types. Moreover, there are no limitations on scalability of the workload, the number of supermarkets can be increased and the number of products offered by a supermarket can also be increased.

SPECjbb ® 2015 [23] provides the performance measurement based on the latest Java application features. It is applicable to all organisations that are interested in measuring Java server performance. The benchmark includes a model that illustrates a supermarket company with an Informations Technology (IT) infrastructure that deals with point-of-sale requests, online purchases and data-mining operations. The metric included in the benchmark is a pure throughput metric. SPECjbb ® 2015also supports visualisation and cloud environments [23].

TPC-C [26] is an On-Line Transaction Processing (OLTP) benchmark. It is an improved version of the previously published benchmarks due to its multiple transaction types, greater complexity of the database and the overall execution structure. TPC-C includes five concurrent transactions of different types and complexity. The involved database is made up of nine different types of tables with large sizes in regards to recording and population. The TPC-C benchmark is measured in transactions per minute (tpmC). It simulates a running computing environment where users execute transactions against a database. The transactions contain entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the corresponding warehouse. This benchmark is not restricted to a specific business area, but targets different market sectors [26].

TPC-E [27] is also an OLTP benchmark that resembles the OLTP workload of a brokerage firm. The main target of the benchmark is the central database that executes transactions associated to the company's customer accounts. Despite the fact that the business model covered by the TPC-E is a brokerage firm, the benchmark can be used on various modern OLTP systems. The benchmark specifies the required combination of transactions it should be able to handle. The TPC-E benchmark is measured in transactions per second (tpsE) [27].

2.2 Workflow Management Systems Benchmarks

A standard benchmark for Workflow Management Systems is not available yet [20]. Threrefore, we are presenting the state-of the art in custom benchmarks that have been proposed during the last years.

SOABench [3] targets to assess the middleware performance in the context of a Service-Oriented Architecture (SOA). In general, it proposes the automatic generation and execution of testbeds for benchmarking SOAs. As a use case for the proposed framework workflow engines supporting the Web Services Business Process Execution Language (WS-BPEL 2.0) [14] are used as the Systems Under Test (SUT). For the experiments four different workloads are defined which express basic control flow structures of the WS-BPEL language (i. e., Sequential, FlowNoDep, Flow, While). For each defined workload the authors separate four different load situations that span from low to high system loading. The defined metric is limitted to response time for all the executed experiments. The tests target two open source and one proprietary WfMSs.

ActiveVOS [1] targets solely the performance of the ActivevVOS WS-BPEL WfMS. For the performance tests four workload mixes are used. The tests are executed on a variable request rate of maximum 50 users. Althought the configuration of the infrastructure underlying the WS-BPEL engine is described in detail, results on the performance tests are not further discussed.

Sliver [6] is a WS-BPEL WfMS for mobile devices. In order to evaluate the performance of the proposed protytipical implementation, the authors are evaluating the Sliver engine with twelve WS-BPEL patterns. The performance of the Sliver WfMS is measured with respect to three different infrastructures (PC, PDA and Phone) and compared to one more WfMS (i. e., ActiveBPEL) on a PC infrastructure. Also in this case, the examined metric is the reponse time of the WfMS.

Dit et al. [4] propose a workload model for benchmarking WS-BPEL WfMS. For the derivation of the model the authors simulate real world traffic conditions in order to better define the end-users that characterize it. For the performance tests a two phase workload is defined that implements a WS-BPEL correlation. The SUT is the ActiveBPEL engine[2]. The experiments run for 2 minutes in total and simulate 2000 users. The defined metrics are success/failure rate, response times and round-trip delays.

Intel & Cape Clear [8] asseses the performance of the Cape Clear WS-BPEL engine running on Intel ®servers. This white paper introduces the concept of executing different real-world process models for the different performance tests. Consequently, with a focus on execution of the throughput test, a loan-approval process is executed, a correlation-rich process model for the load and recovery tests. For the throughput tests (in terms of transactions per minute) 1 to 8 servers are used, as well as a variety of different clients, which range from 20 to 100. During the load tests the SUT is loaded with up to one million live long running instances. The measurements are taken in pre-defined times points.

[2] http://www.activevos.com/content/developers/education/sample_active_bpel_admin_api/doc/index.html

SWoM [17] conducts load tests on one proprietary engine in order to measure throughput. The defined workload for execution is four simple WS-BPEL processes. The injected WS-BPEL process contains also the invocation of external services, which is continuously called by the testing clients. Each experiment executed 24.000 instances in a total of approximately 40 minutes. The client emulates 30 requestors whose think time for subsequent requests was adjusted to run with respect to the CPU load, keeping it in between of 50% and 60%.

Micro-Benchmark BPMN 2.0 Workflow Management Systems [19] constitutes a micro-benchmark (i. e., a toy benchmark) that targets to measure the preformance of BPMN 2.0 [9] WfMSs. The goal of the micro benchmark is to define the correlation between the fundamental structures of the BPMN 2.0 language to the performance of the BPMN 2.0 WfMS, as well as reveal any potential bottlenecks. Six different workload mixes are defined for the micro-benchmark, which are derived from the subset of the BPMN 2.0 control-flow workflow patterns that were actually found to be applicable in real-world process models [12]. The control flow patterns used are the sequence flow, the exclusive choice, the simple merge, the parallel split, the synchronization patterns [**?**]. The benchmark is run by an automated, isolated, reproducible, and reliable benchmark environment, called the BenchFlow framework [5]. The analysed metrics are the reponse time, the throughput (business process instances per second), and resource utilization for the CPU and RAM.

