

# **IBM Research Report**

## **Proceedings of the 10<sup>th</sup> Advanced Summer School on Service Oriented Computing**

**Johanna Barzen<sup>1</sup>, Rania Khalaf<sup>2</sup>,  
Frank Leymann<sup>1</sup>, Bernhard Mitschang<sup>1</sup>, Editors**

<sup>1</sup>University of Stuttgart  
Germany

<sup>2</sup>IBM Research Division  
One Rogers Street  
Cambridge, MA 02142-1203  
USA



Research Division

Almaden – Austin – Beijing – Brazil – Cambridge – Dublin – Haifa – India – Kenya – Melbourne – T.J. Watson – Tokyo – Zurich

# **The 10<sup>th</sup> Advanced Summer School on Service-Oriented Computing**

June 27 – July 1

2016

Hersonissos, Crete, Greece

The 10<sup>th</sup> Advanced Summer School on Service Oriented Computing (SummerSOC'16) continued a successful series of summer schools that started in 2007, regularly attracting world-class experts in Service Oriented Computing to present state-of-the-art research during a week-long program organized in several thematic tracks: patterns, formal methods for SOC, computing in the clouds, data science, e-Health and emerging topics. The advanced summer school is regularly attended by top researchers from academia and industry as well as by graduate students from programs with international acclaim, such as the Erasmus Mundus International Master in Service Engineering.

During the morning sessions at SummerSOC renowned researchers gave invited tutorials on subjects from the themes mentioned above. The afternoon sessions were dedicated to original research contributions in these areas: these contributions have been submitted in advance as papers that had been peer-reviewed. Accepted papers were presented during SummerSOC and during the poster session. Furthermore, PhD students had been invited based in prior submitted and reviewed extended abstracts to present the progress on their theses and to discuss it during poster sessions. Some of these posters have been invited to be extended as a full paper, which are included in this Technical Report.

Also, this year the “Christos Nikolaou Memorial Ph.D. Award” to honor Prof. Christos Nikolaou’s career-long contributions in university education and research was established. The first winner of this Christos Nikolaou Memorial Ph.D. Award is Jörg Lenhard, who presents the core ideas of his awarded thesis in this Technical Report too. The award is not only an honor and distinction but is associated with 2000€ for the awardee, sponsored by StartTech Ventures.

Johanna Barzen, Rania Khalaf, Frank Leymann, Bernhard Mitschang  
- Editors -

## **Content**

### **Winner of the Christos Nikolaou Memorial Ph.D. Award:**

**On the Suitability of Process Model Similarity Metrics for Evaluating Replaceability ..... 1**  
J. Lenhard

### **Poster Session: Papers**

**Towards Function and Data Shipping in Manufacturing Environments: How Cloud Technologies leverage the 4th Industrial Revolution ..... 16**  
M. Falkenthal, U. Breitenbücher, M. Christ, C. Endres, A. Kempa-Liehr, F. Leymann, and M. Zimmermann

**Flexible Execution and Modeling of Data Processing and Integration Flows ..... 26**  
P. Hirmer

**A Decision Support System for the Performance Benchmarking of Workflow Management Systems..... 41**  
M. Skouradaki, T. Azad, U. Breitenbücher, O. Kopp, and F. Leymann

### **Poster Session: Extended Abstract**

**Adaptable Digital Enterprise Architecture with Microservices ..... 59**  
J. Bogner and A. Zimmermann

# On the Suitability of Process Model Similarity Metrics for Evaluating Replaceability

Jörg Lenhard

Department of Mathematics and Computer Science,  
Karlstad University, 65188 Karlstad, Sweden  
`joerg.lenhard@kau.se`

**Abstract.** In the field of process-aware information systems, much work has been devoted to developing metrics for determining the similarity between process models. Similarity assessment is important for a wide array of applications and one area that has received relatively little attention so far is the replaceability assessment of executable process models. Replaceability assessment is relevant during software migration, e.g., when upgrading to a new execution platform. In this setting, it might be necessary to replace process models that can no longer be executed on the newer platform. Many of the existing metrics are ill-suited for replaceability assessment, because they were developed for non-executable models and tend to abstract from details that are decisive for the aforementioned application scenario. This paper discusses existing metrics for similarity assessment in a literature review and selects a subset of the body of metrics as candidates for replaceability assessment. Using an exemplary computation, it recommends a particular metric, TAR-similarity, for this purpose. Through this evaluation, this paper can be seen as a motivation for developing better node mapping functions.

**Keywords:** replaceability, similarity, software metrics, process model

## 1 Introduction

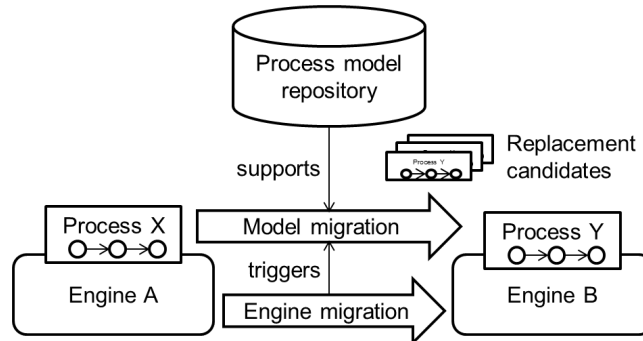
In the software industry, changes take place at an unprecedented pace. Market pressure forces enterprises to constantly revise and update their IT-systems to maintain competitive advantages and to react to customer demands [4]. The necessity to integrate new technologies or features, whilst keeping existing functionality and coping with increasing load, leads to a need for continuous evolution of system structure with respect to software and hardware [24].

In the area of distributed systems, the trends of service-orientation [27] and process-awareness [8] have emerged to cope with such challenges. Computing systems are built as loosely coupled sets of services, with a trend towards microservices [26], which interact by exchanging messages to provide higher-level functionality. This approach eases system evolution, since, in a well-designed system, singular services can be changed or upgraded without impact on the overall system. Message exchanges between services are oftentimes structured



by capturing them in explicit process representations or *process models* [8]. During the implementation phase of the system [8, p. 12], executable models can be deployed and instantiated on a particular type of middleware, called *process engines*. In this setting, an executable process model corresponds to a piece of application software, whereas the process engine represents the execution environment. Process engines have considerable influence on the runtime quality of service provided by the hosted applications. Consequently, there is value in selecting the best performing engine and migrating to newer engines, as evidenced by approaches on engine benchmarking [11, 12], engine selection [13], or the advent of cloud-based engines with improved performance characteristics [15].

Process engines and models interface via language standards, such as the *Business Process Model and Notation* (BPMN) [17]. A major motivation for standardization is the portability of process models among engines enabled by standards. If a model and a set of engines conform to a standard, the process model can be freely migrated between the engines. In theory, this defends from vendor lock-in and potentially eases system evolution. However, work on standard conformance of engines [12, 14] demonstrates that standard support in engines is very diverse in practice. Despite the existence of standards, engines tend to implement differing subsets of the specified features. Thus, it might not necessarily be feasible to migrate an executable process model to a newer engine, even if it is implemented in a standardized notation. In this case, a straight-forward mode of action, described by software quality standards [16], is the replacement of an application by an alternative one. In other words, if a process model cannot be migrated, it could be considered to replace the existing model with a different model that runs on the newer engine. This replacement scenario is the motivation for this paper.



**Fig. 1.** Migration Scenario Leading to Process Model Replacement

Clearly, a replacement is only possible if alternatives to the original model do exist and they are sufficiently *replaceable* with each other. An eventual source of replacement candidates are process model repositories [35]. Management of such repositories is a common task for enterprises that adopt process-aware and

service-oriented technologies. If replacement candidates are available in such a repository, it is possible to compute the *replaceability* [16] for all paired combinations of the enacted model and its alternatives to determine the best fitting replacement candidate. This migration scenario is visualized in Fig. 1. The migration from *Engine A* to *Engine B* triggers the necessity to migrate *Process X*, which unfortunately cannot be modified to enable execution on *Engine B*. Therefore, it needs to be replaced and this is supported by a *process model repository* that supplies a number of *replacement candidates*, which can be modified to run on *Engine B*. From these candidates, *Process Y* is found to be most replaceable. Therefore *Process X* is replaced by *Process Y* during the migration.

The property of central concern here is denoted as *replaceability* by software quality models [16]. In essence, it translates to *similarity* [38]. If two process models are very similar to each other, they can replace one another with relative ease. This implies that metrics for evaluating process model similarity seem suited for evaluating replaceability and opens up a large space of related work. Many similarity metrics have been proposed in the literature, e.g., [1, 2, 6, 18, 19, 23, 25, 31, 33, 34, 37]. What is more, comparative studies that evaluate these metrics do exist [3]. Thus, the addition of yet another set of replaceability metrics on top of the existing body of metrics is neither desirable, nor likely to form a novel contribution. Instead, in this paper, we provide a discussion of existing similarity metrics for the purpose at hand. We select a subset of metrics based on a categorization from [3] and discuss these metrics more closely. Furthermore, we outline crucial problems that remain for using these metrics to evaluate replaceability, essentially a proper node mapping function, and propose to take node types into account during evaluation. With the help of an exemplary computation, we are, thus, able to identify one metric, TAR-similarity [37], as most suitable for our use case.

The approach applied in this paper has been sketched in [20] and the paper is based on an excerpt of [21, Chap. 7]. We cannot provide the same level of detail here and try to distill key ideas into the format of this paper. For a more detailed elaboration, we refer the interested reader to [21]. The remainder of the paper is structured as follows: We start with a review of existing metrics in Sect. 2, including the categorization from [3] according to their type and intended area of application. Through this categorization, we select a subset of applicable metrics, provide a closer discussion of these metrics, and ultimately choose two metrics for an evaluation. Next, Sect. 3 states deficiencies for replaceability assessment, proposes a way of dealing with them, and evaluates the selected metrics by means of an exemplary computation. This allows to decide on which metric fits best. Finally, the paper concludes with a summary.

## 2 Review of Existing Metrics

Similarity metrics, and, therefore, also replaceability metrics, measure the distance between objects [29, 36]. The smaller the distance between the objects is, the more similar and, thus, the more replaceable they are. In our case, the ob-

jects are executable pieces of software. Based on the definition of a similarity metric given in [3, Sect. 2.4], a replaceability metric can formally be defined as follows:

**Definition: Replaceability Metric**

$$REPL(p^1, p^2) = \frac{1}{1 + dist(p^1, p^2)}, \text{ where} \quad (1)$$

- $dist : Process \times Process \rightarrow \mathbb{R}_0^+$  is a function that computes the distance between two process models.
- $p^1, p^2 \in P$ , where  $P$  is the set of all process models and  $(P, dist)$  forms the *metric space* [36].

The crucial difference between replaceability metrics lies in their definition of the distance function.

Similarity metrics for process models can further be classified depending on the entities which form the basis of the computation. This results in a classification in terms of *labels*, *structure*, or *behavior* [9].

**Label Similarity:** Metrics based on label similarity compute the similarity of process models based on the names, i.e., the labels, assigned to their elements. If the labels of two elements found in the models  $p$  and  $p'$  are identical, these elements are considered to be identical as well. If all elements of  $p$  are also found in  $p'$ , and no more, regardless of the structuring of the process graph, then  $p$  and  $p'$  are considered to be identical and  $REPL(p, p') = 1$ . Examples of metrics for label similarity can be found in [1, 6, 18, 25, 34]. The problem with label similarity is that the labels assigned to process elements are normally written in natural language and, hence, they are seldom identical. For instance, the label of an activity in  $p$  might be “Check Order”, whereas  $p'$  contains an activity labeled “Order Checking”. Though the labels are not identical, their distance could be considered as small. Certain natural language comparison techniques, such as the string edit distance [22], or semantic techniques that utilize a thesaurus, as for instance found in [6, 10], can be used to improve the similarity computation among labels.

**Structural Similarity:** Metrics based on structural similarity compare the structure of the process graphs of two models. The smaller the distance between the structure of the graphs, the more similar they are. The distance between graphs can be measured through the graph edit distance of process models [7]. This distance corresponds to the number of insertions, deletions, and substitutions of process elements that are needed to transform the graph of one process model into another. The higher this number is, the greater is the distance. Structural similarity metrics have for instance been defined in [6, 7, 23].

**Behavioral Similarity:** Metrics based on behavioral similarity focus on the execution behavior of process models [19]. They are often computed based on

the execution traces of process models or the execution dependencies among the activities of two models. First, possible traces or execution dependencies are computed based on the model. Then, these traces or dependencies are compared to the traces or dependencies belonging to another model. The higher the overlap between these sets of traces or dependencies is, the higher is the similarity of the process models. Behavioral similarity metrics can for example be found in [6, 31, 33].

A crucial problem of approaches in this area is the requirement to map the nodes in one process model to one or more nodes in the other model. This is necessary to identify if traces really are similar. To achieve this, approaches for behavioral similarity often make use of approaches for label similarity.

A review of the existing metrics and their classification according to areas of application can be found in [3]. This classification can be used to narrow the set of relevant metrics and to decide which metrics are a potential fit for our use case. In [3], Becker and Laue distinguish seven application areas for similarity metrics: i) The simplification of change in process variants, ii) process merging, iii) facilitation of reuse, iv) management of process model repositories, v) automation of process execution, vi) compliance assurance with normative models, and vii) service discovery. The areas of process execution automation and service discovery are closest to our focus of application. The authors remark that “*automation is usually concerned in SOA applications*” and service discovery is “*closely connected to the goal of automation*” [3, Sect. 4.2]. For these areas of application, the authors recommend behavioral metrics that compute similarity based on the dependencies among the nodes of the process graph in favor of metrics based on label or structural similarity. In particular, these metrics are *dependency graphs* [2] and their improvement in *TAR-similarity* [37], the *string edit distance of sets of traces* [33], *causal behavioral profiles* [31], and *causal footprints* [6]. From this set of metrics, we consider TAR-similarity and causal behavioral profiles to be applicable for a replaceability computation. The reasons for excluding the remaining metrics are explained in the following subsections.

## 2.1 Direct Precedence Relationships Among Activities

The metrics captured by dependency graphs [2], TAR-similarity [37], and the string edit distance of sets of traces [33], are all based on a similar idea: They compute the similarity of process models by considering direct precedence relationships among the activities in a process model.

In [2], the direct precedence relationships among activities correspond to the so-called *dependency graph* of a process model. To determine process model similarity, first, all direct precedence relationships of two models, i.e., their dependencies graphs, are computed. In a second step, these sets of direct precedence relationships are compared and the higher their overlap is, the more similar the two process models are. More precisely, the distance between the dependency graphs is equal to the number of dependencies that are not present in both of the graphs. In the case of [2], direct precedence relationships among activities are

considered, regardless of gateways that might be placed between two activities. This means that conditional branching or parallelism in a process model is not taken into account. This is a clear drawback of the approach.

