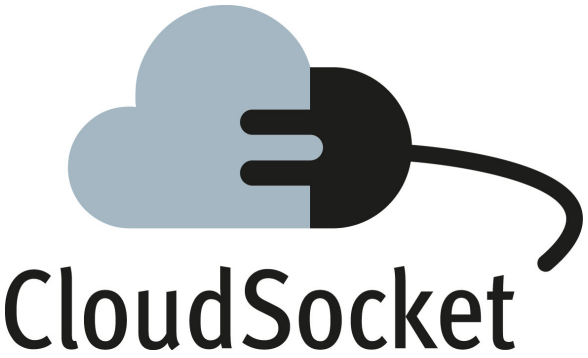

BPAAS MONITORING AND EVALUATION BLUEPRINTS

D3.5

Editor Name	Kyriakos Kritikos (FORTH)
Submission Date	December 31, 2016
Version	1.0
State	Final
Confidentially Level	PU

Co-funded by the Horizon 2020
Framework Programme of the European Union

EXECUTIVE SUMMARY

Business processes (BP), irrespectively on their deployment means, might be well-designed by considering particular goals and might be validated based on different mechanisms. They might also be allocated properly by considering allocation rules as well as functional and non-functional requirements at the technical level. However, no matter what techniques and methods are involved in the BP design and allocation, the fact that these techniques are exploited by humans with their own individual perceptions on the domain and different expertise and capabilities cannot be neglected. In this sense, the respective BP design and allocation products will not be perfect. They might have neglected the modelling of particular situations or might have under-estimated some technical capabilities. In this sense, there is a need to have appropriate means for identifying potential issues with respect to the design, deployment and execution of a BP as well as potential points for further improvements such that then the respective adjustments can be made to improve and optimise the BP under current investigation, thus completing the BP lifecycle.

The latter task is the main subject of the evaluation phase in the BP management lifecycle [36]. It can involve performing various types of analysis in order to derive the respective business intelligence knowledge needed to improve a BP. Such types usually span techniques and methods for Key Performance Indicator (KPI) evaluation and for process mining over (process) execution logs. Such techniques and methods can then give rise to a respective BP evaluation framework which could stand as a standalone product or as part of an overall BP Management System (BPMS).

In the context of the CloudSocket project and Task T3.3, the main goal is the production of a BP Evaluation Environment giving support to the evaluation of cloud-based BPs, i.e., Business Processes as a Services (BPaaSes). It could rather seem that such an environment could just be realised as a collection of existing tools over different types of analysis. However, this is not the actual case. As indicated in the main content of this deliverable, migration of BPs in the cloud raises various issues which generate major challenges on the BP evaluation process. Moreover, each tool can have its own dependencies over the available information, leading to the need to create a sophisticated mechanism for information retrieval and linking over disparate information sources. In addition, extra business intelligence knowledge might need to be derived which is not yet supported in the existing BP Evaluation frameworks. Besides, existing techniques and methods seem to exhibit accuracy issues which need to be resolved by the use of semantics.

Based on the above analysis, the goal of T3.3 is to perform research and development tasks which will focus on extending existing techniques and methods in BP evaluation as well as propose new ones. It also involves the exploitation of semantic information as well as the capability to semantically enhance existing information. A first result towards pursuing this main goal of T3.3 has been incarnated in the content of this deliverable, D3.5, in the form of BPaaS monitoring and evaluation blueprints, i.e., ideas, concepts and work in process and prototypical implementations of extensions of existing or new evaluation algorithms, methods and tools. These blueprints comprise: (a) the *Information Harvesting & Linking* blueprint dedicated to the extraction of information from different environments and components of the CloudSocket prototype, its semantical enhancement and linking and subsequent storage in a semantic Knowledge Base (Semantic KB); (b) the *KPI Analysis* blueprint which focuses on the evaluation of KPIs and their drill-down to find root cases of problems via exploiting a semantic approach; (c) the *Best Deployment Discovery* blueprint which enables the discovery of best deployment for BPaaS, even when an execution history is not available for them; (d) the *Event Pattern Detection* blueprint which is able to discover sets of events which lead to the violation of KPIs as well as map these sets into adaptation strategies in order to complete the specification of BPaaS adaptation rules; (e) the *Process Mining* blueprint which can execute prominent process mining algorithms as well as semantically enhance them. Apart from these five blueprints, which actually exhibit the main harvesting and analysis functionality, two additional blueprints have been proposed which focus on the way the knowledge in the Semantic KB can be structured to support the