2.3 Decision Support Systems and Decision Support

The term Decision Support System is used to describe an information system that maintains decision-making tasks of a business or an organisation [16]. Generally, the different types of DSSs can be summarised as follows: *Data-driven DSS* provides access to large knowledge bases in order to extract information; *Communication-driven DSS* supports the shared access on a specific task where more than one person is involved in working on it; *Document-driven DSS* data is retrieved and manipulated in form of a document; *Knowledge-driven DSS* or *Expert Systems* provide professional problem-solving in terms of defined rules and procedures; *Model-driven DSS* provide functionality by offering different models for which the data and parameters are provided by the user. The three different components inside a DSS are: the knowledge base where all relevant information is stored, the conceptual model that defines different decision criteria and the user interface that presents the required output [16].

3 Concept and Specification

3.1 Requirements Analysis

This section deals with the functional and non-functional requirements that were considered to be relevant for our DSS, here after refered to as *DSS4Middle-warePBenchmarking*. The following list describes the most important recognized

Functional Requirements (FR). We concluded on them through an thorough literature review and practical experience.

- FR_1 *Visualisation of the Decision Support System*: Extract data from the *DSS-4MiddlewarePBenchmarking*in a human recognisable manner. In order to provide support to the decision-making for performance benchmarking.
- FR_2 *Management and Configuration*: Create, Read, Update and Delete (CRUD) operations should be provided for each conceptual entity of the database and should be accessed through the user interface level. It should allow the user to i) create or add new entries; ii) read, retrieve, search, or view existing entries; iii) update or edit existing entries; iv) delete/deactivate existing entries.
- FR_3 *Dynamic Querying*: The system should provide dynamic querying by excluding dependencies that cannot be combined together. For example, in the case requesting information for a Java Server, the system should exclude parameters that are solely related to WfMS.
- FR_4 *Reporting Support*: The *DSS4MiddlewarePBenchmarking*should support the decision making by providing responses that are logically reduced by the provided input, or indicate failure to respond if a corresponding reply is not available.
- FR_5 *Realiability of Data*: The *DSS4MiddlewarePBenchmarking*should contain data coming from reliable resources such as scientific papers and documentations of standard benchmarks.

Moreover, the *DSS4MiddlewarePBenchmarking*should also satisfy the non-functional requirements described in the following list:

- NFR_1 *Usability*: The developer should specify all main and relevant benchmark characteristics according to the decision tree with an easy and interactive interface. The interface should be graphical, web-based, user-friendly and should allow querying the knowledge base. The software platform should be self-explanatory to the user, and the user should know exactly what the required steps are.
- NFR_2 *Consistency*: The system should allow the user to view the results of the selected criteria in a consistent manner. All included operations should always behave in a predictable and consistent manner.
- NFR_3 *Performance*: The knowledge base conducted from the decision tree should be exposed as a Representational State Transfer (REST)-ful web application. The DSS should respond with success or failure in realistic times (seconds).
- NFR_4 *Web-based development*: Portability, cross-platform support and a user-friendly interface should be supported through a web-based solution.
- NFR_5 *Easy installation*: The configuration and installation of the software should be rapid and easy.
- NFR_6 *Guidelines, compliance & documentation*: For future extension and modification the source code should be supported by following well established software engineering guidelines and descriptive documentation.

3.2 Conceptual Model

This subsection introduces a conceptual model presented in the form of an Entity Relationship (ER) diagram, shown in Fig. 1. The ER diagram represents the key decision points as entities and the corresponding dependencies as relationships. The knowledge base and functionality of the *DSS4MiddlewarePBenchmarking*rely on this conceptual model. The entities described in the conceptual model were recognized through the extended review of the benchmarks described in Sect. 2. Due to the fact that all benchmarks have similar domains of application, their structure is overlapping in many aspects. The recognized entities are derived from the recognized overlapping information. Concepts that did not share a common usage in all the studied benchmarks were not included in the ER model. More particularly, the conceptual model includes relevant information that is required for the construction of new domain-related benchmarks. For purposes of a better engineering each entity of the ER diagram is marked with an identifier (ID) attribute. These IDs are meant for "internal" usage of the *DSS4MiddlewarePBenchmarking*and, thus, they are not further explained.

The *system* refers to the type of system that the user is interested in retrieving the data for. The system contains a type which refers to the type of the benchmarked system (i. e., Java Server, Workflow Management System, etc.). The various *benchmarks* focus on addressing the performance of different types of systems. Each benchmark is characterised by its name, the consortium that has proposed it, and if it is widely accepted and adopted as a standard benchmark. Furthermore, based on their compositions, benchmarks can be categorized into types: synthetic benchmark, application benchmark, micro/toy benchmarks etc. Each system has a set of factors or components whose performance affect the performance of the overall system. As for example, in the SPEC ® JMS 2007benchmark recognizes as performance factors the hardware configuration, the JMS server software, the Java Virtual Machine software and the network performance.

The individual systems that are deployed and benchmarked are refered to as *System Under Test (SUT)*. The benchmark related attributes of a SUT are the hardware configuration of its comprising artifacts, the enteprise name of the product (i. e., Camunda[3], Apache Active MQ[4], etc.), as well as its software license, and the version of the benchmarked system. In this work, the hardware configuration has been assumed as unstructured data described in free text. However, in future work, the harware configuration may constitute an entity by itself describing in detail the number of servers used, if the SUT was deployed on virtual or physical machines, the technical details of the machines and other configuration related information. It is widely accepted that a representative benchmark should follow hardware configurations similar to the consumer environment [7]. Therefore, the *DSS4MiddlewarePBenchmarking*should indicate the minimum necessary hardware requirements of the SUTs.