An extension of dependency graphs that tries to tackle this issue is formed by TAR-similarity [37]. The term TAR stems from the *transition adjacency relation*, which is a special form of a direct precedence relationship. The TAR does not only consider direct control dependencies among activities, but also takes into account the interleaving of activities that are executed in parallel. This means that if a process model uses inclusive (OR) or parallel (AND) gateways, there is a larger amount of dependencies in the dependency graph, which is called the *TAR set* here. Apart from this, TAR-similarity is computed in the same fashion as dependency graphs, i.e., by comparing the amount of shared adjacency relations of the TAR sets,  $TAR_1$  and  $TAR_2$ , of two process models,  $p^1$  and  $p^2$ , to all relations. Based on the definition of the similarity metric in [37], the distance function,  $dist$ , can be defined as follows:

$$dist_{TAR}(p^1, p^2) = \frac{|(TAR_1 \cup TAR_2)|}{|(TAR_1 \cap TAR_2)|} - 1 \quad (2)$$

Although TAR-similarity takes gateways into account, it is still limited to direct adjacency relations. An improvement of TAR-similarity that tries to eliminate the requirement of directness can be found in the projected TAR [28]. This extension tries to relax the restriction of direct dependencies, by first computing a projection of the original process model that eliminates so-called silent steps in the model. TAR-similarity is then computed based on the projected model. The problem with the approach presented in [28] is that it cannot automatically be determined which parts of the process model are considered as silent. Here, human judgment is required. For this reason, we omit the projected TAR from further consideration.

Another metric that is quite similar to dependency graphs and TAR-similarity is the string edit distance of sets of traces [33]. This metric is based on the analysis of execution traces. However, as the traces are computed based on the process graph, the difference between sets of traces and a dependency graph is mainly one of terminology. A more notable difference of this metric lies in the fact that it does not necessarily focus on binary, i.e., direct, relations only. Instead, it also takes larger sets of activity sequences into account, which are called *words of length  $n$* , or  *$n$ -grams*. An  $n$ -gram is a trace of the process model that includes exactly  $n$  subsequent activities. To calculate the string edit distance of sets of traces, all possible  $n$ -grams for a process model have to be computed. Thereafter, the distance of two process models corresponds to the aggregated string edit distance of all  $n$ -grams. Although [33] does not address the computational complexity of this approach, it is clear that the calculation is challenging. As a result, the usage of a high value of  $n$  is not feasible in practice and  $n$ -grams of length two, called bi-grams, are most frequent. In this case, the string edit distance of sets of traces corresponds to TAR-similarity. For this reason, we only consider TAR-similarity further.

## 2.2 Causal Footprints

Causal footprints are a behavioral similarity metric proposed in [6]. A causal footprint of a process model corresponds to the set of its activities and the execution dependencies among them. These execution dependencies are not limited to direct precedence relationships. To obtain a causal footprint, a set of look-back links and a set of look-ahead links is computed for every activity in a process model. The set of look-back links of an activity  $A$  corresponds to the set of all activities that may precede the execution of  $A$ . Similarly, the set of look-ahead links of  $A$  corresponds to the set of all activities that may be executed after  $A$  has finished. The causal footprint of a process model corresponds to all sets of look-back and look-ahead links of the activities in the model. The sets of look-ahead and look-back links are treated as vectors, and the similarity of two process models is computed through the cosine of their vectors.

Based on their observations in [3], Becker and Laue discourage the usage of causal footprints, due to their computational inefficiency. Despite the usage of the reference implementation of causal footprints and a very moderate test set of only eight small process models, they experienced computation times that were more than five times as large as for any other similarity metric. Hence, we do not consider causal footprints any further.

## 2.3 Causal Behavioral Profiles

As the name indicates, causal behavioral profiles [31] refer to a behavioral similarity metric. Like the other metrics, its mechanism of computation bases on relations between activities. These relations are not limited to direct precedence relations, as for the metrics discussed in Sect. 2.1. Instead, the metric considers four categories of behavioral relations between activities. Given two activities,  $A_1$  and  $A_2$ , these relations are: i) Strict order relation ( $A_1$  is always executed before  $A_2$ ), ii) co-occurrence relation (if  $A_1$  is executed in a process instance,  $A_2$  must be executed as well, and vice versa), iii) exclusiveness relation ( $A_1$  and  $A_2$  are never executed in the same process instance), and iv) concurrency relation ( $A_1$  may be executed before  $A_2$ , but also the other way round). The set of all relations among the activities of a process model is its *behavioral profile*.

The similarity of two process models is computed by comparing their behavioral profiles. More precisely, the amount of shared execution relations among activities is compared to the amount of all execution relations. In this sense, causal behavioral profiles are very similar to TAR-similarity. The main difference between the two lies in what kind of relations among activities are considered. Behavioral profiles are necessarily larger than TAR sets, since they include a relation for every pair of activities in a process model. However, there is an additional notable distinction. To be able to compare the execution relations among activities of the two process models, it is necessary to establish which activities in the two models correspond to each other. Only then, it is possible to determine if two behavioral relations are the same. If it is the case that, for a specific activity, there are no corresponding activities in the partner process

model, all behavioral relations that involve this activity are ignored. As [31] puts it: “*Solely activities that are aligned by the correspondence relation are considered*”. This is a significant difference from TAR-similarity, for which such relations are still part of the TAR sets. Furthermore, it implies that causal behavioral profiles heavily depend on a proper correspondence function. Weidlich et al. use a trace equivalence correspondence function based on execution traces and activity labels, but require that the function must be injective. This means it is not applicable if there are activities in the first process model for which no corresponding activity in the second process model can be found.

**Table 1.** Reviewed Similarity Metrics for Replaceability Evaluation

Metric	Ref.	Object in Focus	Issues
Dependency Graphs	[2]	direct precedence relations	gateways are ignored, superseded by TAR-similarity
TAR-similarity	[37]	direct precedence relations	
Edit Distance of Set of Traces	[33]	n-grams	inefficient for larger $n$
Causal Footprints	[6]	look-back and look-ahead links	inefficient computation
Causal Behavioral Profiles	[31]	behavioral relations among activities	

The results of the review are summarized in Tab. 1. The table depicts the metrics that seem suitable for our area of application, according to the categorization specified in [3].

### 3 Metrics Selection

Evaluating the replaceability of executable software is not what the designers of similarity metrics originally had in mind. Due to this, existing similarity metrics share a number of deficiencies for a replaceability evaluation, which we discuss in the following subsection, Sect. 3.1. Based on this discussion, we evaluate the remaining metrics from the previous section using a set of synthetic process models, similar to [3], in Sect. 3.2.

#### 3.1 Deficiencies of Existing Metrics for Replaceability Evaluation

To be applicable in many settings and use cases, all metrics discussed in Sect. 2 are computed on the basis of a formalism, such as Petri nets, or other abstractions of concrete process models. To enable the metrics computation, a process model has to be translated into the formalism. During this translation, language-specific execution semantics of particular language elements, except for control-flow routing constructs, are mostly lost. This is acceptable in general-purpose

application scenarios, in particular those that deal with non-executable and abstract process models anyway. In fact, such scenarios are what most metrics have been designed for. For instance, only three out of the 22 metrics evaluated in [3] are targeted at similarity assessment of executable process models by their authors. This results in several problems when trying to use the metrics for an evaluation of the replaceability of executable software. These problems may render certain metrics unsuitable for this use case.

To understand these issues, it is important to recall the purpose of a replaceability evaluation here: To investigate how well a given executable process model can replace a second model in a given execution environment, because the second one contains language elements that are not supported. As a consequence, the concrete language elements have to be taken into account during the replaceability evaluation. Abstracting from the concrete vocabulary of the language, which is what practically all metrics do, defeats the purpose of the evaluation to begin with. The deficiencies for a replaceability evaluation resulting from this abstraction level can be summarized as follows:

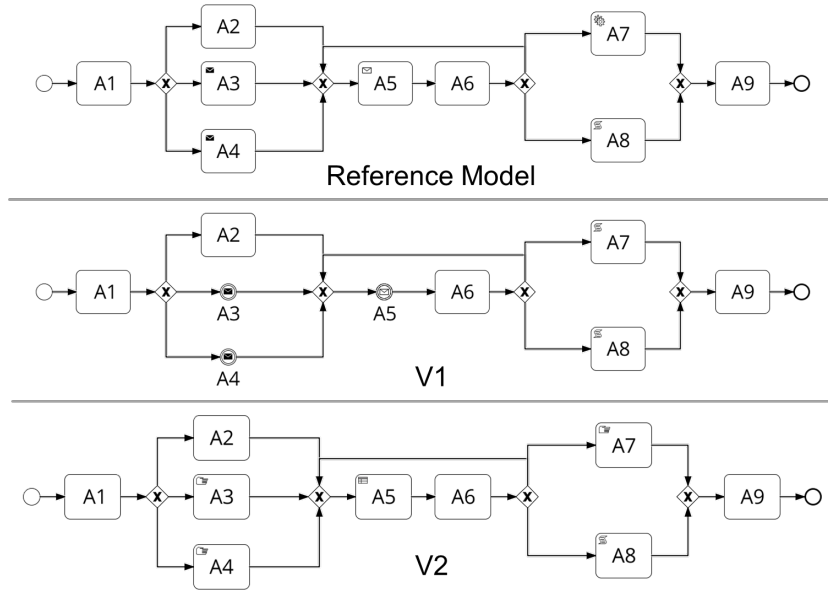
1. *Generality*: In most cases, similarity is computed based on the activities and connectors among them. The vocabulary of a process language may not be limited to these sets of elements. For instance, in the case of BPMN [17], activities and gateways clearly fit to this model. However, it needs to be stated, in what way events fit into the above categories. Moreover, it is seldom made clear how a metric should deal with hierarchical decomposition in a process model.
2. *Node Mapping*: Practically all approaches assume that it is possible to map the nodes, in particular the activities or their execution traces, between two process models, i.e., that it is possible to determine if an activity of process model  $p_1$  *corresponds* to one or more activities of process model  $p_2$ . This is referred to as the *matching problem* in [32]. In the definition of most structural or behavioral metrics, this aspect is left open or, if discussed at all, deferred to approaches that determine the correspondence between activities based on their labels. This underspecification is problematic, due to the impact of the node mapping on the similarity metric. As [3, Sect. 7] puts it, “*the quality of the mapping between nodes [...] has a significant contribution to the quality of a similarity measure*”. This is even more evident when it comes to executable models. Activity labels on their own are of no importance for the execution semantics of activities. As a consequence, relying on activity labels only for finding corresponding activities is a questionable assumption. Instead, the execution semantics of a concrete activity, as for instance captured in its type (e.g., message sending, script execution, business rule execution, etc.), are a decisive factor. When considering a process language, such as BPMN, a variety of node types with different execution semantics exist. None of the approaches explicitly takes these types into account.

All in all, the issue of generality can be resolved with relative simplicity. In the case of BPMN, as done in [3], events can be included in the replaceability



evaluation by considering them as nodes of a process model in the same fashion as activities. Hierarchical decomposition is more difficult to incorporate. It can be achieved by either treating a *SubProcess* as a single node in the same fashion as tasks and events, or by ignoring the decomposition and embedding the contents of a *SubProcess* into the parent process. The latter is the strategy favored by the metrics we consider for evaluation here [31,37].

The issue of constructing a node mapping is more critical. By ignoring node types, and, hence, parts of the execution semantics of the process model, a similarity metric risks to rate process models as similar, although they are dissimilar in reality, or vice versa. This can be demonstrated by considering the similarity of the BPMN process models depicted in Fig. 2<sup>1</sup>. These are a reference model, RM, a first variant of the reference model, V1, and a second variant of the reference model, V2. The structure of the process graph is identical in all cases and so are



**Fig. 2.** Process Models for Replaceability Computation

the labels of the nodes. We use identical labels and structure to eliminate any influence of label or structural similarity and to enable an isolated consideration of the node types. When ignoring node types, the dependency graphs and execution traces for the process models, as well as their causal footprints and behavioral profiles are identical. As a result, all process models are considered as identical

<sup>1</sup> The structure of the models is identical to the structure of the reference model from [3]. We adjusted the types of the activities from ordinary tasks to specific BPMN tasks and introduced events in V1.

to each other with the discussed metrics. This is problematic, since two of the models are less similar, when taking the concrete node types into account. At first glance, it may seem that V2 is more similar to the reference model than V1, since V1 uses events, whereas V2 only uses activities. However, the opposite is true: The execution semantics of V1 are identical to RM, although it uses events instead of activities. In V1, *Send-* (A3, A4) and *ReceiveTasks* (A5) are replaced by *Send-* and *ReceiveEvents*. These process elements have the same execution semantics in BPMN. Furthermore, the *ServiceTask* (A7) in RM is replaced by a *ScriptTask*. A script implemented in a programming language can be used to emulate almost any other task, including a *ServiceTask*. Therefore, V1 and the reference model are in fact fully replaceable. In contrast, in V2, activity A5 is not a *ReceiveTask* that blocks until a message is received from an external party, but a *BusinessRuleTask*, which has very different execution semantics. Thus, it cannot be considered as corresponding to activity A5 of the reference model. The same reasoning applies to activities A3, A4, and A7, which are *ManualTasks* in V2, but *Send-* and *ServiceTasks* in the reference model. Similar to [3], the process models depicted in Fig. 2 can be used for evaluating replaceability metrics. A suitable replaceability metric should consider V2 less fit to replace RM than V1 and RM.

To address this issue, we propose a node mapping approach that takes execution semantics of nodes into account to operationalize the metrics for the replaceability evaluation of executable software. [21, Chap. 6] defines a *set of alternatives* for an element of a language in order to measure the adaptability of process models. This set contains all semantically equivalent alternative process elements that can be used to replace a certain element, as described above. Using this set, nodes with similar execution semantics are considered as *corresponding* nodes in the replaceability computation. This notion can easily be combined with approaches for label similarity. For instance, two nodes can be considered as corresponding if their labels are identical *and* their types correspond to each other. Due to limited space, we refer the interested reader to [21] for a formal definition of the respective node mapping function and a mapping of the language elements of BPMN [17].

### 3.2 Discussion of Metrics Performance

Based on the discussion from the previous section, it is possible to decide between the two remaining metrics, TAR-similarity [37] and causal behavioral profiles [31], through an exemplary computation. As stated in Sect. 3.1, a suitable metric should find V1 to be very similar, if not identical, to the reference model. At the same time, the metric should find V2 to be dissimilar from the reference model. In the following, we discuss the values of these metrics based on the notion of correspondence from the previous section and judge the results, which can be found in Tab. 2. To determine corresponding nodes, we combine label identity and execution semantics.

**TAR-similarity:** In the case of TAR-similarity, the TAR set of the reference model,  $TAR_{RM}$ , resolves to  $\{(A1, A2), (A1, A3), (A1, A4), (A2, A5), (A3,$

**Table 2.** Results of the Metrics Evaluation

Metric	$RM \leftrightarrow V1$	$RM \leftrightarrow V2$
Desired Result	1	< 1
TAR-similarity	1	0.16
Causal Behavioral Profiles	1	1

$A5$ ),  $(A4, A5)$ ,  $(A5, A6)$ ,  $(A6, A7)$ ,  $(A6, A8)$ ,  $(A7, A9)$ ,  $(A8, A9)\}$ . For V1, the nodes A3, A4, A5, and A7 are events and tasks that differ in their type from their counterparts in the reference model. However, based on the notion of correspondence outlined above, they correspond to the respective tasks in the reference model. Thus, the TAR set of V1,  $TAR_{V1}$ , resolves to the same set as for the reference model,  $TAR_{RM}$ . Hence,  $dist(RM, V1) = 0$  and  $REPL(RM, V1) = 1$ . Put differently, the TAR-similarity metric rates the reference model and V1 as fully replaceable. This is the desired result.