analysis functionality exhibited by the rest of the blueprints: (f) the *Evaluation Ontology* is able to capture dependency/deployment models for BPaaSes while (g) the OWL-Q KPI Extension is able to sufficiently model KPIs and the respective metrics involved.

Some of the above blueprints might not be yet near a final production state. To this end, the next and final deliverable in T3.3, named as D3.6 - BPaaS Monitoring and Evaluation Prototypes, will focus on further analysing the research and development efforts dedicated to these blueprints. For all blueprints, possible future work directions have also been identified which could be followed in the next period. In case they are do followed, they will also be reported on D3.6.

Finally, we conclude by indicating that: (a) the BPaaS Evaluation Environment, comprising all the aforementioned blueprints, ambitions to produce business intelligence knowledge which can really grab the attention of brokers; (b) D3.5 has paved the way via which the final prototypes could be adopted and incorporated in the final implementation of the CloudSocket platform. In particular, for the latter, it has been designated that a parallel approach is being followed in which mature features of prototypes / blueprints are immediately adopted once they are delivered, such that the work in T3.3 and T4.2 goes hand by hand. It should also be highlighted that the selection of features is the one that does matter here and is being promoted, as the structuring of the modules reflects the architecture of the BPaaS Evaluation Environment in the CloudSocket platform implementation.

PROJECT CONTEXT

Workpackage	WP3: Business Process as a Service Research
Task	T3.3: BPaaS Evaluation Environment Research
Dependencies	Input to WP4

Contributors and Reviewers

Contributors	Reviewers
Kyriakos Kritikos (FORTH), Chrystostomos Zeginis (FORTH), Daniel Seybold (UULM), Wilfrid Utz (BOC)	Wilfrid Utz (BOC), Yongzheng Liang (BWCON), Simone Naldini (MATHEMA)

Approved by: Stefan Wesner (UULM) as WP 3 Leader

Version History

Version	Date	Authors	Sections Affected
0.1	November 01, 2016	Kyriakos Kritikos (FORTH)	Initial version, TOC
0.2	November 11, 2016	Kyriakos Kritikos (FORTH)	First full draft version ready for review
0.21	December 21, 2016	Yongzheng Liang (BWCON), Wilfrid Utz (BOC), Simone Naldini (MATHEMA)	Reviewer feedback
1.0	December 23, 2016	Kyriakos Kritikos (FORTH)	Final version

Copyright Statement – Restricted Content

This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content.

This is a restricted deliverable that is provided to the community under the license Attribution-No Derivative Works 3.0 Unported defined by creative commons http://creativecommons.org

You are free:

	to share within the restricted community — to copy, distribute and transmit the work within the restricted community
Under the following conditions:	
	Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
	No Derivative Works — You may not alter, transform, or build upon this work.

With the understanding that:

Waiver — Any of the above conditions can be waived if you get permission from the copyright holder.

Other Rights — In no way are any of the following rights affected by the license:

- Your fair dealing or fair use rights;
- The author's moral rights;
- Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work.
This is a human-readable summary of the Legal Code available online at:

http://creativecommons.org/licenses/by-nd/3.0/

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION AND PROBLEM STATEMENT

1.1 Introduction

This deliverable introduces blueprints, which are related to the monitoring and evaluation of BPaaS. Such blueprints can be exploited so as to perform various types of BPaaS analysis and lead to the production of different forms of business intelligence knowledge. Such knowledge can then be exploited to close the BPaaS lifecycle and enable to optimise the BPaaS under consideration.