The *application scenario* in a benchmark should cover similar target audience. Consequently, the decision on what application scenario to propose is closely

[3] https://camunda.org
[4] http://activemq.apache.org

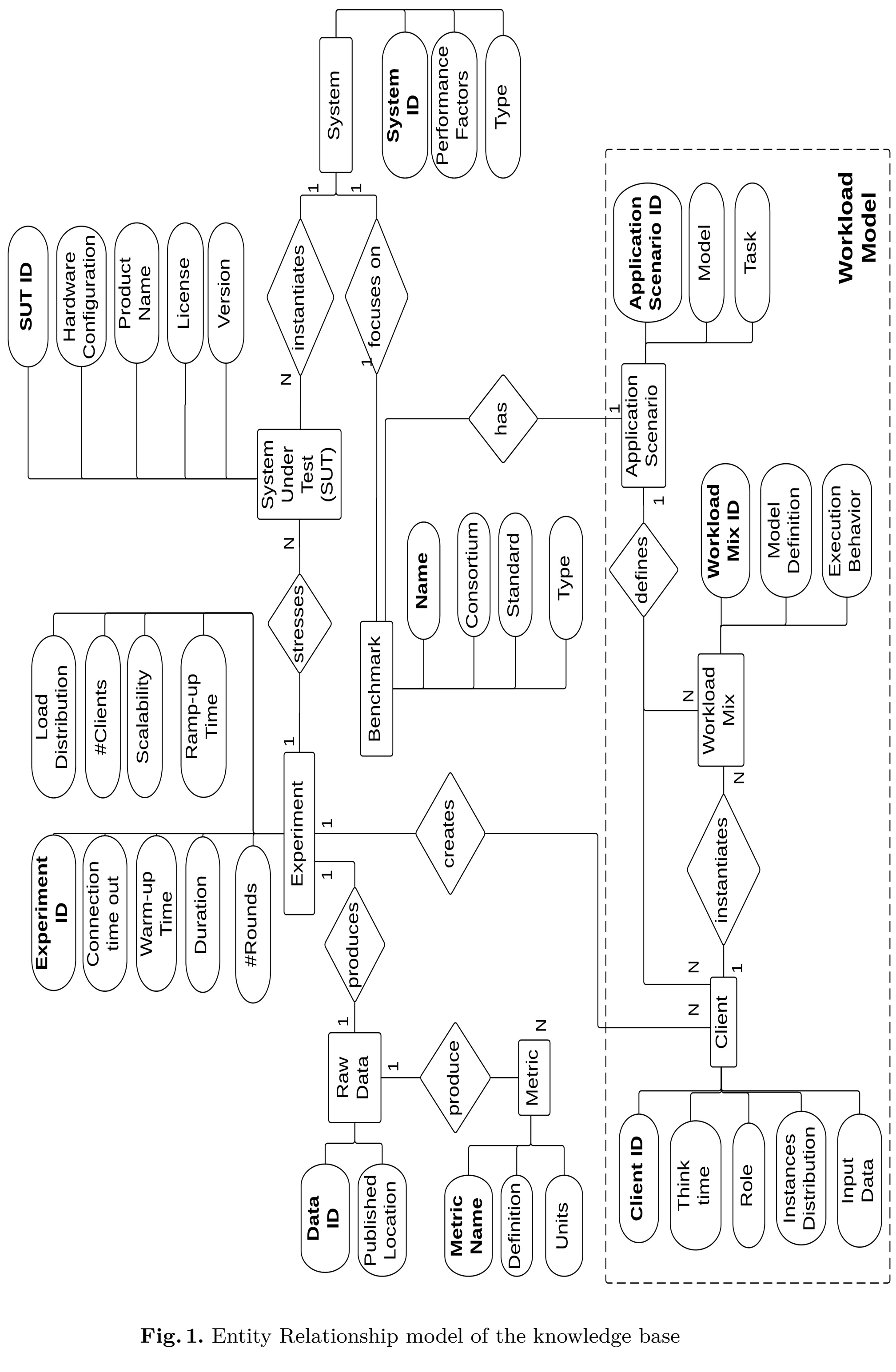


Fig. 1. Entity Relationship model of the knowledge base

related to the type of benchmark that the user would like to construct or apply. It is important to assure that the benchmark can be suitable for a large number of users [7]. Every benchmark covers an application scenario that provides conceptual frameworks for a specific area containing underlying components that are well-known for applications of this kind [10]. Furthermore, the application scenario should be chosen in a way where different subsets of the functionality offered by the underlying technology are stressed [18]. All conducted benchmarks cover a specific business model along with defined tasks that are being fulfilled. Each application scenario defines different users, which with respect to different roles, focus on various tasks. The roles defined in the application scenario could be grouped according to the nature of their tasks. We grouped these tasks in two groups, namely admin and regular user. Most of the conducted application scenarios had a clear distinction between administrative tasks and normal operations, therefore, this solution was considered most suitable. For instance, the SPEC ® JMS 2007benchmark four different participants were involved in the application scenario [22]. The Company Headquaters responsible for the accounting of the company can be classified as the admin of the scenario. The Distribution Centers, Supermarkets and Suppliers are involved in tasks related to their capable functionalities, therefore, these were grouped as (regular) users.

As discussed, each application scenario involves different roles that have a focus on different tasks. The roles defined in the application scenario could be grouped according to the nature of their tasks. During the benchmark the roles are instantiated and executed by the participating *clients*. The clients are responsible to periodically initiate various instances of the *workload mix*, i. e., the data to issue to the SUT in order to execute the performance tests. The time for which the client waits before instantiating the next instance is defined as the think time of the client. Each client might initiate different types of instances for a workload mix. This is defined by the instances distribution attribute. In order to start an instance of the workload mix the client might also provide initiating input data. Each instance of the workload mix is connected to its definition. The definition, for example, might be a reference model of the workflows or the definition of a database query. During its execution the workload mix will follow a specific behavior. For example, with what percentage a condition statement will evaluate to true. The application scenario, the workload mix and the client are grouped together to form the *workload model* [5]. Namely, the workload model can be described as the group of components that are needed for stressing the system.