When considering V2, the tasks A3, A4, A5, and A7 do not correspond to their counterparts in the reference model. Therefore, the TAR set of V2,  $TAR_{V2}$  resolves to  $\{(A1, A2), (A1, A3_{V2}), (A1, A4_{V2}), (A2, A5_{V2}), (A3_{V2}, A5_{V2}), (A4_{V2}, A5_{V2}), (A5_{V2}, A6), (A6, A7_{V2}), (A6, A8), (A7_{V2}, A9), (A8, A9)\}$ . Consequently,  $TAR_{RM} \cap TAR_{V2}$  is limited to  $\{(A1, A2), (A6, A8), (A8, A9)\}$ . Since  $|TAR_{RM} \cup TAR_{V2}|$  resolves to 19, the replaceability metric bears the following value:  $REPL(RM, V2) = 1 / (19 / 3) \approx 0.16$ . The reference model and V2 are not considered as identical.

Summarizing the results, TAR-similarity, when computed based on our our notion of correspondence, passes the evaluation, as it rates RM and V1 as replaceable and RM and V2 as dissimilar.

**Causal Behavioral Profiles:** Although the source defining causal behavioral profiles [31] uses BPMN process models as example, it states that it is necessary to map the models into a formal representation. This mapping is not clarified in the paper, but deferred to another paper [5]. The applicability of this mapping for BPMN 2.0 process models is questionable, since it refers to an outdated version of BPMN and, therefore, ignores aspects that are imperative to the execution semantics of a BPMN 2.0 process, such as different task types. Nevertheless, using our notion of correspondence, metrics computation for the process models is rather straight-forward. The structure of the three process models is identical, hence, their behavioral profiles are also identical, given they are computed based on node labels or execution traces. As before, the distinguishing difference lies in the nodes that differ between the models: A3, A4, A5, and A7. In the case of the reference model and V1, these nodes are considered to correspond to their counterparts by our notion of correspondence. Therefore, the two process models are considered to be identical and fully replaceable. Again, this is the desired result.

However, the assessment of the replaceability of the reference model and V2 uncovers a crucial problem. For A3, A4, A5, and A7, no corresponding nodes can be found for the reference model in V2 and vice versa. In this

case, as described in Sect. 2.3, all behavioral relations involving these nodes are omitted from the behavioral profiles. This means, replaceability is solely computed by considering the behavioral relations among A1, A2, A6, A8, and A9. These relations are completely identical in both models. Therefore, the result is the same as for the comparison of the reference model and V1: For causal behavioral profiles,  $REPL(RM, V2) = 1 = REPL(RM, V1)$  applies, and the two models are considered fully identical. This is not a desirable result, since we expect  $REPL(RM, V2) < 1$ .

This finding indicates that causal behavioral profiles are not applicable for our use case of computing replaceability. It seems that our notion of node correspondence is too restrictive for a meaningful application of the metric.

The result of the discussion in this section can be expressed as follows: TAR-similarity has been found to be applicable for the evaluation of replaceability. In contrast, causal behavioral profiles have not passed the test, and we cannot recommend this metric for our application scenario.

## 4 Summary and Conclusion

In this paper, we discussed the suitability of existing metrics for process model similarity, for evaluating the replaceability of executable process models. Essentially, replaceability can be reduced to a problem of similarity. For the computation of this property, a large body of metrics does already exist. A categorization according to the area of application limits the field of promising metrics to dependency graphs, TAR-similarity, the string edit distance of sets of traces, causal footprints, and causal behavioral profiles. The main issue is that the metrics in their current form fail to take node types into account, i.e., the absence of a proper node matching function. When taking node types into account, TAR-similarity [37] seems to be the most promising candidate metric.

The evaluation presented here cannot be considered as a final answer, but rather as a motivating example, and further work is needed. The focus of this work should lie less on the definition of new metrics, but rather on the evaluation of proper node mapping functions. Furthermore, studies that compare metrics would benefit significantly from the existence of sufficiently large corpora of process models that can be used as a benchmark. Such corpora are under development [30], but are still far from being accepted as a reference benchmark.

## References

1. Akkiraju, R., Ivan, A.: Discovering Business Process Similarities: An Empirical Study with SAP Best Practice Business Processes. In: 8th International Conference on Service Oriented Computing (ICSOC). pp. 515–526. San Francisco, CA, USA (December 7–10 2010)
2. Bae, J., Caverlee, J., Liu, L., Rouse, W.B.: Process Mining, Discovery, and Integration using Distance Measures. In: International Conference on Web Services. pp. 479–488. Chicago, USA (September 2006)

3. Becker, M., Laue, R.: A Comparative Survey of Business Process Similarity Measures. *Computers in Industry* 63(2), 148–167 (2012)
4. Bosch, J.: Speed, Data, and Ecosystems: The Future of Software Engineering. *IEEE Software* 33(1), 82–88 (Jan/Feb 2016)
5. Dijkman, R., Dumas, M., Ouyang, C.: Semantics and Analysis of Business Process Models in BPMN. *Information and Software Technology* 50(12), 1281–1294 (2009)
6. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. *Information Systems* 36(2), 498–516 (2011)
7. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: *Business Process Management Conference*. pp. 48–63. Ulm, Germany (September 2009)
8. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley (2005), ISBN: 978-0-471-66306-5
9. Dumas, M., García-Bañuelos, L., Dijkman, R.: Similarity Search of Business Process Models. *IEEE Data Engineering Bulletin* 32(3), 23–28 (2009)
10. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: *Asia-Pacific Conference on Conceptual Modelling (APCCM)*. pp. 71–80. Ballarat, Australia (January/February 2007)
11. Ferme, V., Ivanchikj, A., Pautasso, C.: A Framework for Benchmarking BPMN 2.0 Workflow Management Systems. In: *13th International Conference on Business Process Management (BPM 2015)*. Innsbruck, Austria (August 2015)
12. Geiger, M., Harrer, S., Lenhard, J., Casar, M., Vorndran, A., Wirtz, G.: BPMN Conformance in Open Source Engines. In: *9th International IEEE Symposium on Service-Oriented System Engineering (SOSE)*. San Francisco Bay, USA (March/April 2015)
13. Harrer, S.: Process Engine Selection Support. In: *OTM Academy*. Amantea, Italy (October 2014)
14. Harrer, S., Lenhard, J., Wirtz, G.: Open Source versus Proprietary Software in Service-Oriented: The Case of BPEL Engines. In: *11th International Conference on Service Oriented Computing (ICSOC)*. pp. 99–113. Berlin, Germany (December 2–5 2013)
15. Hoenisch, P., Schulte, S., Dustdar, S., Venugopal, S.: Self-Adaptive Resource Allocation for Elastic Process Execution. In: *IEEE Sixth International Conference on Cloud Computing*. pp. 220–227. Santa Clara, CA, USA (June 2013)
16. ISO/IEC: Systems and software engineering – System and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models (2011), 25010:2011
17. ISO/IEC: ISO/IEC 19510:2013 – Information technology - Object Management Group Business Process Model and Notation (November 2013), v2.0.2
18. Klinkmüller, C., Weber, I., Mendling, J., Leopold, H., Ludwig, A.: Increasing Recall of Process Model Matching by Improved Activity Label Matching. In: *11th International Conference on Business Process Management*. pp. 211–218. Beijing, China (August 26–30 2013)
19. Kunze, M., Weidlich, M., Weske, M.: Behavioral Similarity – A Proper Metric. In: *9th International Conference on Business Process Management*. pp. 166–181. Clermont-Ferrand, France (August, September 2011)
20. Lenhard, J.: Improving Process Portability through Metrics and Continuous Inspection. In: Reichert, M., Oberhauser, R., Grambow, G. (eds.) *Advances in Intel-*

- ligent Process-Aware Information Systems. Springer-Verlag, Germany (2016), to appear
21. Lenhard, J.: Portability of Process-Aware and Service-Oriented Software: Evidence and Metrics. Ph.D. thesis, University of Bamberg, Germany (2016)
  22. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady* 10(8), 707–710 (1966)
  23. Li, C., Reichert, M., Wombacher, A.: On Measuring Process Model Similarity Based on High-Level Change Operations. In: 27th International Conference on Conceptual Modeling. pp. 248–264. Barcelona, Spain (October 2008)
  24. Mens, T., Wermelinger, M., Ducasse, S., Demeyer, S., Hischfeld, R., Mehdi, J.: Challenges in Software Evolution. In: 8th International Workshop on Principles of Software Evolution. Lisbon, Portugal (September 2005)
  25. Minor, M., Tartakovski, A., Bergmann, R.: Representation and Structure-Based Similarity Assessment for Agile Workflows. In: ICCBR. pp. 224–238. Belfast, Northern Ireland, UK (August 13–16 2007)
  26. Newman, S.: Building Microservices. O'Reilly Media (February 2015)
  27. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented Computing. *Communications of the ACM* 46(10), 24–28 (October 2003)
  28. Prescher, J., Mendling, J., Weidlich, M.: The Projected TAR and its Application to Conformance Checking. In: Entwicklungsmethoden für Informationssysteme und deren Anwendung (EMISA). pp. 151–164. Vienna, Austria (September 13–14 2012)
  29. Santini, S., Jain, R.: Similarity Measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(9), 871–883 (1999)
  30. Thaler, T., Dadashnia, S., Sonntag, A., Fettke, P., Loos, P.: The IWi Process Model Corpus. Tech. Rep. 199, Publications of the Institute for Information Systems, Saarland University, Saarbrücken, Germany (October 2015)
  31. Weidlich, M., Mendling, J., Weske, M.: Efficient Consistency Measurement Based on Behavioral Profiles of Process Models. *IEEE Transactions on Software Engineering* 37(3), 410–429 (2011)
  32. Weidlich, M., Sagi, T., Leopold, H., Gal, A., Mendling, J.: Predicting the Quality of Process Model Matching. In: 11th International Conference on Business Process Management. pp. 203–210. Beijing, China (August 26–30 2013)
  33. Wombacher, A., Li, C.: Alternative Approaches for Workflow Similarity. In: 7th International Conference on Services Computing. pp. 337–345. Miami, Florida, USA (July 2010)
  34. Yan, Z., Dijkman, R., Grefen, P.: Fast Business Process Similarity Search with Feature-Based Similarity Estimation. In: 18th International Conference on Cooperative Information Systems (CoopIS). pp. 60–77. Crete, Greece (2010)
  35. Yan, Z., Dijkman, R., Grefen, P.: Business Process Model Repositories – Framework and Survey. *Information and Software Technology* 54(4), 380–395 (2012)
  36. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, *Advances in Database Systems*, vol. 32. Springer (2006), ISBN 978-0-387-29146-8
  37. Zha, H., Wang, J., Wen, L., Wang, C., Sun, J.: A workflow net similarity measure based on transition adjacency relations. *Computers in Industry* 61(5), 463–471 (2010)
  38. Zhou, Z., Gaaloul, W., Gao, F., Shu, L., Tata, S.: Assessing the Replaceability of Service Protocols in Mediated Service Interactions. *Future Generation Computer Systems* 29(1), 287–299 (2013)

# Towards Function and Data Shipping in Manufacturing Environments: How Cloud Technologies leverage the 4<sup>th</sup> Industrial Revolution

Michael Falkenthal<sup>1</sup>, Uwe Breitenbücher<sup>1</sup>, Maximilian Christ<sup>2</sup>,  
Christian Endres<sup>1</sup>, Andreas W. Kempa-Liehr<sup>2,3</sup>,  
Frank Leymann<sup>1</sup>, and Michael Zimmermann<sup>1</sup>

<sup>1</sup> University of Stuttgart, Institute of Architecture of Application Systems  
Universitätsstr. 38, 70569 Stuttgart, Germany

[lastname]@iaas.uni-stuttgart.de

<sup>2</sup> Blue Yonder GmbH

Ohiostr. 8, 76149 Karlsruhe, Germany

maximilian.christ@blue-yonder.com

<sup>3</sup> University of Freiburg, Freiburg Materials Research Center  
Stefan-Meier-Str. 21, 79104 Freiburg, Germany

kempa-liehr@mf.uni-freiburg.de

**Abstract.** Advances in the field of cloud computing and the Internet of Things are boosting the 4<sup>th</sup> industrial revolution. New research and developments foster the emergence of smart services, which augment conventional machinery to become smart cyber-physical systems. The resulting systems are characterized by providing preemptive functionality to automatically react on circumstances and changes in their physical environment. In this paper we sketch our vision of how to automatically provision smart services in manufacturing environments, whereby the paradigms of function and data shipping are specifically considered. To base this approach upon a clear understanding of influences, we point out key challenges in the context of smart services for Industry 4.0.

**Keywords:** data shipping, function shipping, fourth industrial revolution, cyber-physical systems, TOSCA

## 1 Introduction and Background

The availability of cheap sensors and the increasing connectivity between devices are the drivers behind the accelerating availability of data in industrial operations [2,20]. The evolving Internet of Things (IoT) is formed by “*embedded devices (Things) with Internet connectivity, allowing them to interact with each other, services, and people on a global scale*” [15] and is a central enabler of Industry 4.0, which heavily relies on predicting future device states by combining the knowledge of device attributes with historic and current sensor readings.

*“Industry 4.0 is a collective term for technologies and concepts of value chain organization”* [11] and is a synonym for the 4<sup>th</sup> industrial revolution.

An important application of Industry 4.0 is the anticipation of future device states in the context of predictive maintenance [14]. Its task is the discrimination of properly working machines from those, which are likely to evolve a specific risk of failure. Collected data might describe a fleet of machines, with each machine being characterized by certain time invariant data (e.g., geo-coordinates, date of putting into operation), time variant control parameters (e.g., the currently used tool head), collections of regularly updated sensor data (e.g., time series containing pressure and temperature measurements), and the results of some successive inspection reports, which might indicate specific technical flaws. On basis of the inspection reports, the machines can be divided into two groups: Machines for which a specific technical flaw has been observed and machines, for which this flaw has not been observed yet. The optimization task of predictive maintenance is to determine the relation of statistical health factors to operating costs and failure risks by means of machine learning algorithms [21].

This optimization task is provided by so-called *smart services*, which are not only *“reactive or even proactive”* but actually *“fundamentally preemptive”* and *“based upon hard field intelligence”* given by *“awareness and connectivity”* [1]. The vast amount of mounted sensors on machinery enables a smart service to perform detailed analyses of production steps [9] and to subsequently influence the production flow by adapting machine configurations and adjustments in an automated fashion. As a consequence, these kind of machines become smart cyber-physical systems. It is expected that this approach will contribute significantly to the expectations being associated with Industry 4.0.