We need to highlight that blueprints here mean different kinds of proposals, which are not yet on an appropriate maturity level, including sketches of ideas and first but incomplete realisations of such ideas. This actually signifies that the state and type of blueprints described in this deliverable may vary in the end. However, the major goal is to be able to evolve at least the most critical and promising blueprints until the end of WP3 in order to enable their adoption by the main CloudSocket prototype implementation in WP4.

1.2 Project Context

This is the very first deliverable of Task T3.3 that deals with the monitoring and evaluation of BPaaS. This will be followed by the next and final deliverable, numbered as D3.6 and named as "BPaaS Monitoring and Evaluation Prototypes", which will analyse the actual prototypes that will be produced mapping to the blueprints presented in this document.

The scope of monitoring should be clarified here in order to have a clear thematic distinction between the series of deliverables of T3.3 and those involved in T3.2. BPaaS monitoring in the context of T3.3 deals with the computation of KPI metrics, i.e., high-level metrics which are computed on a coarse-grained timely manner for the sake of the broker and cover mainly business aspects. For instance, KPI metrics in this case will be evaluated in periods of days, weeks, months or years. Moreover, this KPI metric computation is performed in an offline manner, even when no instances of a BPaaS are currently running. On the other hand, T3.2 deals with the monitoring of metrics at lower levels of abstraction, thus covering mainly technical aspects, which are computed in fine-grained time periods. The goal is here to monitor the BPaaS while is being executed in order to react in a timely manner when a respective problematic situation occurs.

T3.3 is however related to both T3.2 and T3.1 in the sense that respective results, information artifacts and functionality from the latter tasks need to be exploited in order to deliver accordingly the respective blueprints of T3.3. Chapter 4 clearly explicates the current information artifacts that need to be in place by the latter two tasks, while Chapters 2 and 3 explain how they are stuctured when gathered and semantically linked as well as the main mechanisms that are exploited in order to retrieve them, respectively, which of course create some functional dependencies over the respective blueprints of the T3.1 and T3.2 tasks. To enable the reader to have a better understanding on this, we can explicate that from T3.1, the main focus is on information regarding both functional and non-functional aspects in BPaaS description, while the main focus from T3.2 is over BPaaS allocation plus monitoring information.

This dependency between the current task and tasks T3.1 and T3.2 also justifies the time period of T3.3 operation which has been started in the middle of the T3.2 period and after T3.1 has already been finished.

By considering the actual functionality promised by the implementation of the BPaaS Evaluation Environment in WP4, the current set of blueprints make a perfect match with the respective capabilities expected. In fact, we can clearly indicate that the developments in T3.3 and T4.2 go hand by hand, also due to the shortage in time until the end of the research and development tasks in this project, where blueprints with appropriate maturity level are

immediately being adopted by T4.2. This has been actually witnessed by the first blueprint dealing with KPI analysis for which one of the two major capabilities (i.e., KPI analysis) has already been delivered. This is also in slight contrast with other tasks, e.g., T3.2, where the situation is actually mixed having blueprints that have already been adopted as well as other blueprints that are still in development and might be adopted in the end as interesting add-ons to the existing CloudSocket prototype functionality implemented in the context of particular environments or components.

Of course, we also need to highlight that while (research) blueprints might be adopted by WP4, this does not necessarily mean that all individual blueprint capabilities may be adopted. This actually depends on the current maturity level of the implementation. By considering again the current blueprint that has been adopted, the KPI Analysis one, this blueprint comprises two main capabilities, namely KPI evaluation and KPI drill-down, where only the first is mature enough and has been selected.

1.3 Research Problem

The main research problem attempted to be addressed by the current research in T3.3 is the devising of analysis techniques that can assist in the monitoring and evaluation of a BPaaS with the main aim to find potential places for BPaaS correction and improvements. While this problem has been well researched in the literature in the context of business processes (BPs), the offering of BPaaSes, i.e., business processes over the cloud, brings about new challenges that have not been anticipated before. For instance, public clouds are not controllable and are external, which leads to challenges on how to properly monitor a BPaaS and which information sources to exploit for such monitoring. Moreover, in the past, BP evaluation has not considered cross-layer issues which need to be tackled not only due to the layering that service-orientation brings about but also due to dependencies involved in these layers. This means that both monitoring and evaluation need to be performed in bottom-up and top-down manner, respectively, to enable the appropriate propagation of measurements to the higher-levels by also catering for covering measurability gaps as well as performing root-cause analysis from high-level business KPI violations to low-level technical ones.