The execution of the workload model is highly dependent and connected to the *experiments*. The experiments can be basically considered as the orchestrators of the overall benchmark methodology. The experiments create clients that in turn instantiate the workload mix. The clients are created with respect to a load distribution function which can be normal, bursting, etc. At the end, the experiment needs to store how many clients were instantiated for the produced results. The experiments also contain information on the scalability as for some of the benchmarks the ability to scale the workload was provided. For instance, in the SPEC ® JMS 2007 [22] benchmark, a natural way to scale the workload was provided. To be precise, two different types of scaling were provided, the horizontal approach supported scaling in terms of increasing the number of supermarkets, while in the vertical approach the number of products sold in each supermarket was increased. If the scalability is defined, then the instantiation of clients and workload mix need to be adjusted accordingly. Each experiment is repeated for a predefined number of rounds and lasts for a specific time (duration). In case the system will not respond the experiment will time out after a predefined amount of time (time-out duration). The warm up time occurs at the beggining of the experiment and refers to the time that the system needs for initialization before reaching a stable state. Likewise, ramp up time is called the time that the experiment driver need to reach the maximum level of workload. The durations of the warm up and the ramp up times are excluded from the measuremens of the experiment. The product of an experiment are the *raw data*. As suggested by research guidelines the raw data should be published online in order to foster reproducibility. Thus, the raw data are linked to their location of publication.

Finally, the raw data are analyzed in order to derive meaningful metrics. One of the characteristics of a good benchmark is the usage of meaningful and understandable *metrics* [7], thus the selection of the metrics plays an important role in the design of the benchmark. For instance, the metric used for the SPECjbb ® 2015benchmark is defined as business operations per second [7] and measures SPECjbb bops units. Another example is the throughtput metric of the TPC-C benchmark defined as transactions per minute and measured in the tpm unit.

3.3 Model of Dependencies

The complexity of designing a new benchmark is caused by the high dependencies between its participating artifacts. Figure 2 shows the dependencies between the recognized artefacts of a benchmark. The system is the most affecting factor of the benchmark's design. As seen in Fig. 2 it affectrs the definitions of the performance factors, as well as the design of the application scenario, the system under test, the experiment and the workload mix. The performance factors are derived directly from the system, as the participating components of the systems are these having an impact on the performance. The application scenario is also strongly dependent on the system as it has to be compliant with a representative use case of the system's usage. The system plays also a big role in the definition of the metrics to be computed, as the computed metrics should be interesting, relevant and representative of the system's performance. Finally, the systems under test have to be compliant with the defined system type. For example, in the case of WfMSs, the systems under test should be compliant with either the BPEL or the BPMN 2.0 process modeling language, in order to derive consistent results and be able to apply a fair comparison between the SUT. Lastly, the design of the experiments is also dependent to the SUT, as the benchmarking infrastructure that runs the experiments, the instructed load and data are also strongly bound to the it. The workload mix is also affected by the system and

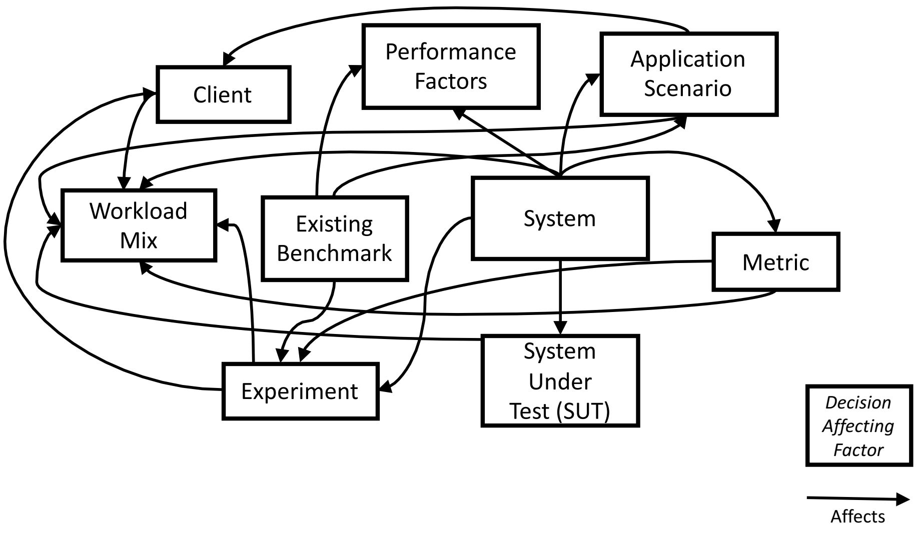
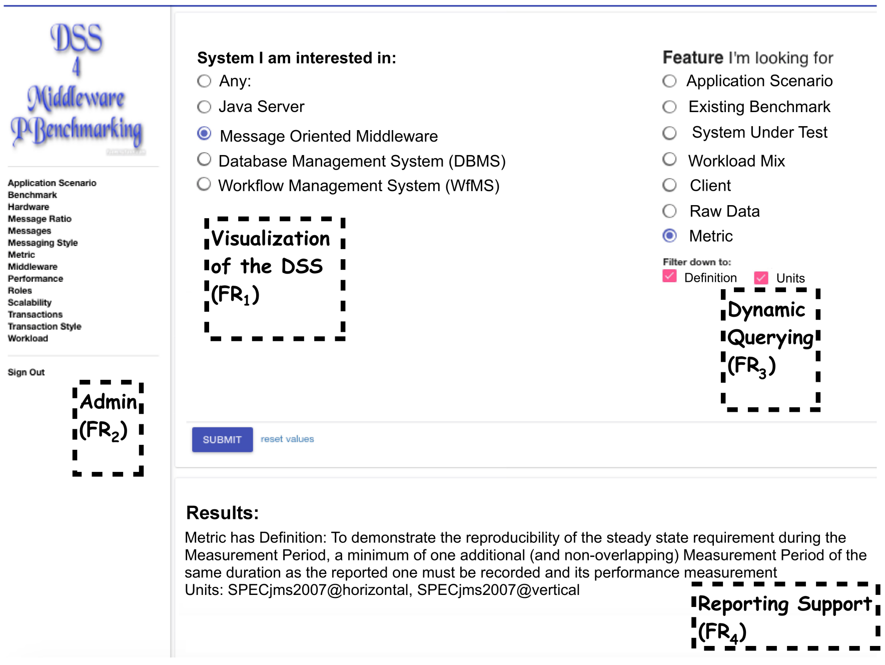


Fig. 2. Conceptual Model of Dependencies in Benchmarks Design

the system under test, as its definition, design and behavior should be compliant to the system's type.