In such scenarios, huge amounts of metering data are generated. In case of critical operations, this data has to be processed in parallel in order to react to the process under optimization in a timely manner [12]. Accordingly, in many applications it is insufficient or almost impossible to transfer the data to a central data store or a public cloud environment providing adequate processing power and storage for analyzing the data [10]. Further, latencies and limited network bandwidth make centralized processing unsuitable. Instead, analytics or data aggregation functionality has to be provisioned as close to the data sources as possible. Such a scenario, where functionality is shipped and provisioned close to the data sources, is called *function shipping*.

While some applications benefit from the function shipping approach, also other use cases exist that do not require strict reaction times for processing analysis. Instead, the sensor data from different sources and respective meta-information need to be merged, which raises the demand for self-documenting file formats [18]. In these scenarios, the data have to be transferred to a powerful central execution environment, either self-hosted or in a public cloud environment. This approach is called *data shipping*, because the data is transferred to the functionality that has to process the data. Summarizing, different provisioning strategies for providing and running analytics functionality seem to be feasible to cover the needs of smart service development and operation.



However, there is still a lack of proper technologies and tools in order to efficiently support the development and provisioning of smart services. Further, the concepts of function and data shipping have to be particularly applied to manufacturing environments considering the specific challenges such as handling of secret production data or time constraints for processing the data.

Thus, in this paper we introduce a standards-based vision of how to automatically provision smart services in manufacturing environments, whereby the paradigms of function and data shipping are specifically considered. To base this approach upon a clear understanding of related issues, we point out key challenges in the context of smart services for Industry 4.0. The envisioned approach is currently implemented in the course of the project SePiA.Pro<sup>4</sup> (*Service Platform for intelligently optimizing Applications in Production and Manufacturing*) but should be applicable to smart services in general [19].

The remainder of this paper is structured as following: We discuss the key challenges for provisioning smart services in Section 2. We sketch a self-contained and secure packaging format for smart services in Section 3 and discuss how it addresses the identified challenges and how it enables to establish function and data shipping in the context of smart services for Industry 4.0. In Section 4, we explain how to utilize the packaging format by a toolchain and conclude this paper in Section 5.

## 2 Key Challenges

There are several key challenges, which have to be addressed in order to develop and run smart services for, respectively, in manufacturing environments. We categorize the identified challenges into *organizational challenges* and *technical challenges*. Organizational challenges mainly arise from the connection of formerly disconnected sets of data and reflect issues concerning the collaboration of former organizationally distinct (legal) entities like different departments, organizational units, subsidiaries, or even different companies. The technical challenges, in contrast, address difficulties that are driven by different technologies, statistical aspects of the machine learning algorithms and programming principles. In SePiA.Pro, we will investigate and address both types of challenges.

### 2.1 Organizational Challenges

**Restricted Access to Smart Services** Since smart services contain analytics algorithms that process detailed metering data from production processes, critical intellectual property of a company in the form of information about actual processing steps and produced parts could be reengineered from the algorithm implementations. Thus, access to a smart service, specifically the implementations of the contained analytics algorithms has to be restricted to authorized personal only. Neither non-authorized personal shall be able to inspect nor to configure the algorithms.

---

<sup>4</sup> <http://projekt-sepiapro.de>

**Data Security** As mentioned in the former section, smart services process data that contain trade secrets and confidential information. Thus, specifically in the case of data shipping, where data are transferred to an analytics environment, it has to be assured that access to the data, the data sources, and the persistency layers is strictly controlled. This implies that shipped data have to be encrypted.

**Data Ownership** The discussed data security aspects are closely related to data ownership issues. While smart services can process data close to the machinery, there might also be situations where data has to be transferred to an external execution environment, such as a public cloud or the data center of the smart service developer. The latter case especially emerges if smart service developers establish new business models, whereby they also offer processing environments for the analysis of the metered data as a service, besides the mere development and integration of smart services. Thus, the data owners, which are typically the companies that operate the metered machinery, must not forfeit control on where their data is sent to.

**Algorithms as Intellectual Property** Even though a customer purchases and uses a smart service, this does not necessarily imply that they also obtain the rights to reuse, manipulate, or adapt an algorithm contained in a smart service. So, the intellectual property of the developer of a smart service has to be respected by restricting access to the smart service itself.

**Smart Service Integrity** In order to enable new business models for smart service developers, an app store like concept seems to be necessary. Developers need the possibility to offer their smart services via platforms such as public repositories or marketplaces, which allow customers to easily search, purchase, and utilize smart services. This will boost the acceptance and success of smart services. However, if smart services are provided that way, their integrity has to be assured. This means that a smart service developer needs a means to assure that algorithms and data contained in a smart service are not corrupted or manipulated from any third-party.

## 2.2 Technical Challenges

**Use Case-specific Function and Data Shipping** In many industrial applications of machine learning algorithms the volume of the generated data forbids their transport to centralized databases or computing centers [10]. Instead, techniques for an efficient reduction of the data volume and local data analysis close to the machinery are needed [5]. Thus, the ability to ship and execute functionality, either to analyze or to aggregate data as close to the production environment and machinery as possible, is inevitable.

**Fast Adaption of Analytics Algorithms** There are two reasons why the machine learning parts of smart services need to adapt quickly. Firstly, the algorithms are mostly tailored to specific analysis scenarios. Thus, they handle intrinsic characteristics of the data to process. If there are changes in the underlying dynamics and the characteristics of the metered data, the algorithms have to be retrained in order to deal with such concept drifts. For example, it could be that a new product is produced on the same machinery. Secondly, the machine learning algorithms have to be adapted and reconfigured if the production environment or the optimization objectives itself are changing. Here, one could think of a physical reordering of the existing machinery as an exemplary cause for the adaption of the machine learning model.

**Stream and Batch Processing of Data** Most of the metering data from machinery is not captured in a static but a streaming fashion resulting in more and more timely annotated data sources. To unleash the full potential of this data for use cases such as predictive maintenance it is necessary to process it on the fly, which typically results in stream processing approaches. Besides, there are also batch processing scenarios, which require to store raw data in order to be processed and analyzed later on. For example, the training procedures for most machine learning algorithms are working in a batch fashion.

**Heterogeneous Technologies** Smart services are complex compositions of different technologies in order to meter data from machinery and process them based on machine learning algorithms. Thus, it is up to the smart service developer which technologies and libraries to use for implementing the analytics algorithms. Further, the technical circumstances of the production environments and the machinery as well as the available IT infrastructures, platforms, middlewares, and applications on the customer side increase the number of heterogeneous technologies and software artifacts that are required in order to implement a smart service. Exemplarily, the component to fetch metering data from machinery might be designed to communicate via OPC-UA<sup>5</sup> with metered machinery in a specific production environment, while MQTT<sup>6</sup> has to be used in another one. To enable the efficient development of smart services, developers are forced to design the different components of a smart service as replaceable as possible.

**Modularity** In order to gain savings in terms of development time and, thus, time to market, components of a smart service have to be self-contained and interoperable with other components by stable interfaces. To raise great benefits regarding development costs for creating new smart services or adapting existing ones to new production environments, components of a smart service need to be packaged to be easily reusable.

---

<sup>5</sup> <https://opcfoundation.org/about/opc-technologies/opc-ua/>

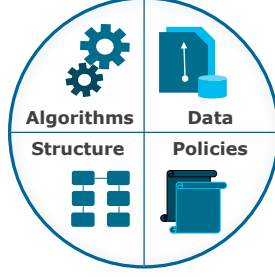
<sup>6</sup> <http://mqtt.org>

**Smart Service Provisioning** As a result of the formerly described challenges, smart services have to be designed as analytics artifacts that can be dynamically changed. This can be based on changes of the analyzed data, which has direct impact on the used machine learning algorithms. Moreover, this can also be owing to changing technical circumstances at customer side or the emergence of new technologies. Thus, smart services should be fully automatically provisionable in order to efficiently deal with their dynamic character.

### 3 Self-Contained Packaging Format for Smart Services

In order to enable efficiently developing, shipping, and deploying smart services, a self-contained packaging format for smart services is inevitable. Thus, in SePiA.Pro, one of the main objectives is to develop a *Smart Service Archive (SMAR)* format that enables bundling all artifacts of a smart service. The conceptual structure of a SMAR is depicted in Figure 1. The Smart Service Archive format is based on TOSCA [17,16], an OASIS standard that enables describing applications to be provisioned in a portable manner. In particular, SMARs are based on the TOSCA *Cloud Service Archive (CSAR)*, which is a standardized archive format for packaging all required data to enable the automated provisioning of the respective application. TOSCA and CSARs are explained in the following in order to point out how the standardized format has to be enhanced and extended to support the packaging of smart services.

CSARs contain several data required to automatically provision an application. First, a CSAR contains a *topology model*, which is a directed graph describing the structure of the application to be provisioned. The topology model consists of nodes, which represent the components of the application such as Web-servers and virtual machines, and edges between these nodes, which represent the dependencies between the components. Nodes are called *node templates*, the dependencies are called *relationship templates*. The TOSCA standard provides a means to specify types for node and relationship templates in the form of so called *node types* and *relationship types*, which allows to specify the semantics of the respective templates on a type level. Those types enable to populate a topology model by different manifestations and instances of a node or relationship, respectively. Therefore, they are reusable in arbitrary topology models and provide *reusable building blocks* for creating new applications. The actual artifacts that implement the components, such as Java classes or other binaries of an analytics algorithm, can be placed into the archive by means of so called *deployment artifacts*. These artifacts are associated with the corresponding node template for which they provide the implementation. Through this format, TOSCA tackles challenges regarding the reusability of components by means of node types. However, based on the presented key challenges in Section 2, it seems to be valuable to also treat algorithms and data along with policies as first level modeling concepts for smart services. Therefore, the SMAR format needs to extend the CSAR format by (i) *algorithms*, (ii) *data*, and (iii) *data policies* as conceptual first level modeling entities, as depicted in Figure 1. Although TOSCA already



**Fig. 1.** Concept of the Smart Service Archive (SMAR)

supports policies for cloud applications in general [17,22] and description languages, such as USDL [3], can be combined with TOSCA to add non-functional service descriptions [8], a strict data policy approach for manufacturing data is missing. Treating algorithms and data as first level modeling concepts allows to enable function and data shipping approaches, either by clearly defining how functionality has to be shipped close to the data, or by enforcing the provisioning of smart services based on policies and further rules through an adequate toolchain. Therefore, SePiA.Pro will research on how already available TOSCA modeling concepts can be generally reused, which of the key challenges outlined in this paper can be translated into already existing modeling concepts of TOSCA, and where the TOSCA modeling approach eventually has to be extended by new concepts to support the requirements of smart services.

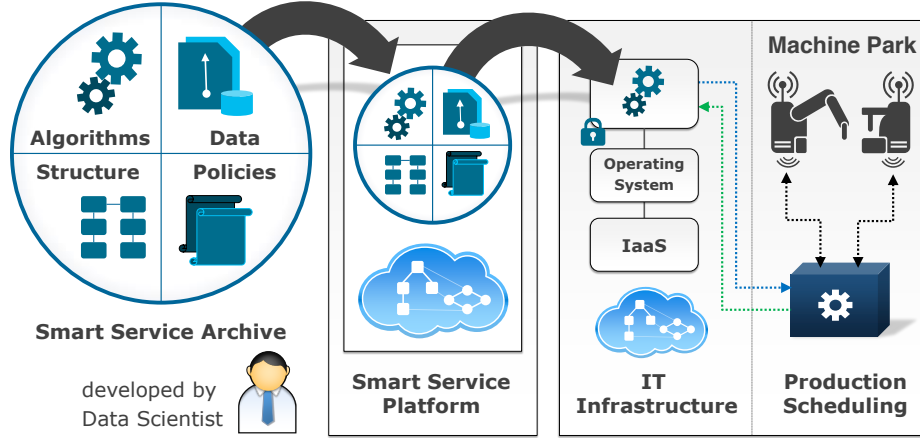
#### 4 Function and Data Shipping for Manufacturing Environments

The capability to ship functionality as close to metered machinery, or to ship data to powerful analytics environments is vital to tailor smart services to different use cases. Besides the archive format for smart services, as introduced in Section 3, also a *Smart Service Ecosystem* to enable (i) modeling, (ii) shipping as well as (iii) provisioning and managing smart services is required and one of the main deliverables of the SePiA.Pro project.

To support these features, the planned toolchain will base on tools from the *OpenTOSCA Ecosystem*<sup>7</sup>, which is an open source toolchain that supports modeling and provisioning of CSARs. The open-source TOSCA modeling tool *Winery*<sup>8</sup> [13], which supports modeling of topology models using the visual notation VINO4TOSCA [7], will be extended and enhanced to a *Smart Service Design Platform* in order to support modeling of SMARs. Further, a *Smart Service Repository and Self-Service Portal* will be developed based on OpenTOSCA's *Vinothek* [6] in order to efficiently ship smart services packaged in the SMAR

<sup>7</sup> <https://www.iaas.uni-stuttgart.de/OpenTOSCA/>

<sup>8</sup> <https://projects.eclipse.org/projects/soa.winery>



**Fig. 2.** Function and Data Shipping Enabled by the Self-Contained Packaging Format for Smart Services

format on the one hand, and to trigger the provisioning of smart services by end users on the other hand. Finally, the *OpenTOSCA Container* [4], an open-source runtime environment for CSARs, will be extended and enhanced in order to be able to process the new SMAR format.

The Smart Service Ecosystem will support the development and application of smart services as shown in Figure 2. The SMAR format is the key enabler to automatically provision smart services. As shown on the left of the figure, a data scientist can develop an analytics algorithm and put it into a SMAR along with references to data that have to be processed. References to data can point to files also contained in the archive, e.g., if the algorithm requires static data that does not change over time. But they can also point to sensors from machinery that constantly deliver metering data. The data can be secured by adding specific data policies. These could declare, e.g., that the metering data must not leave the IT infrastructure of the customer that will run the smart service. All these steps in Figure 2 will be supported by the Smart Service Design Platform<sup>9</sup>.

SMARs can then be deployed on a Smart Service Platform that includes the extended OpenTOSCA Container. The Smart Service Platform is capable of provisioning new instances of a smart service, which is bundled in a deployed SMAR. This is conceptually illustrated in Figure 2 by the IT infrastructure box, where an application stack is sketched that consists of an operating system, installed on a virtual machine in a private infrastructure as a service cloud. The analytics algorithms developed by the data scientist are provisioned on top of the operating system and wired to the sensors in the machine park via a production scheduling system. In this scenario, the production scheduling system wraps

<sup>9</sup> SMARs that are packaged in this manner can be uploaded to a Smart Service Repository and Self-Service Portal to ease the shipping to customers. For the sake of simplicity this step is omitted in Figure 2.

the access to metering data from machinery as well as to functionality that enables to adjust and to configure machinery programmatically. The connection is established bidirectionally to receive metering data from the machinery and to also send commands for adjustments of the machinery back to the production scheduling system. Thus, this provisioning flow indicates how machinery can be augmented by smart services and how functionality can be automatically shipped and provisioned closely to the sensors.

Other scenarios could cause to provision the analytics stack along with the algorithms in a public cloud environment, e.g., if the data to process is allowed to physically leave the IT infrastructure of the data owner. In this case, functionality is shipped automatically to the respective public cloud and, besides, also data is shipped to the analytics stack in the public cloud environment.