The well-known business to IT alignment problem is also transferred in the context of BP and BPaaS evaluation. KPIs can be considered as indicators concerning the performance of a BP. Such KPIs are initially captured by humans taking the role of a business expert in a textual form. This form is however not machine-processable and interpretable. As such, it has to be transformed. This transformation is performed manually by an process performance / evaluation expert. In D3.2 [33] an initial proposal for transforming high-level business goals, like (semantically-annotated) KPIs, to low-level ones, in the form of technical non-functional requirements has been provided. This transformation, however, is not complete and might not cover all possible aspects. As a result, the intervention of the evaluation expert is still needed. In our view, what is still missing to facilitate the correct determination of the machine-processable KPI is the capability to experiment with different KPI metric formulas as well as to enable the expert to exploit different information sources, both external and internal, to cover the aforementioned required aspects. The mechanisms for exploiting different information sources are missing from both the BPaaS evaluation systems as well as from the respective KPI modelling languages. Moreover, the KPI metric formula exploration is not allowed in other systems or is not facilitated via the use of appropriate languages. In this way, apart from employing information harvesting mechanisms, we also advocate the use of a kind of mathematical language that could be used by the expert to specify the KPI metric formula. As such, through model transformation, the respective description will be mapped to a formal KPI specification that could then be assessed by the corresponding BPaaS evaluation system. By employing such a mathematical language, the modeller focuses on the main formula specification and is not required to provide additional technical details that might overwhem him / her and for which a deep exploration of the information space needs to be provided. The system should be able to complete these technical details automatically during the model transformation.

Apart from focusing on KPI monitoring and evaluation, other aspects are also worth to be investigated and will make the life of a BPaaS broker easier. In particular, the broker needs to be guided in the discovery of appropriate allocation and re-configuration points for BPaaS while he / she might also need to have a closer and deeper look over the current structure of a particular BP. Concerning allocation and re-configuration point discovery, we advocate that there is a need for new analysis techniques, which are able to confront two major problems: (a) find the best deployments for a BPaaS; (b) find event patterns which lead to the violation of KPIs / Service Level Objectives (SLOs) or even adaptation rules which are able to resolve these violations. The first kind of discovery would certainly enable the broker to investigate more optimised deployment capabilities for an existing BPaaS. It might also enable proposing deployment capabilities for a new BPaaS based on the similarity that this BPaaS has with other existing BPaaSes. The second kind of discovery enables the derivation of rules which can drive the adaptation of the BPaaS during runtime when problematic situations occur.

Apart from the fact that BPaaS allocation occurs at different levels and many different kinds of events might occur across all layers of the BPaaS stack (BPaaS, WfaaS, SaaS, PaaS & IaaS), both kinds of discovery require extensive deployment and performance information which might not be available from a single information source. By following the architecture paradigm of the CloudSocket prototype implementation, different components are involved in the deployment specification, in the deployment enforcement and in the production of monitoring information. To this end, information from all of these sources needs to be collected, linked and appropriately stored to enable performing the aforementioned analysis tasks.

While semantics and ontologies are considered in some cases, especially due to their main advantages which include more formal description and enabling of advanced reasoning tasks, we advocate that they need to be actually taken into account in all types of BPaaS analysis required by the broker. This will also greatly enhance the accuracy of the knowledge derived from these analysis tasks. Some directions over the exploitation of the semantics can be seen in the literature, especially for KPI analysis and process mining [3]. However, this is not done in a holistic way by also catering for the appropriate (semantic) linkage of the corresponding information gathered in the context of all analysis tasks that might need to be performed by a BPaaS Evaluation Environment.