The application scenario of the benchmark affects the clients, as they are the executors of the application scenario. Likewise, the design of the workload mix is also affected by the application scenario as its execution from the clients is basically completing the use cases defined in it. The design of the application scenario is also dependent on the existing benchmarks, as historical data stemming from applied practices can provide information on best practices or detected pitfalls. Likewise, the design of the experiment and the recognition of the performance factors are also affected by the information derived from the already applied benchmarks. For instance, it is likely that a benchmark applied on a system for the very first time will not recognize all the performance affecting factors in its initial design, or might design experiments that are not considering all the possible pitfalls for reliable measurements.

As expected, the experiments are also directly affecting the design of the workload mix and the clients. This is because the application of the experiments is driven by the execution of the clients and the instances of the workload mix. As already discussed, the clients are providing the input data to initialize instances of the workload mix. Consequently, the design of the workload mix is also affected by the client's design. The design of the experiments is affected by the design of the metric. The experiment should follow the design of the metrics and target to performance tests that will produce meaningful raw data out of which we will derive the performance metrics. Consequently, the metric's design will also affect



System I am interested in:
- Any:
- Java Server
- [✓ Message Oriented Middleware]
- Database Management System (DBMS)
- Workflow Management System (WfMS)

Feature I'm looking for
- Application Scenario
- Existing Benchmark
- System Under Test
- Workload Mix
- Client
- Raw Data
- [✓ Metric]

Filter down to: [✓ Definition] [✓ Units]

Results:

Metric has Definition: To demonstrate the reproducibility of the steady state requirement during the Measurement Period, a minimum of one additional (and non-overlapping) Measurement Period of the same duration as the reported one must be recorded and its performance measurement
Units: SPECjms2007@horizontal, SPECjms2007@vertical

Fig. 3. Functional requirements for the user interface

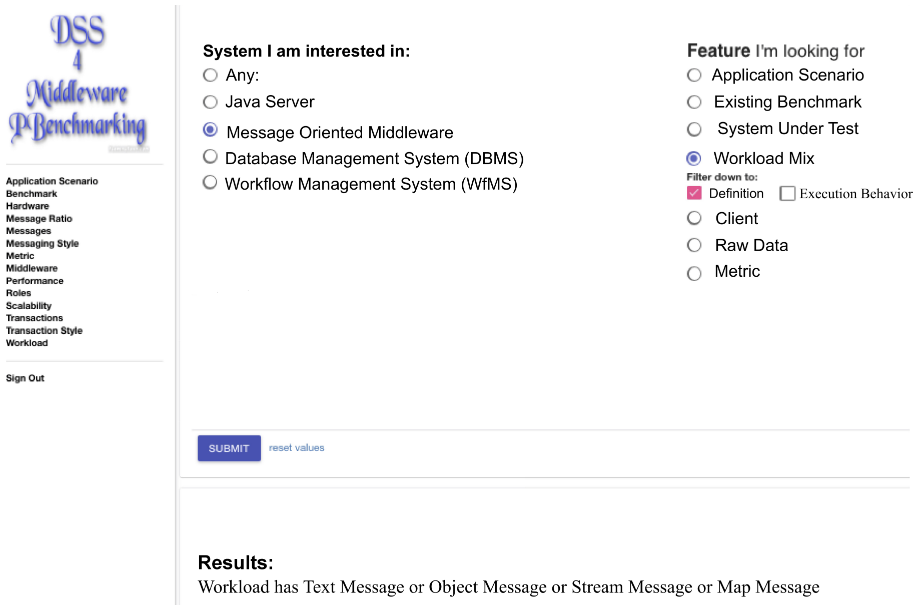
the design of the workload mix in order to be representative and drive to relevant raw data.

In general, we observed that in the design of the benchmark most of the artifacts are dependent on at least one of the rest. The system artifact is the one that mostly affects the design. This is reasonable as the system artifact is the goal of the benchmark. Therefore, the benchmark's design is centered on it. On the other side, the workload mix seems to be the artifact whose design is affected more by the surrounding components. Again this is reasonable, as the workload mix plays an important role to the benchmark's execution and its design affects heavily the derived results and quality of the benchmark.

4 Implementation

In order to realize the *DSS4MiddlewarePBenchmarking*system we have utilized the emerging javascript technologies MongoDB[5], Express.js[6], AngularJS[7], and Node.js[8]. This set of technologies are combined together to the open-source

[5] https://www.mongodb.com
[6] http://expressjs.com
[7] https://angularjs.org
[8] https://nodejs.org/en/



Application Scenario
Benchmark
Hardware
Message Ratio
Messages
Messaging Style
Metric
Middleware
Performance
Roles
Scalability
Transactions
Transaction Style
Workload

Sign Out

System I am interested in:

- Any:
- Java Server
- [✓ Message Oriented Middleware]
- Database Management System (DBMS)
- Workflow Management System (WfMS)

Feature I'm looking for

- Application Scenario
- Existing Benchmark
- System Under Test
- [✓ Workload Mix]
  - Filter down to: [✓ Definition] Execution Behavior
- Client
- Raw Data
- Metric

Results:
Workload has Text Message or Object Message or Stream Message or Map Message