Finally, scenarios are possible, where the analytics stack is provisioned at a data center of the data scientist. This last scenario shows that also pure data shipping approaches are possible by the depicted toolchain in combination with the SMAR format since the extended format allows also packaging data only.

## 5 Conclusion

The presented key challenges are the objects of investigation for the project SePiA.Pro. A huge potential of smart services to generate business value in the field of Industry 4.0 on the one hand and the missing formats to provision smart services on the other hand are motivating this research project. SePiA.Pro will transfer the presented challenges into research questions in order to refine and enhance the TOSCA standard to the field of smart services in production environments. The resulting format will enable function and data shipping scenarios in the context of Industry 4.0. Although the project is still in its opening stages at the time of writing this paper, the presented vision of a new packaging format for smart services along with an OpenTOSCA-based toolchain show how cloud computing technologies may boost developments in the sector of manufacturing.

**Acknowledgments.** This work is partially funded by the project SePiA.Pro (01MD16013F) of the BMWi program Smart Service World.

## References

1. Allmendinger, G., Lombreglia, R.: Four strategies for the age of smart services. *Harvard Business Review* 83(10), 131 (2005)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. *Computer Networks* 54(15), 2787–2805 (2010)
3. Barros, A., Oberle, D. (eds.): *Handbook of Service Description: USDL and Its Methods*. Springer (2012)
4. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In: *Proceedings of the 11<sup>th</sup> International Conference on Service-Oriented Computing (ICSOC 2013)*. pp. 692–695. Springer (2013)

5. Bolón-Canedo, V., Snchez-Maroo, N., Alonso-Betanzos, A.: Feature Selection for High-Dimensional Data. *Artificial Intelligence: Foundations, Theory, and Algorithms*, Springer (2015)
6. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F.: Vinothek - A Self-Service Portal for TOSCA. In: *Proceedings of the 6<sup>th</sup> Central-European Workshop on Services and their Composition (ZEUS 2014)*. pp. 69–72. CEUR-WS.org (2014)
7. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., Schumm, D.: Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA. In: *On the Move to Meaningful Internet Systems: OTM 2012 (CoopIS 2012)*. pp. 416–424. Springer (2012)
8. Cardoso, J., Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: Cloud computing automation: Integrating usdl and toska. In: *Proceedings of the 25<sup>th</sup> International Conference on Advanced Information Systems Engineering (CAiSE 2013)*. Springer (2013)
9. Chaouchi, H.: *The Internet of Things: Connecting Objects*. ISTE, Wiley
10. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 29(7), 1645–1660
11. Hermann, M., Pentek, T., Otto, B.: Design Principles for Industrie 4.0 Scenarios. In: *Proceedings of the 49<sup>th</sup> Hawaii International Conference on System Sciences (HICSS)*. pp. 3928–3937 (2016)
12. Kempa-Liehr, A.W.: Performance analysis of concurrent workflows. *Journal of Big Data* 2(10), 1–14 (2015)
13. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: Winery – A Modeling Tool for TOSCA-based Cloud Applications. In: *Proceedings of the 11<sup>th</sup> International Conference on Service-Oriented Computing (ICSOC 2013)*. pp. 700–704. Springer (2013)
14. Mobley, R.K.: *An introduction to predictive maintenance*. Elsevier Inc., 2 edn.
15. Mukhopadhyay, S.C. (ed.): *Internet of Things, Smart Sensors, Measurement and Instrumentation*, vol. 9. Springer International Publishing (2014)
16. OASIS: *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0* (2013)
17. OASIS: *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0* (2013)
18. Riede, M., Schueppel, R., Sylvester-Hvid, K.O., Kühne, M., Röttger, M.C., Zimmermann, K., Liehr, A.W.: On the communication of scientific data: The full-metadata format. *Computer Physics Communications* 181(3), 651–662 (2010)
19. Smart Service Welt Working Group/acatech (Eds.): *Smart Service Welt Recommendations for the Strategic Initiative Web-based Services for Businesses*. Tech. rep., Berlin (2015)
20. Sundmaeker, H., Guillemin, P., Friess, P., Woelffle, S.: Vision and challenges for realising the internet of things. *European Commission Information Society and Media* (2010)
21. Susto, G.A., Schirru, A., Pampuri, S., McLoone, S., Beghi, A.: Machine Learning for Predictive Maintenance: A Multiple Classifier Approach. *Transactions on Industrial Informatics* 11(3), 812–820 (2015-06)
22. Waizenegger, T., Wieland, M., Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Mitschang, B., Nowak, A., Wagner, S.: Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing. In: *On the Move to Meaningful Internet Systems: OTM 2013 Conferences*. pp. 360–376. Springer (2013)



# Flexible Execution and Modeling of Data Processing and Integration Flows

Pascal Hirmer

Institute of Parallel and Distributed Systems,  
Universität Stuttgart,  
Universitätsstraße 38,  
70569 Stuttgart, Germany,  
Pascal.Hirmer@ipvs.uni-stuttgart.de

**Abstract** Today, the amount of data highly increases within all domains due to cheap hardware, fast network connections, and an increasing digitization. Deriving information and, as a consequence, knowledge from this huge amount of data is a complex task. Data sources are oftentimes very heterogeneous, dynamic, and distributed. This makes it difficult to extract, transform, process and integrate data, which is necessary to gain this knowledge. Furthermore, extracting knowledge oftentimes requires technical experts with the necessary skills to conduct the required techniques. For my PhD thesis, I am working on a new and improved approach for data extraction, processing, and integration by: (i) facilitating the definition and processing of data processing and integration scenarios through graphical creation of flow models, (ii) enabling an ad-hoc, iterative and explorative approach to receive high-quality results, and (iii) a flexible execution of the data processing tailor-made for users' non-functional requirements. By providing these means, I enable a more flexible data processing by a wider range of users, not only limited to technical experts. This paper describes the approach of the thesis as well as the publications until today.

**Keywords:** Big Data, Data Integration, Data Flows, Pipes and Filters

## 1 Summary of the Research Problem

Today, the highly advanced connectivity of systems as well as their increasing collaboration lead to new challenges regarding data exchange, data processing and data integration. Especially the emerging topics Internet of Things (IoT), advanced manufacturing, or case management are in need of a distributed, flexible processing and integration of heterogeneous data to gain important knowledge. Furthermore, the possibility to integrate data sources “ad-hoc” is pursued to enable high flexibility. In this context, ad-hoc means: (i) enabling an iterative and explorative “trial-and-error”-like data integration using different data sources without the need of creating, e.g., complex Extract-Transform-Load (ETL) processes and data warehouses [4], (ii) the flexible adding and removing

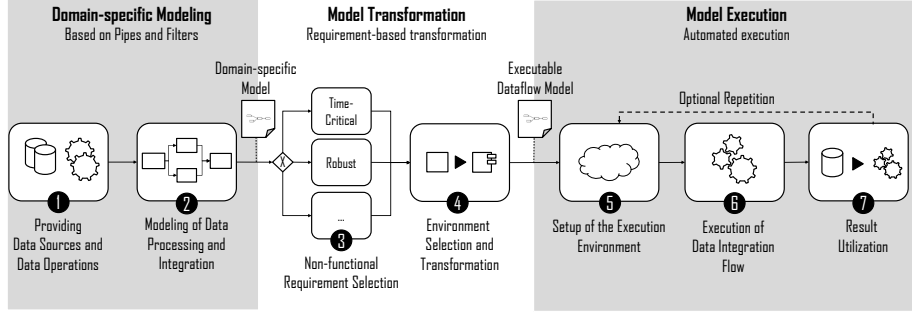
of data sources with low effort (preferably automatically), and (iii) an easy adaptation of the way data is processed and integrated considering the newly added or removed data sources. However, this flexibility also leads to a high complexity compared with statically defined data integration techniques such as ETL processes. Those flexible scenarios are oftentimes realized using data flow and streaming technologies based on different execution models. The use of a data flow model enables its flexible generation based on dynamic needs. That is, if a new data source is added to an existing integration, the flow model can be re-generated and re-executed in order to include it. Currently, existing solutions are mostly tailor-made for a specific use case scenario and do not offer a generic solution. Another issue in this research area is coping with unstructured data. Most systems do not support the simultaneous processing and integration of structured and unstructured data. However, integrating unstructured data can lead to valuable information. For example, in advanced manufacturing environments, the natural language input of a worker or textually described manuals can lead to higher-level information, which can e.g., be used for an automatic dissolving of occurring problems as described in [13]. In conclusion, the main challenges of this research area are: (i) providing a means to model data processing and integration without having to design complex ETL processes or data warehouses using flow-based models, (ii) automatic tethering of different data sources, both structured and unstructured, (iii) reducing the necessary technical expertise of the involved users through a high degree of automation, (iv) integrating heterogeneous data formats in an ad-hoc manner as defined before, and (v) achieving a flexible execution of the data processing, i.e., a dynamic execution dependent on non-functional requirements. These issues are addressed by this PhD thesis. How I will cope with this research problem is described throughout this paper. All described concepts are based on previous author publications that are listed in Sect. 7.

The remainder of this paper is structured as follows: in Sect. 2, the proposed approach is described by introducing a new method that copes with the described research problem. After that, in Sect. 3, I propose an architecture this method can be applied to. In Sect. 4, the prototypical implementation of the concept is described as a proof of concept. Sect. 5 contains related work and Sect. 6 gives a summary as well as an outlook on future work. Finally, Sect. 7 lists selected thesis-related author publications.

## 2 Proposed Approach

I plan to address the research issues mentioned in the previous chapter in multiple steps, which are depicted in Figure 1.

In the first step, the focus lies on the tethering of data sources to be processed and integrated and on providing data processing operations as services. On the one hand, data sources have to be bound for data extraction, on the other hand the data sources and the contained data to be used for processing and integration should be cleaned and transformed into a uniform data format to be easy to process and to ensure a high quality of the results.



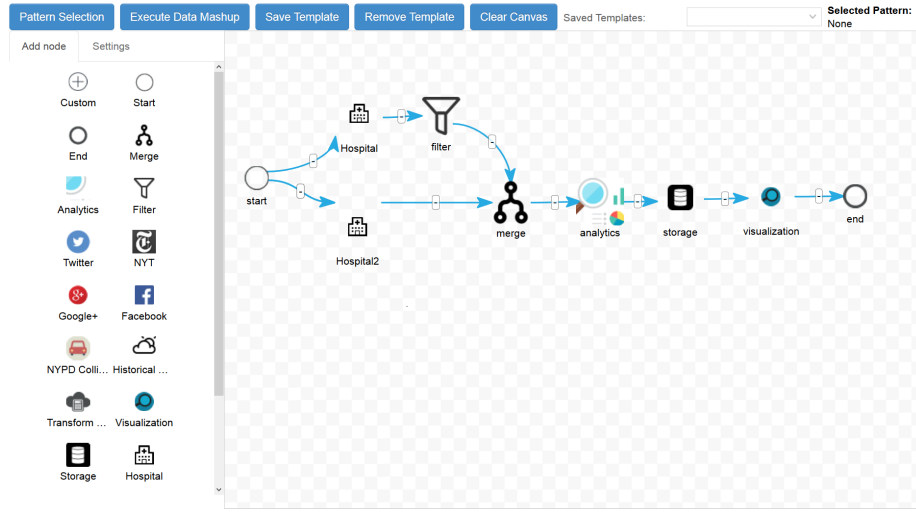
**Figure 1.** Data Processing and Integration Approach [12]

The binding of data sources is conducted through the so called *resource management platform* (RMP), a middleware component that abstracts from the concrete data sources and provides the data for integration in the suitable format. The RMP offers the following functionality: (i) automated binding of data sources through adapters, (ii) normalization and transformation of data to achieve a uniform data structure, (iii) data cleaning to improve data quality, and (iv) data provisioning through a uniform REST interface. The RMP is a powerful data platform that highly improves the efficiency and effectiveness for data processing because the data is already provided in the right format and has a high degree of quality. The concepts of the Resource Management Platform have been partly published in [10,11]. Once the data sources are bound to the resource management platform, they can be used for modeling of how data is processed and integrated.

Furthermore, services need to be registered that process the data that is extracted and provisioned by the RMP, i.e., conduct data processing operations such as filters, aggregation, or even sophisticated data analysis such as clustering or classification [19]. These services can be registered at a *service registry* and can then be used for flow-based modeling of data processing and integration scenarios as described next.

In the second step, the data sources and data operations to be used for data processing and integration are defined through graphical modeling based on the Pipes and Filters pattern [14]. In this pattern, the *filters* are software components such as e.g., web services, that have the same interfaces and the same data exchange format. These filters are interconnected using the *pipes*. The main advantage is that, due to the uniform interfaces, filters can be interconnected in an arbitrary manner. The necessary uniform data exchange format is achieved through the resource management platform. An example for such a Pipes and Filters based model is depicted in Fig. 2. This example was modeled using the implemented prototype for the approaches presented in this paper, which is called *FlexMash* (cf. Sect. 4). As depicted, data sources available through the RMP and data operations provided by services are shown in a palette on the left. These can be inserted into the canvas and can be connected to each other via drag and drop. Furthermore, once the model is finished, the single nodes can be

## FlexMash Builder



**Figure 2.** Example for Pipes and Filters-based modeling of data processing and integration

configured, for example, by defining the criteria for a data filter operation. In this example, the goal is integrating data from two hospitals, conducting analysis, and visualize the result. To realize this, the data is extracted, one of the resulting data sets is filtered because not all of its information are needed, and the data sources are merged, e.g., by a join over the primary key attribute ID. Next, the merged data is used for analysis, e.g., for clustering or classification, is stored, and finally visualized, e.g., in a dashboard. At this point, the resulting model is *non-executable*. This is important because it enables dynamic mapping onto different execution formats (cf. step four of the method).

After creating the model, in the third step of the method, the modeler can select non-functional requirements from a requirement catalog. This catalog contains well-known requirements that often occur when processing and integrating data. For example, such requirements are *robustness*, *security*, *efficiency*, and so on. These requirements are described in a text-based manner by providing users, typically non-IT experts, with the information necessary to choose the requirements most suitable for their needs, i.e., for their use case scenario. Furthermore, several requirements can be combined with each other. However, obviously, some of the requirements cannot be combined, for instance, robustness and efficiency is very hard to unify. These restrictions are also described in the requirement catalog. Based on the selected non-functional requirements, the runtime environment to process and integrate the data will be built automatically. As a consequence, the execution environment could be different for each execution. This leads to a very

dynamic execution, tailor-made for each user. Based on the selected requirements, the runtime environment is set up in a modular way, as described in step five.

In step four, the execution environment is selected based on the defined non-functional requirements. This is done through a graph-based algorithm, which traverses so called *requirement trees*, as described more detailed in Sect. 3.4 and in [7]. By doing so, a tailor-made execution environment, containing an execution engine (e.g., a BPEL workflow engine) and other necessary components, can be built suited for the non-functional requirements of each single user. Once the most suitable execution environment is selected, the domain-specific non-executable flow model created in step two is transformed into a suitable executable representation (e.g., a BPEL workflow) to be processed by the selected execution engine.