Based on the above analysis, the actual BPaaS Evaluation Environment (see Section 3.2 for its architecture) takes an overall semantic approach which is incarnated in the form of a *Semantic Knowledge Base* (Semantic KB) able to store appropriately linked and described BPaaS deployment and monitoring information. Concerning modelling, two language blueprints connected to each other enable the coverage of the two major information aspects, namely deployment and monitoring. The Evaluation ontology (see Section 2.2) covers the deployment aspect by also adopting the type-instance pattern, while the OWL-Q [40] KPI extension (see Section 2.3) covers the monitoring aspect with a specific hook over which monitored object is associated to the deployment aspect. The retrieval, the semantic enrichment and linking of the two information kinds is performed by the Information Harvesting and Linking (Harvester in short) blueprint (see Section 3.3) which employs the same mechanism to obtain information from different sources as well as already existing annotations along with the respective concept relationships in the two ontologies to perform the linking.

The main kinds of analysis offered by the BPaaS Evaluation Environment map to respective individual blueprints. The *KPI Analysis* blueprint (see Section 3.4) deals with the realisation of KPI evaluation and drill-down capabilities through the exploitation of a transformation mechanism which maps the definition of KPI metrics to SPARQL[1] queries that are posed over the *Semantic KB*.

The *Best Deployment Discovery* blueprint (see Section 3.5.2) aims at deriving best deployments for BPaaSes by exploiting the deployment and execution history of these BPaaSes or of other BPaaSes that match them. This derivation is enabled via the introduction of a rule-based approach which relies on two KB kinds: the semantic KB

[1] https://www.w3.org/TR/rdf-sparql-query/

and a normal fact-based KB. The second one is used for retrieving only appropriate information from the first one and includes rules that drive the respective derivation. The rationale for this hybrid approach is based on the fact that the current rule support in semantic KBs is limited.

The *Event Pattern Detection* blueprint (see Section 3.6) aims to discover those event patterns which lead to the violation of KPIs and SLOs by employing a logic-based data mining approach. These event patterns are then mapped to actual adaptation rules for BPaaS by following a semi-automatic event pattern to adaptation strategy method. This method relies on the (expert-supplied) knowledge of which single events are mapped to which adaptation actions as well as the history profile of adaptation action and strategies that have been executed in the past in the context of the current BPaaS under examination.

Finally, the *Process Mining* blueprint (see Section 3.7) enables first the exploitation of existing process mining algorithms from the state-of-the-art as well as their semantic enrichment by being operated over semantically-enhanced logs derived from the Semantic KB. In the future, this blueprint might include extensions of existing process mining algorithms that attempt to improve them as well as new process mining algorithms that could focus on organisational aspects.

In overall, we actually see the devising and realisation of very critical and important blueprints which can really assist in the BPaaS evaluation task of the broker. This leads to producing a quite sophisticated BPaaS Evaluation Environment which delivers interesting business intelligence knowledge that could really enable improving the allocation and adaptive behaviour of BPaaS. This can also justify in part the immediate adoption of some of these blueprints in the current implementation (WP4) of the CloudSocket prototype.

1.4 Structure

The rest of this deliverable is structured as follows. Chapter 2, after analysing the respective state-of-the-art, is dedicated to modelling aspects by explaining the main languages that have been developed and are exploited in the context of the BPaaS Evaluation Environment. On the other hand, Chapter 3, first conducts a state-of-the-art over the different kinds of analysis offered for BPs and workflows and then describes in detail the main analysis kinds offered by the BPaaS Evaluation Environment. Both Chapters 2 and 3 end with particular future work directions which might be followed in the final period of the T3.3 task. Chapter 4 explicates the kinds of input expected from other environments to enable the proper functioning of the BPaaS Evaluation Environment while also explains the main types of output that can be generated by the latter environment. Finally, Chapter 5 summarises this deliverable by also indicating the actual process to be followed for incorporating the major and most mature results to WP4 (implementation).