Fig. 4. Validation of use case 1

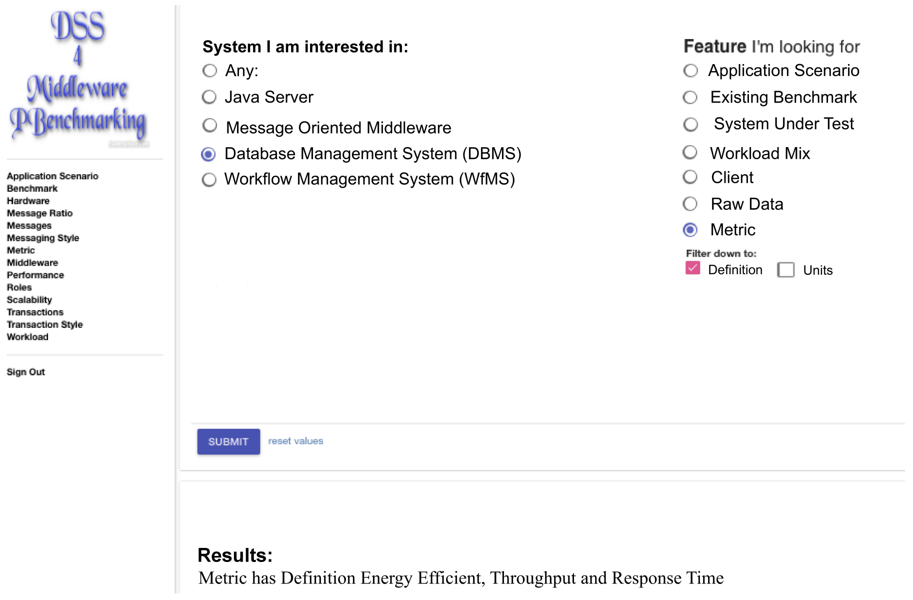
MEAN [9] technology stack to enable the development of dynamic web applications. With the usage of this technology stack, we were able to develop back-end services along with a web-based user interface in the front-end. The functional requirements have been implemented through REST APIs that interract with the back-end and deliver the results to the end user through a user-friendly interface. Currently, the reasoning and responses of the *DSS4MiddlewarePBenchmarking*are implemented through querying the knowledge base with respect to the user defined parameters. In other words, the *DSS4MiddlewarePBenchmarking*can be currently seen as a *document driven* DSS.

5 Validation

In this section, we validate our prototypical implementation of the *DSS4MiddlewarePBenchmarking*. The *DSS4MiddlewarePBenchmarking*is available as open source[10]. The implementation of the functional requirements $FR_1 - FR_4$ as they are recognized in Sect. 3.1 is shown with labels on Fig. 3.

FR_2 is satisfied through the administation panel, which enables the easy editing and expansion of the conceptual model through the graphical interface. The validation of FR_5 to provide related and reliable answers to the user is

[9] http://mean.io
[10] https://bitbucket.org/tayyabaazad/dss4middlewarepbenchmarking/



System I am interested in:
Any:
Java Server
Message Oriented Middleware
[✓ Database Management System (DBMS)]
Workflow Management System (WfMS)

Feature I'm looking for
Application Scenario
Existing Benchmark
System Under Test
Workload Mix
Client
Raw Data
[✓ Metric]
Filter down to:
[✓ Definition] Units

Results:
Metric has Definition Energy Efficient, Throughput and Response Time

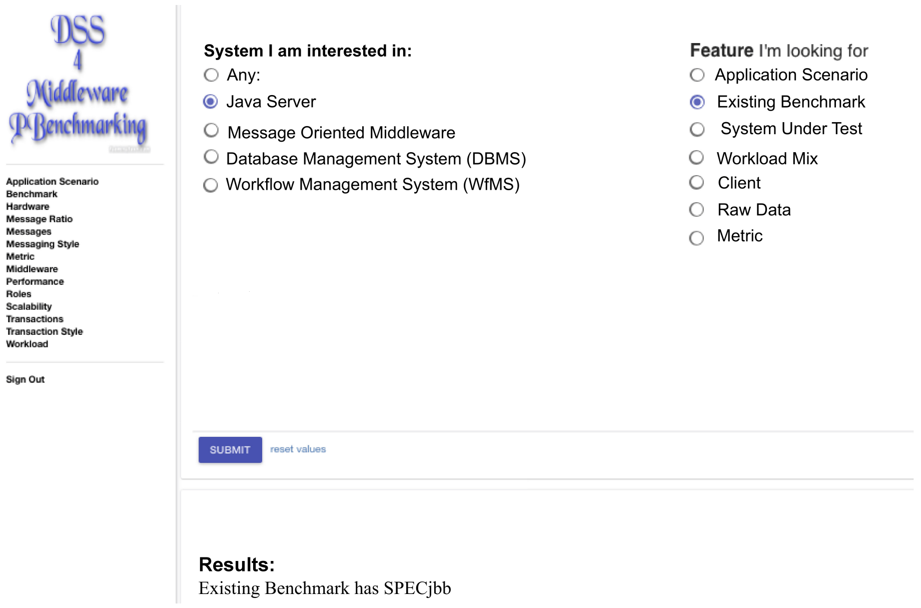
Fig. 5. Validation of use case 2

provided through three representative use cases. In the following, we present four identified use cases along with their responses.

Use case 1: What are the data types that the workload messages of my MOM-based system can have?
The query specification and result of the use case 1 are shown in Fig. 4. In this case the user chooses the *MOM* as the system to focus and specifies the feature that s/he is looking for as *workload*. This will narrow down the choice of workload to filter down to *model definition* and eliminate the returned replies. The response is that message types are specified to be as text message, or object message, or stream message, or map message.

Use case 2: What data types and metrics of the workload have to be considered when benchmarking a DBMS-based system?
In this case, the user selects the Database Management System (DBMS) as the system to target and then specifies the *metric* and filters down to its *definition*. Finally, the response is *energy efficient*, *throughput* and *response time* as defined metrics.