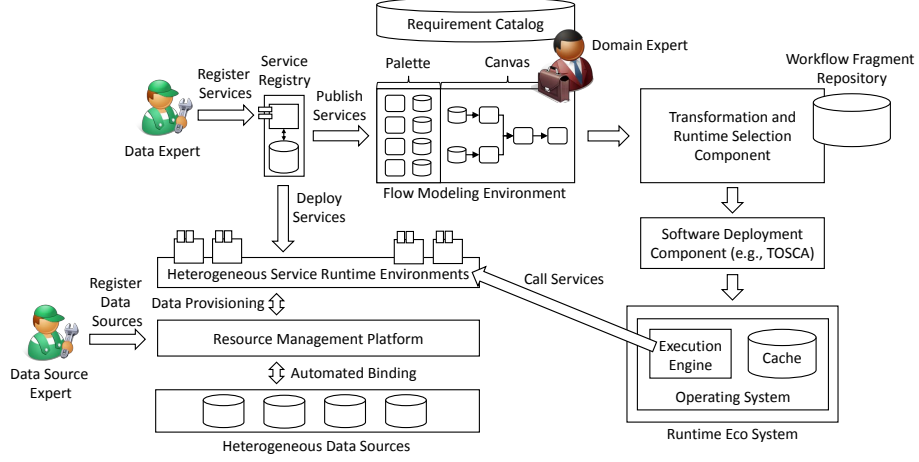
In the fifth step, the execution environment selected in step four is set up automatically. For example, cloud computing technologies can be used for the automated provisioning into arbitrary environments. In [8], we show how this can be achieved using the Topology and Orchestration Specification for Cloud Applications (TOSCA). In TOSCA, the software components to be set up are represented in application topologies through so called Node Templates (representing the software components) that are connected via Relationship Templates, defining their communication. These topologies can be automatically generated based on the component selection of step 4 and can be used for automated provisioning using the results of previous work [6].

In step six, the transformed, executable flow model that was created in step three is executed in the engine that was set up in the previous step. On execution, services are invoked for each step of the Pipes and Filters based model. The services have been registered in the first step of the method and offer diverse functionality for data processing and integration. In my prototype, I use approved execution engines such as BPEL workflow engines for execution because they offer many features that, e.g., ensure error handling, efficiency, security, etc. As already mentioned, the selection of the engine highly depends on the non-functional requirements of the users.

Finally, the last step is the usage of the integrated result, for example for visualization, analytics, or other value-adding approaches. This, however, is not part of this thesis.

### 3 Architecture

I introduce the architecture depicted in Fig. 3 to realize the approach proposed in Sect. 2. In this architecture, several components are involved that have been or will be designed to realize the introduced concepts. Those are (i) the service registry to manage data processing services, (ii) the resource management platform to manage data sources and normalize data to be processed, (iii) the modeling tool to create Pipes and Filters-based data processing and integration flows, (iv) the transformation and runtime selection component to select the runtime environment and transform the model into an executable representation, (v) the



**Figure 3.** Architecture of the approach

software deployment component to set up the execution environment, and (vi) the runtime eco system to execute the data processing and integration flows. These components are described in detail in the following.

### 3.1 Service Registry

The *service registry* enables data experts to provide services that process data. Data experts have deep knowledge of data processing, data integration, and analytics, and can provide the algorithms and the corresponding implementations for these operations. The implementations are encapsulated in services that process the data. On registration, two actions are initiated: (i) the service is automatically deployed into a suitable service runtime dependent on the type of service, e.g., Java SOAP Web Services or NodeJS-based REST services, and (ii) an abstract entry for the service is created in the palette of the modeling tool, so that domain experts can use it to model data integration and processing flows. The automated deployment of the services is conducted using the Topology and Orchestration Specification for Cloud Applications (TOSCA). In TOSCA, the environment of the service can be specified using so called topologies. These topologies can be used for automated provisioning of services on different kinds of infrastructure components. To be able to provide an entry for the service in the modeling tool’s palette, the properties, i.e., its input parameter (e.g., a filter criteria) have to be specified when registering the service. The registered services are stored in a database to enable their synchronization with the modeling tool, their re-hosting when the service or its runtime crashes, and the creation of multiple service instances to enable scalability. The services are tightly coupled with the resource management platform, which is described next.

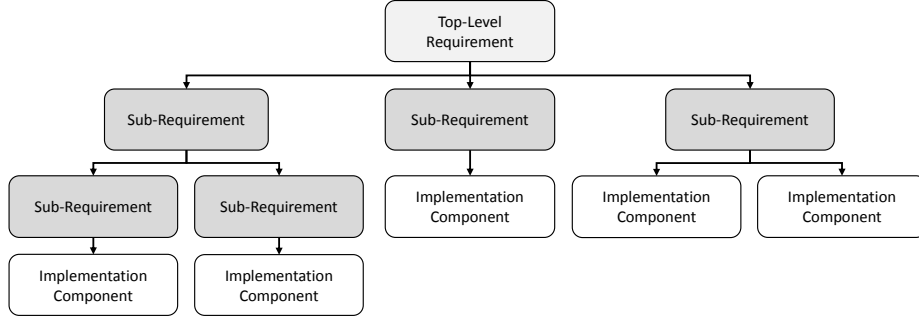
### 3.2 Resource Management Platform

The *resource management platform (RMP)* is a middleware component that offers an abstraction from the underlying data sources to be processed by our approach, and furthermore, contains functionality to transform data into a uniform format. As described, this is a precondition for Pipes and Filters-based processing of data. The resource management platform enables an easy binding of data sources through *data source adapters*. These adapters are stored in a repository and can be deployed on demand. More precisely, on registration of a new data source, data experts only have to provide some information about the data source such as the type (e.g., MySQL) and access information (IP, port, user, password), and the adapters can be automatically deployed and started with these information. On start of an adapter, it connects to the data source and is then able to extract information through the RMP. The RMP can then provision the data on demand to the services that process them. The automated deployment of these adapters is conducted similar to the service deployment in the service registry using TOSCA.

Through the RMP, I provide an abstraction from data sources so that data processing services can be implemented more easily. The services only have to access the RMP and do not have to implement the functionality to access the data sources themselves, which is very cumbersome due to their high heterogeneity. Furthermore, the RMP offers functionality to cache data, which enables higher efficiency for data extraction. However, it is important to note that through caching of data, their quality could decrease.

### 3.3 Data Processing and Integration Flow Modeling Environment

Once data sources and data processing services are bound and deployed, respectively, the Pipes and Filters based models for the data integration and processing flows can be created by domain experts. The registered services and data sources, bound to the resource management platform, appear in the palette of the modeling environment on registration. The elements in the palette can then be used for drag and drop based modeling in the modeling canvas. Once one or more data sources and services are moved to the canvas, they can be arbitrarily interconnected, also using drag and drop as depicted in Fig. 2. The arbitrary interconnection is possible through the uniform data exchange format that is provided through the RMP and that is complied by the services. Once the modeler has completed modeling the data integration and processing flows, he can select his non-functional requirements for the execution in a requirement catalog (cf. Fig. 7). These requirements are described in a non-technical manner so that domain experts can understand them and can select them reasonably. A common example for such a requirement is *security*. By selecting the security requirement, the execution is processed in a way that data is encrypted and the communication channels are secured. The descriptions of the requirements also contain drawbacks that could come with their selection. For example, in case of the security requirement, there is a significant overhead for the encryption, which leads to an increased runtime.

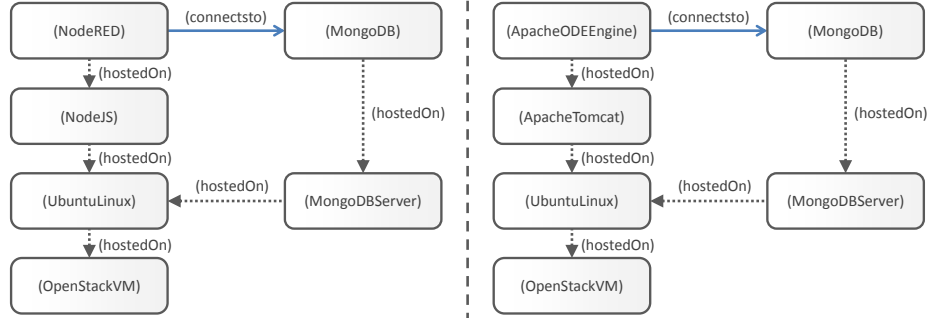


**Figure 4.** Example of a requirement tree containing the top-level requirement, sub-requirements, and implementation components

### 3.4 Transformation and Runtime Selection Component

After modeling, the non executable flow model is transformed into an executable representation. However, the format of this representation depends on the non-functional requirements that were selected by the user in the modeling environment’s requirement catalog. Based on the requirements, a mapping algorithm selects the suitable software components that are able to fulfill them. This algorithm is based on trees that contains two types of nodes: (i) requirement nodes that describe, which requirements have to be fulfilled, whereas a requirement can contain an arbitrary number of sub-requirements, and (ii) implementation component nodes that represent a software component that is necessary to fulfill a requirement. The root node represents a requirement described in the requirement catalog, the implementation component nodes are always leaf nodes of the tree. Figure 4 depicts the structure of these trees. By traversing the tree, the goal is fulfilling all contained sub-requirements and, as a consequence, the top level requirement. Assuming that the tree only contains one leaf node per sub-requirement, this can be done in a straight-forward manner through a depth-first search by selecting all leaf nodes. However, as depicted in Fig. 4 on the right, requirement nodes can have more than one implementation component child node. Due to the fact that the selection of the runtime is conducted fully automatically, there is no possibility to get feedback from the users about which node to choose. Because of that, I use a parameter-based approach based on [16,15]: depending on parameters that are either chosen in the requirement catalog (e.g., preferred cloud provider) or that are extracted from the flow model, the selection of the implementation component can be done automatically. In case it cannot be decided, which node to choose based on the parameters, one node is randomly selected. Due to the fact that the domain expert usually does not have exact preferences regarding the execution and is only interested in that the non-functional requirements are *somehow* fulfilled, a random selection approach seems to be reasonable. Getting user feedback is not an option because the domain experts usually do not have the expertise about technical details or





**Figure 5.** Examples for TOSCA topologies to set up the execution environments. Left: execution in Node-RED, right: execution in an ApacheODE workflow engine

software components. Furthermore, our goal is hiding all technical information from the end user.

Each top-level requirement, as contained and described in the requirement catalog, is represented by a single tree. Consequently, the trees have to be merged to fulfill more than one requirement at once because there could be duplicate implementation component nodes in the trees. Before a traversal of the trees is possible, the trees have to be merged into a single one. To realize this, I use and adapt existing tree merging algorithms because there are already many existing approaches, e.g. [18,3], and developing a new algorithm is out of scope of this thesis. Based on the merged trees, the graph is traversed to find suitable implementation components as described before.

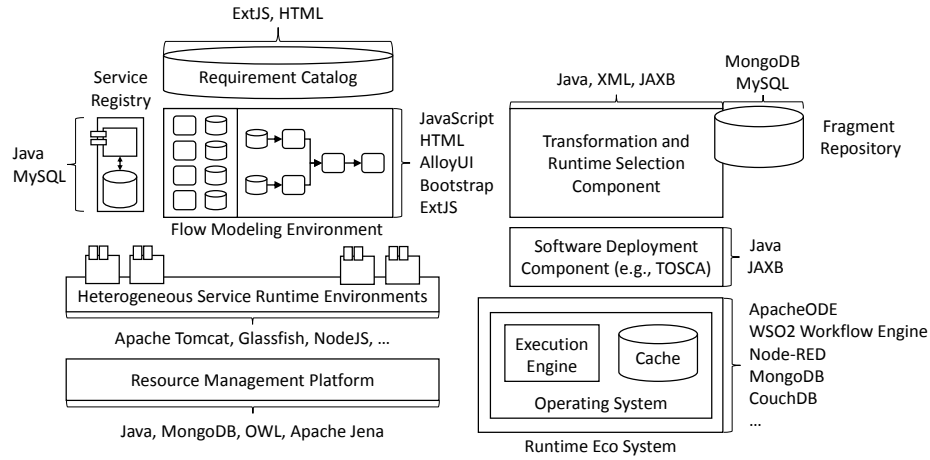
The output of this component is a list of software components that are needed for the execution of the data processing and integration flows dependent on the non-functional requirements of the modelers. This list serves as basis for the deployment of the execution environment, which is conducted by the *Runtime Environment Deployment* component described next.

### 3.5 Runtime Environment Deployment Component

Based on the list provided by the tree traversal, the runtime execution environment can be automatically set up using TOSCA. By doing so, a *TOSCA Topology Template* is created based on the list and is used as input for a corresponding TOSCA runtime. An exemplary topology is depicted in Fig. 5. Details of this step are provided in [10].

### 3.6 Execution Environment

Finally, the data processing and integration flows are deployed into the execution environment, i.e., in the corresponding engine. As described, the execution environment is dependent on the non-functional requirements of the modeler and therefore could be different for each user. On execution, the services are invoked



**Figure 6.** Implementation architecture annotated with the used technologies

in the order as defined in the flow model. By doing so, data is being extracted through the RMP and is being processed through service composition, which is conducted by the corresponding execution engine.

## 4 Prototypical Implementation

I implemented the tool *FlexMash* as a prototype for the concepts of my thesis. This prototype has been demonstrated in 2015 and in 2016 at the International Conference of Web Engineering (ICWE) during the Rapid Mashup Challenge. In both challenge participations, I could achieve winning the second prize. The corresponding publication for the first challenge 2015 can be found in [7], the second publication will be published in January 2017 [9].

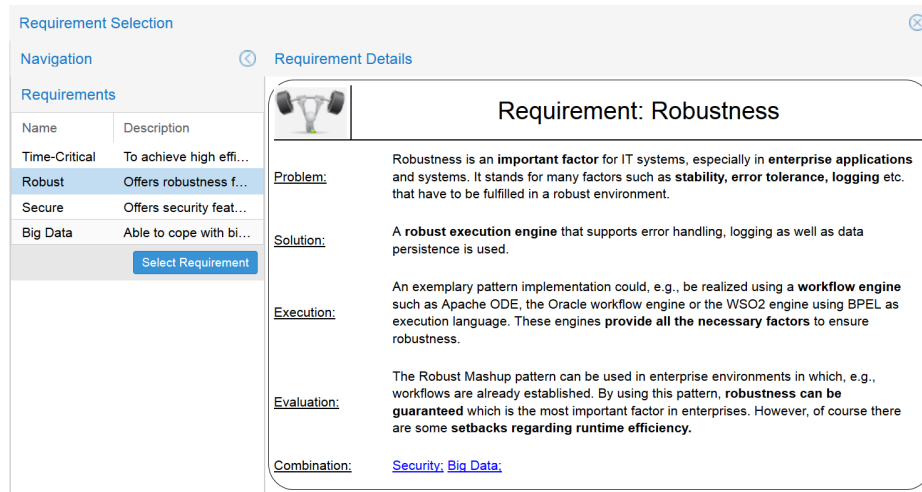
The prototype consists of two parts: (i) the front end containing the flow modeling environment, and (ii) the back end containing the service registry, the service runtime, the resource management platform, the transformation and deployment components, and the execution environment. In the following, I will describe the technologies used for the implementations of these components. The architecture annotated with the corresponding technologies used for implementation is depicted in Fig. 6.