2 BPAAS EVALUATION & MONITORING MODELLING BLUEPRINTS

In order to be able to perform any kind of analysis, there is a need to provide respective languages or meta-models which can dictate in which way the respective information on which the analysis relies can be structured and linked. In the context of the BPaaS Evaluation Environment, we have developed two main ontologies for this purpose: (a) an evaluation ontology which covers dependency aspects; (b) the KPI ontology (as an extension of OWL-Q) which covers the non-functional aspect. These ontologies are interlinked on exactly one point: the actual object that is being measured. In particular, OWL-Q specified measurements are associated to the instances of elements within the hierarchy of a BPaaS. For example, the value for the duration of a task will be a measurement while this measurement will refer to a specific instance of the task concerned. The latter task instance will be associated in turn to other element instances along the BPaaS hierarchy (dependency model) (i.e., workflow instance and BPaaS instance). This enables creating propagation/contribution hierarchies which can be useful for finding root causes of problems, such as SLO and KPI violations, in the form of events patterns. Such event pattern would then map measurements of objects which are related to the object of the KPI measurement according to the current BPaaS hierarchy. In fact, this linkage enables to support at least three kinds of analysis: KPI analysis, best deployment discovery and, as already indicated, event pattern detection.

In the sequel, after conducting a state-of-the-art over KPI and (BP) dependency modelling, we analyse both ontologies that have been developed. In the end, we provide respective future work directions which could be followed to further enhance the capabilities of these ontologies.

2.1 State-of-the-art

The state-of-the-art analysis is separated into two main sub-sections, devoted to the review of related work which is associated to the two main blueprint contributions, i.e., KPI and (BP) dependency modelling.

2.1.1 Dependency Modelling

Dependency modelling has been considered in the literature as a pre-requisite for enabling system monitoring and adaptation. Without such knowledge, both monitoring can be quite limited, covering mainly low abstraciton levels as propagation to higher levels is prohibited, as well as the respective adaptation capabilities can be quite restrained. Most research work has focused on system dependency modelling, especially until the software level, while few work has been devoted to covering dependency information for the workflow and process levels.

Business Process Level. An evaluation meta-model has been proposed in [44] which is an extension of BPMN. This meta-model actually maps different BPMN elements to respective information that can be derived for them. In particular, it establishes connections between activities and measurement data, between the conditions derived and the gateways, between BPMN connecting objects and their probabilities of execution, and between the BPMN artifacts and their content / structure. As such, this meta-model seems more appropriate for visualisation of different kinds of BP analysis results rather as a meta-model on which such analysis kinds can rely.

Business Process & Software Level. An ontology that covers the description of both business and software components along with their interconnections has been proposed in [57]. This ontology seems to cover well these two levels and has modelled all appropriate relationships. Apart from components, it also covers well notions of BPs, activities and services and data. The sole issue is that the ontology does not cover the infrastructure level at all. In addition, the type-instance pattern is not covered. In this sense, it might not be easy in practice to know which instances of a software component have been realised and which of these instances are used to realise

which instances of a particular BP activity. Thus, the ontology can be used nicely to cover static allocation information but not dynamic one that can reflect the current state of a running BP system.

System Level. In [29], the capability to model dependency graphs to cover the functional dependencies in a system is endorsed as the means to capture event correlations and support integrated event management. Dependency graphs are dynamic normal graphs in which nodes map to managed objects in the system while edges between nodes reflect the functional dependencies between these objects. Extensions to dependency graphs can be devised, e.g., using probabilistic networks to also derive a probability to each edge which could partition the functional dependencies into more probable and less probable categories. The author proposal for dependency graphs designated at that time a three-level architecture comprising the levels of services, end systems and network devices.

The above work of Gruschke has been slightly advanced by the work in [15] which proposes to use graph theory in order to support both functional and QoS-aware dependency management. The latter work proposes the capturing of both functional and resource dependencies for a system. While it is apparent what functional dependencies cover, resource dependencies indicate the resource requirements for software components and can be used also to drive the deployment of these components in the appropriate (underlying) infrastructure. The functional dependencies graphs in that work cover relationships between different components, like application tasks, middleware components or any kind of a system module. The resource dependencies connect a component to a specific resource type and can impose two types of restrictions on the latter: (a) percentage like usage (e.g., for CPU) and (b) specific constraints on the resource size (e.g., main memory). We can regard that this work has actually inspired the development of cloud-based deployment languages which could be considered as graphs that more or less cover both types of dependencies.