Use case 3: What are the existing benchmarks to study when benchmarking a Java-server based middleware system?
In this case, the user selects *Java Server* as system to focus and *existing benchmark* as the feature of interest. Here, no filters are applied and the resulting answer is directing to the SPECjbb ® 2015 [23] benchmark.



Application Scenario
Benchmark
Hardware
Message Ratio
Messages
Messaging Style
Metric
Middleware
Performance
Roles
Scalability
Transactions
Transaction Style
Workload

Sign Out

System I am interested in:
- Any:
- [✓ Java Server]
- Message Oriented Middleware
- Database Management System (DBMS)
- Workflow Management System (WfMS)

Feature I'm looking for
- Application Scenario
- [✓ Existing Benchmark]
- System Under Test
- Workload Mix
- Client
- Raw Data
- Metric

SUBMIT reset values

Results:
Existing Benchmark has SPECjbb

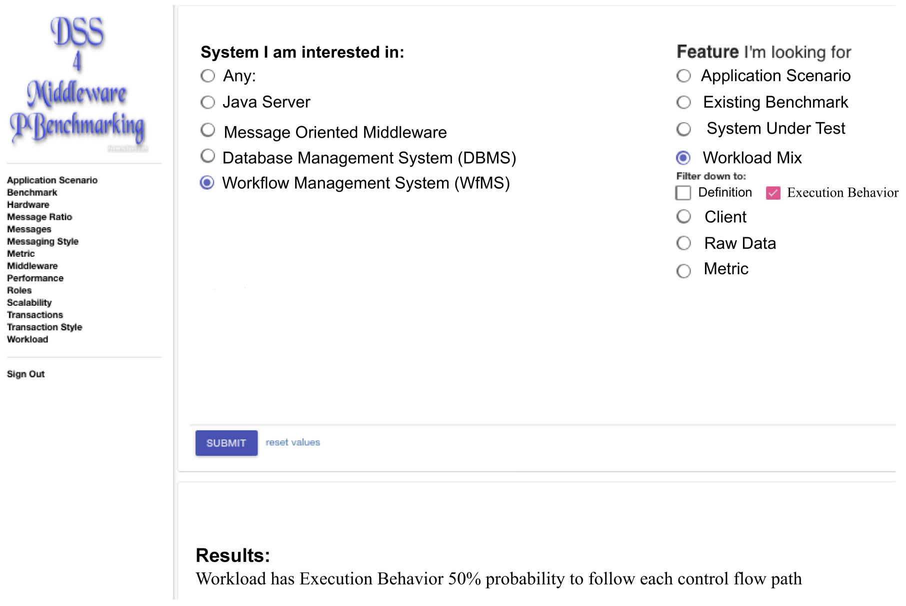
Fig. 6. Validation of use case 3

Use case 4: What is the execution behavior of a workflow instance during a benchmark?
In this case, the user selects *Workflow Management System* as the system of focus and *workload mix* as the feature of interest and filters down the workload mix options to *execution behavior*. The resulting answer indicates that previous works have defined the execution behavior as "50% probability to follow each control-flow path".

6 Related Work

With the evergrowing ratio of information and systems complexity the importance of the DSS is increasing [16]. With a focus on cloud applications, Zimmermann et al. [29] present a tool to support the decision making on architectural design by providing features supporting rapid problem space modeling, UML model linkage, question-option-criteria diagram support, meta-information for model tailoring, as well as decision backlog management. For supporting the decision making of the migration of the application database layer to the cloud, Strauch et al. [24] present a DSS that implements the proposed methodology and supports the user in this very complex decision making. Likewise, Andrikopoulos et al. [2] propose a DSS for the application migration to the cloud.

One example for a web-based DSS implementation is implemented by Ngai and Wat [13], where a risk analysis for the e-commerce sector is provided. Remko



System I am interested in:
- ○ Any:
- ○ Java Server
- ○ Message Oriented Middleware
- ○ Database Management System (DBMS)
- [✓ Workflow Management System (WfMS)]

Feature I'm looking for
- ○ Application Scenario
- ○ Existing Benchmark
- ○ System Under Test
- [✓ Workload Mix]
  - Filter down to:
  - ☐ Definition [✓ Execution Behavior]
- ○ Client
- ○ Raw Data
- ○ Metric

Results:
Workload has Execution Behavior 50% probability to follow each control flow path

Fig. 7. Validation of use case 4

et al. [15] designed a web-based DSS that guides patients to make suitable decisions in case of low back pain. To the extend of our knowledge this work implements the first DSS in the area of middleware benchmarking. Our work follows guidelines and recommendations in regards to building a DSS pointed out by Bhargava et al. [16]. The proposed DSS can be classified as a knowledge-driven approach where the decision-making is based on the defined information stored in the database.

7 Conclusion and Future Work

The application of benchmarks in a specific sector is a vital approach for continuous improvement regarding the effectiveness of the systems. However, the right selection of the required benchmark or decision making when developing a new benchmark comes with the need of extensive research and crucial design decisions. In this work, we provided a DSS to assist benchmark users and developers towards choosing the suitable benchmark or defining key points when building a benchmark for WfMSs. The provided DSS is offered in the form of a Document Driven DSS were the data were retrieved and manipulated as unstructured information. The reasoning of our DSS is executed by the knowledge base through user defined queries. On this we developed a prototypical implementation of the DSS and validated it with respect to four use case scenarios.

In future work, we will consider the expansion of the current conceptual model into a more comprehensive conceptual model. Furthermore, we plan to improve the data visualization by leveraging the visualization of the output. Lastly we will extend the current solution to a knowledge driven DSS by combinining the entity relationship and dependencies diagrams to a Bayesian network and applying rules for calculating the responses. Then we will evaluate it with respect to its performance (response times) and accuracy of the responses.