The front end component is implemented in JavaScript exclusively, supported by the frameworks AlloyUI<sup>1</sup>, ExtJS<sup>2</sup>, and Bootstrap<sup>3</sup>. The underlying data model for the data processing and integration flows is based on the JavaScript Object Notation (JSON), which is already natively supported by JavaScript.

<sup>1</sup> <http://alloyui.com/>

<sup>2</sup> <https://www.sencha.com/products/extjs/>

<sup>3</sup> <http://getbootstrap.com/>



**Figure 7.** Exemplary entry in the requirement catalog





















On graphical modeling, the JSON model is automatically synchronized. The modeling interface is depicted in Fig. 2.






The requirement catalog is also implemented in JavaScript and is fully integrated into the modeling environment. The catalog is designed like a wiki, which means that a navigation between different entries is possible through links. Each page in the wiki is a HTML page that is stored in the file system of the server the modeling environment is hosted on. The implementation of the requirement catalog is inspired by the *PatternPedia* as described by Fehling et al. [5]. An example entry in the requirement catalog is depicted in Fig. 7.

The back end is mostly implemented as web application in Java, running in an Apache Tomcat application server container. The service registry component offers a REST interface to create, retrieve, update or delete the services. The services are deployed using OpenTOSCA [1]. The runtimes of the services are hosted on an OpenStack<sup>4</sup> cluster to enable scalability and availability. These could for example be Java application servers such as Apache Tomcat or Glassfish or a NodeJS-based runtime, depending on the service implementation itself. Currently these runtimes have to be set up manually, however, in the future, they could also be set up using TOSCA. This enables setting runtime environments up only if they are needed. If there is for example no NodeJS service, it makes sense to undeploy the NodeJS runtime to save resources.

The transformation of the data processing and integration flows into an executable representation is implemented as a modular Java library. The trees that map requirements to corresponding implementations are based on XML and are traversed using the Java Architecture for XML Binding (JAXB), which is integrated into the Java Developer Toolkit (JDK). The input of the transformation

<sup>4</sup> <https://www.openstack.org/>

Criteria \ Approach	ETL Tools	Data Analytics Tools	Information Integration Tools	FlexMash
Extensibility				
Cloud support				
Graphical modeling / Usability				
Flexible execution				
User interaction during runtime				

 No Support
  Low Support
  Medium Support
  High Support
  Full Support

**Table 1.** Comparison of the introduced approach with others

component consists of the flow model, and a list of requirements selected by the modeler in the catalog. The transformation of the non-executable model into an executable one (e.g., a BPEL workflow) is based on a fragment repository that enables modular building of these executable models based on fragments. The fragment repository is implemented using a MongoDB<sup>5</sup> document store loosely based on the approach described in [17]. The output of the component is a list of runtime components necessary for execution of the data processing and integration flow, and its executable representation.

The software deployment component uses the Topology and Orchestration Specification for Cloud Applications and the implementation provided in [6,2] to create a TOSCA Topology Template based on the component list generated by the transformation component. This Topology Template is used for deployment of the runtime environment's components using OpenTOSCA [1] on an OpenStack-based IBM PureFlex computing cluster, for the deployment of the executable flow model in the corresponding engine, and for the invocation of this model. This can be done using TOSCA build plans that describe operations to be executed on the deployed software components.

## 5 Related Work

In [9], we conducted an extensive survey of related work and discussed how the approach of this thesis differs from other approaches. As a consequence, I will only show a summary of the results of this survey based on the table depicted in Table 1. We compared our prototype FlexMash with other approaches, i.e., ETL (Extract-Transform-Load) tools, data analytics tools, and information integration tools. As shown in this table, FlexMash exceeds other similar approaches mostly regarding its flexibility in the execution of data processing and integration flows, a high extensibility, and the possibility of interaction during runtime, which has not been described in this paper because it will be published in [9].

<sup>5</sup> <https://www.mongodb.com/>

## 6 Summary

In this paper, I introduced my PhD thesis for flexible execution and modeling of data processing and integration flows. By the proposed approach, I enable a means to model data processing scenarios in an easy manner using so called data integration flows, a means to select non-functional requirements for their execution, an automated binding of data sources and data processing services, and a fully automated set up of the execution environment as well as the execution of the flows. By doing so, technical details can be hidden from the end users through a fully automated approach.

In the future, I will mostly work on optimizations and improvements of the introduced approach because the main features have been implemented as described in Sect. 4 and published as shown in the next section.

## 7 Selected Thesis-Related Author Publications

- Automatic Topology Completion of TOSCA-based Cloud Applications. In: Proceedings des CloudCycle14 Workshops auf der 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI) (2014)
- Flexible Modeling and Execution of Data Integration Flows. In Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing (SummerSOC) (2015)
- Extended Techniques for Flexible Modeling and Execution of Data Mashups. In: Proceedings of 4th International Conference on Data Management Technologies and Applications (DATA) (2015)
- FlexMash - A Flexible Data Mashup Tool. In Proceedings of the Rapid Mashup Challenge @International Conference on Web Engineering (ICWE) (2015)
- SitRS - A Situation Recognition Service based on Modeling and Executing Situation Templates. In Proceedings of the 9th Symposium and Summer School On Service-Oriented Computing (SummerSOC) (2015)
- FlexMash – Flexible Data Mashups Based on Pattern-Based Model Transformation. In: Rapid Mashup Development Tools. Springer International Publishing (2016)
- FlexMash 2.0. In Proceedings of the Rapid Mashup Challenge @International Conference on Web Engineering (ICWE) (2016)
- TOSCA4Mashups – Enhanced Method for On-Demand Data Mashup Provisioning. In: Proceedings of the 10th Symposium and Summer School On Service-Oriented Computing (2016)
- Automated Sensor Registration, Binding and Sensor Data Provisioning. In: Proceedings of the CAiSE 2016 Forum at the 28th International Conference on Advanced Information Systems Engineering (2016)
- Dynamic Ontology-based Sensor Binding. In Proceedings of the 20th East-European Conference on Advances in Databases and Information Systems (2016)

## References

1. Binz, T., et al.: OpenTOSCA - A Runtime for TOSCA-based Cloud Applications. In: ICSSOC. pp. 692–695. Springer (December 2013)
2. Breitenbücher, U., Binz, T., Képes, K., Kopp, O., Leymann, F., Wettinger, J.: Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA. In: Proceedings of the IEEE International Conference on Cloud Engineering (IC2E). pp. 87–96. IEEE Computer Society (März 2014), [http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL\\_view.pl?id=INPROC-2014-21&engl=0](http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2014-21&engl=0)
3. Brown, M.R., Tarjan, R.E.: A fast merging algorithm. J. ACM 26(2), 211–226 (Apr 1979), <http://doi.acm.org/10.1145/322123.322127>
4. Devlin, B.: Data Warehouse: From Architecture to Implementation. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1996)
5. Fehling, C., Barzen, J., Falkenthal, M., Leymann, F.: PatternPedia—Collaborative Pattern Identification and Authoring. In: Proceedings of Pursuit of Pattern Languages for Societal Change-Preparatory Workshop (2014)
6. Hirmer, P., Breitenbücher, U., Binz, T., Leymann, F.: Automatic Topology Completion of TOSCA-based Cloud Applications. In: Proceedings des CloudCycle14 Workshops auf der 44. Jahrestagung der Gesellschaft für Informatik e.V. (GI) (2014)
7. Hirmer, P., Mitschang, B.: FlexMash - Flexible Data Mashups Based on Pattern-Based Model Transformation. In: Rapid Mashup Development Tools. Springer International Publishing (2016)
8. Hirmer, P., Mitschang, B.: TOSCA4Mashups – Enhanced Method for On-Demand Data Mashup Provisioning. In: Proceedings of the 10th Symposium and Summer School On Service-Oriented Computing (2016)
9. Hirmer, P., Mitschang, B.: FlexMash 2.0 – Flexible Modeling and Execution of Data Mashups. In: Rapid Mashup Development Tools. Springer International Publishing (to be published 2017)
10. Hirmer, P., Wieland, M., Breitenbücher, U., Mitschang, B.: Automated Sensor Registration, Binding and Sensor Data Provisioning. In: Proceedings of the CAiSE 2016 Forum at the 28th International Conference on Advanced Information Systems Engineering (2016)
11. Hirmer, P., Wieland, M., Breitenbücher, U., Mitschang, B.: Dynamic ontology-based sensor binding. In: Proceedings of the 20th East-European Conference on Advances in Databases and Information Systems (2016)
12. Hirmer, P., et al.: Extended Techniques for Flexible Modeling and Execution of Data Mashups. In: Proceedings of 4th International Conference on Data Management Technologies and Applications (DATA) (2015)
13. Kassner, L., Mitschang, B.: MaXcept—Decision Support in Exception Handling through Unstructured Data Integration in the Production Context. In: Proceedings of the 48th Hawaii International Conference on System Sciences (2015)
14. Meunier, R.: The pipes and filters architecture. In: Pattern languages of program design (1995)
15. Reimann, P., et al.: Data Patterns to Alleviate the Design of Scientific Work Flows Exemplified by a Bone Simulation. In: Proceedings of the 26th International Conference on Scientific and Statistical Database Management (2014)
16. Reimann, P., et al.: A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design. In: Proceedings of the 22nd International Conference on Cooperative Information Systems (CoopIS 2014), Amantea, Italy (2014)

17. Schumm, D., Dentsas, D., Hahn, M., Karastoyanova, D., Leymann, F., Sonntag, M.: Web service composition reuse through shared process fragment libraries. In: Proceedings of the 12th International Conference on Web Engineering. pp. 498–501. ICWE'12, Springer-Verlag, Berlin, Heidelberg (2012), [http://dx.doi.org/10.1007/978-3-642-31753-8\\_53](http://dx.doi.org/10.1007/978-3-642-31753-8_53)
18. Sun, X., Wang, R., Salzberg, B., Zou, C.: Online b-tree merging. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. pp. 335–346. SIGMOD '05, ACM, New York, NY, USA (2005), <http://doi.acm.org/10.1145/1066157.1066196>
19. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)

All links were last followed on August 19, 2016.

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

Marigianna Skouradaki<sup>1</sup>, Tayyaba Azad, Uwe Breitenbücher<sup>1</sup>,  
Oliver Kopp<sup>2</sup>, and Frank Leymann<sup>1</sup>

<sup>1</sup>IAAS, <sup>2</sup>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 design 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 Management 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 <sup>1</sup> 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

---

<sup>1</sup> <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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 2015 also 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]. Therefore, 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 limited 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 ActiveVOS 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. Although 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 prototypical 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 response 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<sup>2</sup>. 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] assesses the performance of the Cape Clear WS-BPEL engine running on Intel<sup>®</sup> 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.

<sup>2</sup> [http://www.activevos.com/content/developers/education/sample\\_active\\_bpel\\_admin\\_api/doc/index.html](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 performance 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 response 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 referred to as *DSS4MiddlewarePBenchmarking*. The following list describes the most important recognized

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

- FR<sub>1</sub> Visualisation of the Decision Support System:* Extract data from the *DSS4MiddlewarePBenchmarking* in a human recognisable manner. In order to provide support to the decision-making for performance benchmarking.
- FR<sub>2</sub> 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<sub>3</sub> 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<sub>4</sub> 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<sub>5</sub> Reliability 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<sub>1</sub> 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<sub>2</sub> 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<sub>3</sub> 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<sub>4</sub> Web-based development:* Portability, cross-platform support and a user-friendly interface should be supported through a web-based solution.
- NFR<sub>5</sub> Easy installation:* The configuration and installation of the software should be rapid and easy.
- NFR<sub>6</sub> 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.

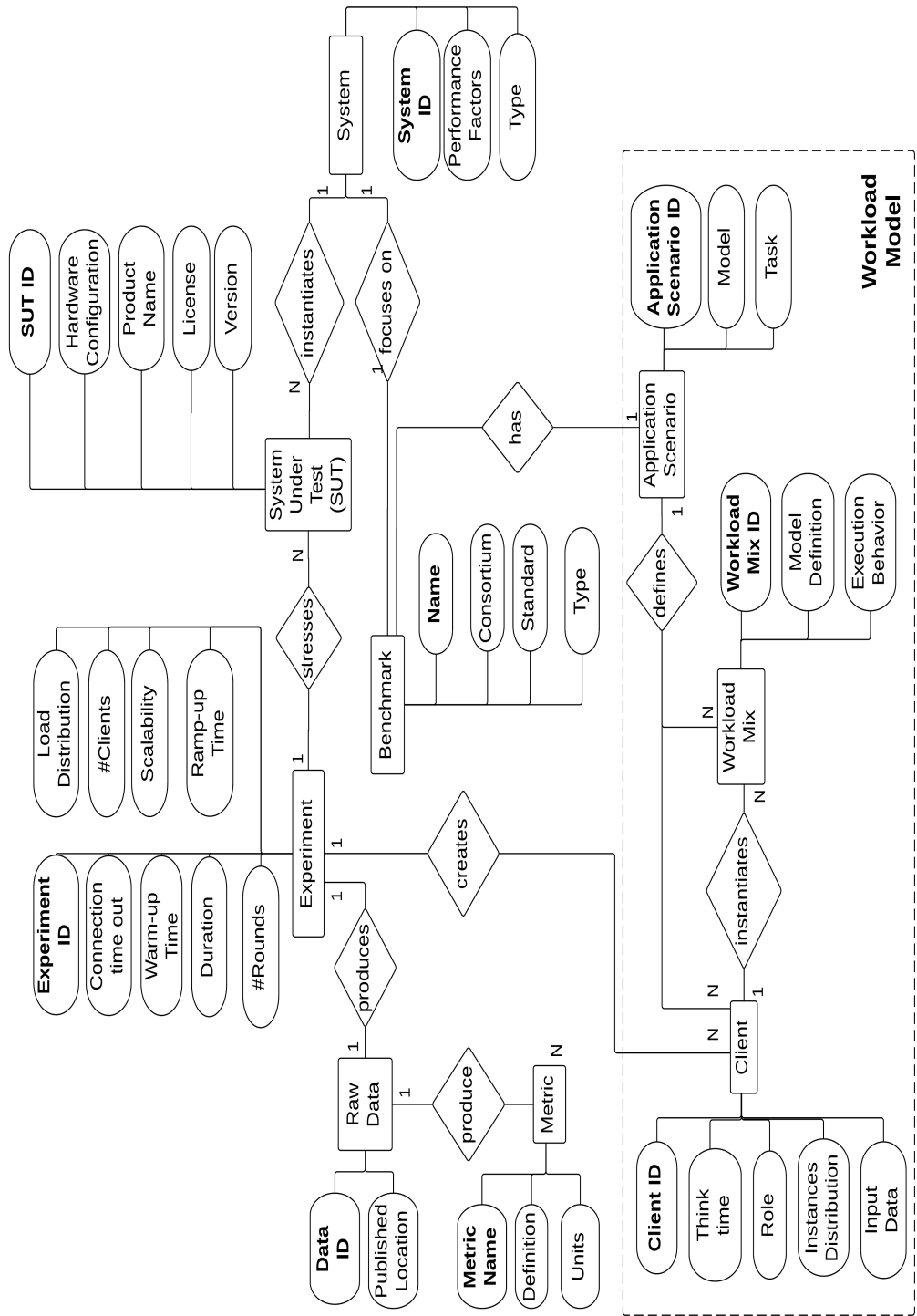
The *system* refers to the type of system that the user is interested in retrieving the data for. The system contains a type which refers to the type of the benchmarked system (i.e., Java Server, Workflow Management System, etc.). The various *benchmarks* focus on addressing the performance of different 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<sup>®</sup> JMS 2007 benchmark 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 referred to as *System Under Test (SUT)*. The benchmark related attributes of a SUT are the hardware configuration of its comprising artifacts, the enterprise name of the product (i.e., Camunda<sup>3</sup>, Apache Active MQ<sup>4</sup>, 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 hardware 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