Hasselmeyer [30] has proposed a simple dependency model which could be realised in the form of a graph centered around the notion of a dependency. Such a dependency connects the dependent component to one or more antecendent components. In addition, it can have a specific type and pattern of access. Different access patterns might be involved (e.g., single, round robin, ordered), characterised by different values on a respective set of properties, which includes the quantity of the remembered services, the quantity of the accessed services and the selection scheme.

An interesting categorisation of dependency models is provided by Ensel [22] which relies on different dimensions, such as the type of edges involved, the level of abstraction, the degree of detail and the types of services / components involved. An interesting distinction between instantiated dependency models and restricted real world models is also supplied which explicates that the former attempt to cover the whole real-world case but it is impossible to manipulate them in a manual manner while the latter map to restricted forms of the real-world case which raise slightly the abstraction level to reduce the size of the graph by merging nodes. After providing this dependency model categorisation, the author proposes a specific modelling process which first generates a real world model, out of the respective system data collected, which is the abstracted into a respective abstract model.

Service and Infrastructure Level. Cloud computing has updated the respective dependency models identified above with a special focus on the intrinsic characteristics of the cloud. We can in general characterise the respective approaches into template-based or instance-based. Template-based approaches attempt to model an abstract model of the dependency (see also work of Ensel [22]) by specifying a graph of abstracted dependencies between types of components. As such, (software) components are mapped to other (software) components via containment relations (e.g., service-based components are deployed on tomcat), while software components are mapped to respective infrastructure components via special containments relations, named as hostings. Infrastructure components are nodes which also either carry requirements over the respective actual infrastructure components to be selected or directly map to such actual components. In the first case, the

requirements drive the so called deployment reasoning which leads to the final selection of the actual infrastructure component offering to be selected per application component to be hosted.

Template-based components are static in nature reflecting a nice abstraction, good for visualisation purposes, over a system. However, they are not able to cover dynamic aspects. This gap is covered by instance-based models which usually follow the models@rutime approach. These models follow the type-instance pattern by explicating how many and which instances of one component are connected to respective instances of another component provided that a respective abstraction dependency between these two components already exists and has been modelled. The instance-based models are suitable for the dynamic management and adaptive provisioning of cloud-based applications.

As the respective analysis of deployment meta-models has been already performed in D3.3 [19], we restrain ourselves only to specific examples for each type of language able to support the corresponding types of dependency models. A prominent and standardised language for the capturing of abstract dependency models is TOSCA [50] which seems to be widely referenced in the literature. On the other hand, a prominent language, focusing on modelling instance-based deployment models, which has already been adopted by the CloudSocket consortium, is CAMEL [53] and especially its deployment meta-model/sub-language. In either case, we can see that most of the languages in the cloud domain go until the software / application level. Thus, they are not capable of capturing dependencies at higher-levels of abstraction and thus also between these levels and the existing modelled ones.

State-of-the-Art Evaluation. A comparison table summarising the evaluation results for the aforementioned approaches (apart from [44] which is not an actual dependency model) is depicted below. This table includes the following set of criteria that were exploited for this evaluation, drawn from the literature or devised from the authors of this deliverable: (a) *abstraction level*: which levels (denoted as 'BP' - business process, 'SE' - service, 'Inf' - infrastructure) in the BPaaS hierarchy are covered; (b) *formalism*: the dependency model formalism used; (c) *runtime*: whether the dependency model covers a dynamic or just a static system view. Dynamic views enable to cover the system evolution and provide support for realising monitoring and adaptation mechanisms; (d) *detail level*: how well the component dependencies are specified.