Acknowledgments This work is funded by the "BenchFlow" DACH project (200021E-145062/1) and by the "SmartOrchestra" the BMWi project (01MD16001F).

References

1. Active Endpoints Inc.: Assessing ActiveVOS performance (2011), http://www.activevos.com/content/developers/technical_notes/assessing_activevos_performance.pdf
2. Andrikopoulos, V., Darsow, A., Karastoyanova, D., Leymann, F.: CloudDSF – The Cloud Decision Support Framework for Application Migration. In: Service-Oriented and Cloud Computing, pp. 1–16. Springer Science + Business Media (2014)
3. Bianculli, D., Binder, W., Drago, M.L.: Automated Performance Assessment for Service-oriented Middleware: A Case Study on BPEL Engines. In: Proc. of the 19th International World Wide Web Conference. pp. 141–150. WWW '10 (2010)
4. Din, G., Eckert, K.P., Schieferdecker, I.: A workload model for benchmarking BPEL engines. In: Proc. of the IEEE International Conference on Software Testing Verification and Validation Workshop. pp. 356–360. ICSTW '08, IEEE (2008)
5. Ferme, V., Ivanchikj, A., Pautasso, C., Skouradaki, M., Leymann, F.: A Container-centric Methodology for Benchmarking Workflow Management Systems. In: Proceedings of the 6th International Conference on Cloud Computing and Service Science. SciTePress (2016)
6. Hackmann, G., Haitjema, M., Gill, C., Roman, G.C.: Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices. In: Proc. of the 4th International Conference of Service Oriented Computing. pp. 503–508. ICSOC '06, Springer (2006)
7. Huppler, K.: The Art of Building a Good Benchmark, chap. Performance Evaluation and Benchmarking, pp. 18–30. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
8. Intel, Cape Clear: BPEL Scalability and Performance Testing. White paper (2007)
9. Jordan, D., Evdemon, J.: Business Process Model And Notation (BPMN) Version 2.0. Object Management Group, Inc (2011)
10. Menasce, D.A., Almeida, V.: Capacity Planning for Web Services: Metrics, Models, and Methods. Prentice Hall, 1st edn. (2001)
11. Moscato, D.R.: Performance assurance for IT. Benchmarking: An International Journal 12(3), 283–284 (2005)
12. Muehlen, M., Recker, J.: How Much Language Is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. In: Bellahsène, Z., Léonard, M. (eds.) Advanced Information Systems Engineering, Lecture Notes in Computer Science, vol. 5074, pp. 465–479. Springer Berlin Heidelberg (2008)
13. Ngai, E., Wat, F.: Fuzzy decision support system for risk analysis in e-commerce development. Decision Support Systems 40(2), 235–255 (Aug 2005)

14. OASIS: Web Services Business Process Execution Language Version 2.0 – OASIS Standard (2007)
15. Peiris, D., Williams, C., Holbrook, R., Lindner, R., Reeve, J., Das, A., Maher, C.: A Web-Based Clinical Decision Support Tool for Primary Health Care Management of Back Pain: Development and Mixed Methods Evaluation. JMIR Research Protocols 3(2) (Apr 2014)
16. Power, D.J.: Decision support systems: A historical overview. In: Handbook on Decision Support Systems 1, pp. 121–140. Springer Science + Business Media (2008)
17. Roller, D.H.: Throughput Improvements for BPEL Engines: Implementation Techniques and Measurements applied in SWoM. Ph.D. thesis, University of Stuttgart (2013)
18. Sachs, K., Kounev, S., Carter, M., Buchmann, A.: Designing a Workload Scenario for Benchmarking Message-Oriented Middleware. In: Proceedings of the 2007 SPEC Benchmark Workshop. SPEC (2007)
19. Skouradaki, M., Ferme, V., Pautasso, C., Leymann, F., van Hoorn, A.: Micro-Benchmarking BPMN 2.0 Workflow Management Systems with Workflow Patterns. In: Proc. of the 28th International Conference on Advanced Information Systems Engineering. CAiSE '16, Springer (2016)
20. Skouradaki, M., Roller, D.H., Leymann, F., Ferme, V., Pautasso, C.: On the Road to Benchmarking BPMN 2.0 Workflow Engines. In: Proc. of the 6th ACM/SPEC International Conference on Performance Engineering. pp. 301–304. ICPE '15 (2015)
21. Standard Performance Evaluation Corporation: SPEC (1995), https://www.spec.org/spec/
22. Standard Performance Evaluation Corporation: SPEC JMS 2007 (2007), https://www.spec.org/jms2007/
23. Standard Performance Evaluation Corporation: SPECjbb2015 (2015), https://www.spec.org/jbb2015/
24. Strauch, S., Andrikopoulos, V., Bachmann, T., Karastoyanova, D., Passow, S., Vukojevic-Haupt, K.: Decision Support for the Migration of the Application Database Layer to the Cloud. In: Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science. CloudCOM'13, Institute of Electrical & Electronics Engineers (IEEE) (Dec 2013)
25. Transaction Processing Performance Council: TPC (1992), http://www.tpc.org/information/benchmarks.asp
26. Transaction Processing Performance Council: TPC-C (1992), http://www.tpc.org/tpcc/
27. Transaction Processing Performance Council: TPC-E (1992), http://www.tpc.org/tpce/
28. Vieira, M., Madeira, H., Sachs, K., Kounev, S.: Resilience benchmarking. In: Resilience Assessment and Evaluation of Computing Systems, pp. 283–301. Springer Science + Business Media (2012)
29. Zimmermann, O., Wegmann, L., Koziolek, H., Goldschmidt, T.: Architectural decision guidance across projects - problem space modeling, decision backlog management and cloud computing knowledge. In: Proc. of the 15th Working IEEE/FIP Conference on Software Architecture. WISCA '15, IEEE Computer Soc.P., U.S. (2015)

All links were last followed on July 1, 2016.