<sup>3</sup> <https://camunda.org>

<sup>4</sup> <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<sup>®</sup> JMS 2007 benchmark four different participants were involved in the application scenario [22]. The Company Headquarters 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,

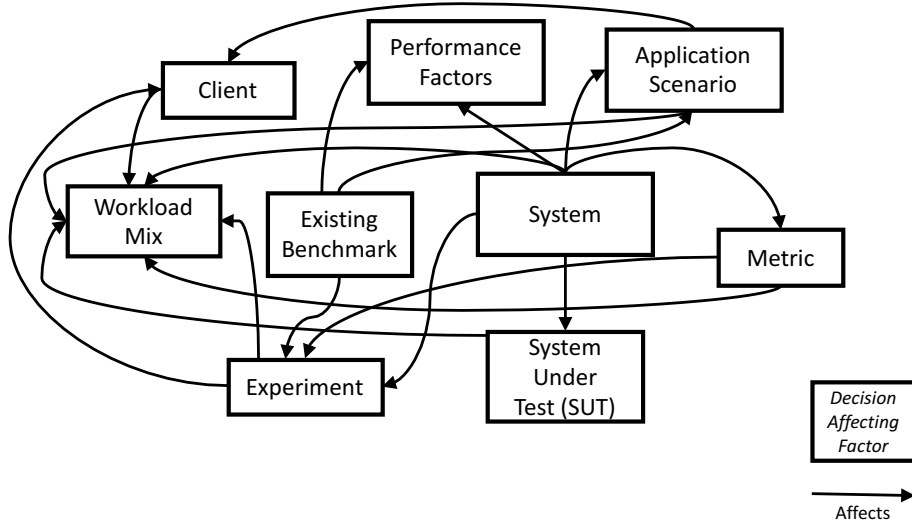


in the SPEC<sup>®</sup> JMS 2007 [22] benchmark, a natural way to scale the workload was provided. To be precise, two different types of scaling were provided, the horizontal approach supported scaling in terms of increasing the number of supermarkets, while in the vertical approach the number of products 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 beginning 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 measurements 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<sup>®</sup> 2015 benchmark is defined as business operations per second [7] and measures SPECjbb bops units. Another example is the throughput 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 affects 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

The screenshot shows a web application for **DSS4MiddlewarePBenchmarking**. On the left is a sidebar with a navigation menu including: Application Scenario, Benchmark, Hardware, Message Ratio, Messages, Messaging Style, Metric, Middleware, Performance, Roles, Scalability, Transactions, Transaction Style, and Workload. Below the menu is a 'Sign Out' link and an 'Admin' button labeled **(FR<sub>2</sub>)**.

The main content area is divided into two columns. The left column is titled 'System I am interested in:' and contains radio buttons for: Any, Java Server, Message Oriented Middleware (selected), Database Management System (DBMS), and Workflow Management System (WfMS). Below this is a dashed box labeled **Visualization of the DSS (FR<sub>1</sub>)**. The right column is titled 'Feature I'm looking for' and contains radio buttons for: Application Scenario, Existing Benchmark, System Under Test, Workload Mix, Client, Raw Data, and Metric (selected). Below this is a 'Filter down to:' section with checkboxes for 'Definition' and 'Units', both of which are checked. To the right of the filters is a dashed box labeled **Dynamic Querying (FR<sub>3</sub>)**.

At the bottom of the main content area is a 'SUBMIT' button and a 'reset values' link. Below this is a 'Results:' section containing text about metric definitions and units. To the right of the results is a dashed box labeled **Reporting Support (FR<sub>4</sub>)**.

**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<sup>5</sup>, Express.js<sup>6</sup>, AngularJS<sup>7</sup>, and Node.js<sup>8</sup>. This set of technologies are combined together to the open-source

<sup>5</sup> <https://www.mongodb.com>

<sup>6</sup> <http://expressjs.com>

<sup>7</sup> <https://angularjs.org>

<sup>8</sup> <https://nodejs.org/en/>

**DSS4MiddlewarePBenchmarking**

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

**SUBMIT** [reset values](#)

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

**Fig. 4.** Validation of use case 1

MEAN<sup>9</sup> 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 interact 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<sup>10</sup>. 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 administration 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

<sup>9</sup> <http://mean.io>

<sup>10</sup> <https://bitbucket.org/tayyabaazad/dss4middlewarepbenchmarking/>

The screenshot displays the DSS 4 Middleware Benchmarking web application. On the left is a vertical navigation menu with the following items: Application Scenario, Benchmark, Hardware, Message Ratio, Messages, Messaging Style, Metric, Middleware, Performance, Roles, Scalability, Transactions, Transaction Style, and Workload. Below the menu is a 'Sign Out' link. The main content area is divided into two columns. The left column, titled 'System I am interested in:', contains radio buttons for 'Any', 'Java Server', 'Message Oriented Middleware', 'Database Management System (DBMS)' (which is selected), and 'Workflow Management System (WfMS)'. The right column, titled 'Feature I'm looking for', contains radio buttons for 'Application Scenario', 'Existing Benchmark', 'System Under Test', 'Workload Mix', 'Client', 'Raw Data', and 'Metric' (which is selected). Below these columns is a 'Filter down to:' section with a checked box for 'Definition' and an unchecked box for 'Units'. At the bottom of the main area is a 'SUBMIT' button and a 'reset values' link. Below the main area, the 'Results:' section shows the text: '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<sup>®</sup> 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

The screenshot displays the DSS4Middleware Benchmarking web application. On the left is a navigation menu with links: Application Scenario, Benchmark, Hardware, Message Ratio, Messages, Messaging Style, Metric, Middleware, Performance, Roles, Scalability, Transactions, Transaction Style, and Workload. Below the menu is a 'Sign Out' link. The main content area is divided into two columns. The left column, titled 'System I am interested in:', contains radio buttons for 'Any:', 'Java Server', 'Message Oriented Middleware', 'Database Management System (DBMS)', and 'Workflow Management System (WfMS)', with the last option selected. The right column, titled 'Feature I'm looking for:', contains radio buttons for 'Application Scenario', 'Existing Benchmark', 'System Under Test', 'Workload Mix' (selected), and 'Metric'. Below these is a 'Filter down to:' section with checkboxes for 'Definition' and 'Execution Behavior' (checked). At the bottom of the main area are 'SUBMIT' and 'reset values' buttons. A 'Results:' section at the bottom states: '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 where 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 combining 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](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: Bellahsene, 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., Koziol, 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.

# Adaptable Digital Enterprise Architecture with Microservices

Justus Bogner<sup>1,2</sup> and Alfred Zimmermann<sup>1</sup>

Herman Hollerith Center, Boeblingen, Germany<sup>1</sup>

Hewlett Packard Enterprise, Boeblingen, Germany<sup>2</sup>

justus.bogner@hpe.com, alfred.zimmermann@reutlingen-university.de

## 1 Introduction and Motivation

The fast moving process of digitization<sup>1</sup> demands flexibility in order to adapt to rapidly changing business requirements and newly emerging business opportunities. New features have to be developed and deployed to the production environment a lot faster. To be able to cope with this increased velocity and pressure, a lot of software developing companies have switched to a Microservice Architecture (MSA) approach. Applications built this way consist of several fine-grained and heterogeneous services that are independently scalable and deployable. However, the technological and business architectural impacts of microservices based applications directly affect their integration into the digital enterprise architecture. As a consequence, traditional Enterprise Architecture Management (EAM) approaches are not able to handle the extreme distribution, diversity, and volatility of micro-granular systems and services. We are therefore researching mechanisms for dynamically integrating large amounts of microservices into an adaptable digital enterprise architecture.

## 2 Microservices and Adaptable Enterprise Architecture

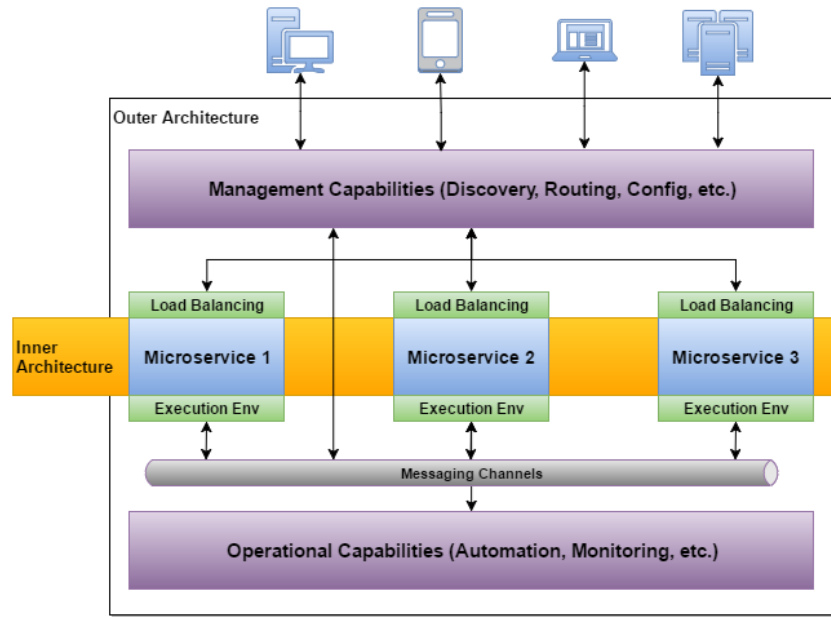
The term microservices became popular around 2013 and refers to a fine-grained style of service-oriented architecture (SOA). James Lewis defines a Microservice Architecture as an approach for building a single application as a set of small independent services.<sup>2</sup> Each of these services runs in its own process and communicates with lightweight mechanisms. Microservices are built around business capabilities, are independently deployable, and may utilize very different technologies. As opposed to big monolithic applications, a single microservice tries to represent a unit of functionality that is as small and coherent as possible. Using MSAs, organizations can expand agility and flexibility for business and IT systems. However, microservices also come with the

---

<sup>1</sup> Westerman, G. et al: Leading Digital: Turning Technology Into Business Transformation. Harvard Business Press, 2014

<sup>2</sup> Newman, S.: Building Microservices: Designing Fine-Grained Systems, O'Reilly, 2015

need for a strong DevOps culture to handle the increased distribution level and deployment frequency. Moreover, while each single microservice is of low complexity, the overall complexity of the system has not been reduced. It has moved from the inner architecture to the outer architecture (see Fig. 1).



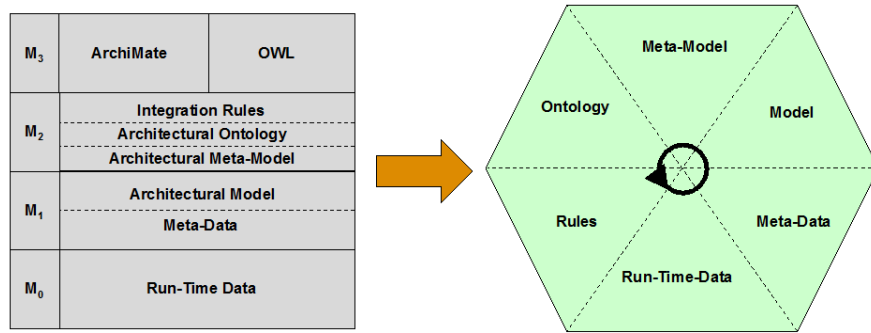
**Fig. 1.** Inner and Outer Architecture of Microservices<sup>3</sup>

In the context of EAM, challenges with microservices are mostly concerned with heterogeneity, distribution, and volatility. To integrate a huge amount of dynamically growing architectural descriptions of very different microservices into a consistent enterprise architecture is a considerable challenge. In order to tackle this problem, we are currently working on the formalization of small-decentralized mini-metamodels, models, and data of architectural microservice descriptions based on the Meta Object Facility standard<sup>4</sup> (EA-Mini-Descriptions, see Fig. 2). M0 as well as M1 serve as local layers to a single microservice (*cell* metaphor) with M0 representing operational run-time or monitoring data and M1 providing metadata (e.g. purpose, API endpoints, or usage costs) and also its inner architectural model (e.g. components or communication channels). Using these two as a foundation, M2 works as a global metamodel layer with information for several communicating microservices (*body* metaphor, combining several *cells*). It holds architectural metamodels and ontologies while providing important integration rules for semi-automatic integration. These metamodels are then included

<sup>3</sup> Based on Olliffe, G.: Microservices: Building Services with the Guts on the Outside, 2015. Retrieved March 18, 2016 from <http://blogs.gartner.com/gary-olliffe/2015/01/30/microservices-guts-on-the-outside>

<sup>4</sup> OMG, 2011, OMG Meta Object Facility (MOF) Core Specification, Version 2.5

into the holistic and dynamically growing EA metamodel from the composition of EA-Mini-Descriptions. Finally, layer M3 defines the semantic representations and languages that are used for modeling and representing these adaptable enterprise architecture metamodels.



**Fig. 2.** Structure of EA-Mini-Descriptions

As a next step, the presented EA-Mini-Descriptions are used with the Enterprise Services Architecture Model Integration (ESAMI) method<sup>5</sup> to perform correlation analysis, which provides an instrument for systematic integration. The iterative approach is based on special correlation matrices handled by a manual process to identify similarities between analyzed model elements. The chosen elements are then integrated according to their most valuable contribution towards a holistic reference model. This continuous model refinement allows to integrate even extremely heterogeneous microservices that may in fact not share a complete metamodel.

### 3 Conclusion and Future Work

The presented architectural properties of microservices demand advanced Enterprise Architecture methodologies for the integration of structures with a micro-granular architecture into an overall adaptable EA. Our EA-Mini-Descriptions can serve as flexible metamodels for microservices that can be combined into larger entities. Through a manual correlation-based model analysis and integration approach, we presented means to merge these microservices into a holistic, but dynamically adjusting reference architecture. Future research may include the automatic machine-supported creation of our EA-Mini-Descriptions (at least partially). Similarly, it may be of interest to support the manual integration decision by automated systems, e.g. via mathematical comparisons (similarity, Euclidean distance), semantic integration rules, or data analytics and data mining techniques. These methods can significantly ease associated manual efforts and reduce the rate of architectural errors in traditional EA models and approaches.

<sup>5</sup> Zimmermann, A. et al: Towards an integrated service-oriented reference enterprise architecture, Proc. 2013 Int. Work. Ecosyst. Archit. - WEA 2013, pp. 26–30, 2013