Approach	Abstraction Level	Formalism	Runtime	Detail Level
[57]	BP, SE	Ontology	no	good
[29]	SE, INF	graph	yes	low
[15]	SE, INF	graph	yes	moderate
[30]	SE, INF	graph	yes	good
[22]	SE, INF	graph	yes	good
[50]	SE, INF	DSL	no	good
[53]	SE, INF	DSL	yes	good
Ours	All	ontology	yes	good

Table 1 - Comparison table for the state-of-the-art in Dependency Modelling

Table 1 clearly shows that our ontology covers all possible levels, captures runtime information and supplies a good level for dependency specification. It is thus better than all other related work. Only [57, 53] are the sole competitors which do not, however, cover all BPaaS hierarchy levels. In addition, the approach in [57] does not

capture runtime information, while CAMEL does not rely on semantics. Based on these evaluation results, we should highlight the following facts: (a) an ontology-based approach is necessary to enable a better integration of dependency information from different information sources as well as inferencing over this information; (b) the good dependency detail level in some modelling approaches represents a place for improvement.

2.1.2 KPI Modelling

As KPI modelling is a pre-requisite for KPI metric measurement and KPI evaluation, a great amount of research work has been devoted in producing meta-models, languages and ontologies focusing on the appropriate description of KPIs. This could also be considered as a major necessity due to the lack of the standardised BP languages to cover appropriately the so called BP context perspective (which includes goal-based and measurement information aspects), as indicated in [42]. Compared to our related blueprint, the OWL-Q KPI extension, however, it is evident that this extension is rather richer in terms of metric measurability aspects. This is due to the fact that OWL-Q, as being adopted, is already rich enough to cover all aspects related to measurability, including the metric, attribute, unit, and value type aspects. In addition, as it is based on ontologies, it also enables support to important reasoning tasks, such as the alignment of non-functional specifications based on their terms. This KPI extension might not score well on the component/object description aspect. This is actually a design decision which has relied on the fact that an additional ontology would cover that aspect in the form of a respective evaluation ontology blueprint. Apart from other interesting base features of OWL-Q, such as the inclusion of rules for semantic domain validation, the OWL-Q KPI extension also focuses on two main issues which have not been actually considered widely in the literature: (a) the addressing of manual measurements and (b) the exploitation of external information during metric formula computation in the form of query results. Both features are however required to cater for the current practice in KPI modelling & evaluation which signifies that measurement production is not always automated and that additional information might be needed in metric formulas apart from the component metrics of the current KPI metric at hand.

The KPI meta-modelling approach in Wetzstein et al. [65] is an attempt to increase the level of abstraction in service-based systems by including the BP level. In this level, a new kind of metric called process performance metric (PPM) is envisioned which actually participates directly in the definition of KPIs (by considering that KPIs are conditions over metrics). PPMs are measured by process measurement directives which include probes over processes, activities and process data. As such, the respective KPI meta-model include the coverage of process-related artifacts that are linked to (software) service-oriented ones. The proposed KPI meta-model does not seem to cover the infrastructure level, while it is not semantics-based. In addition, the coverage of measurement aspects is rather moderate and the same holds for the specification of KPIs which are considered simply as thresholds over PPM metrics.

A simpler KPI meta-model is proposed by Motta et al. [48] which includes the notions of KPIs, SLOs and metrics, creates respective relationships between them and connects them to BPs and respective business models. However, this KPI meta-model can be considered as quite poor, especially in the coverage of all possible measurability aspects. What seems to be covered is the supply of formulas for metrics which relies on the HIGO framework [47]. In addition, the connection to the dependency aspect is actually missing preventing the actual measurement of high-level KPIs from lower-level ones.

A simple metric meta-model is proposed in [9] which defines for a metric four kinds of information: (a) the name of the metric; (b) its data type; (c) the object being measured; (d) the metric mapping function which explicates how to compute the metric from an underlying database (i.e., via SQL formulas). This is a rather very simplified meta-model which looks also to be database-specific. In addition, it presupposes that a kind of hierarchy between the objects to be measured has already been established in the database from which also correlations between objects and respective metrics can be drawn. Other measurability aspects seem not to be touched, while KPIs are not modelled.

In [51], a Business Motivation meta-model has been proposed which attempts to map business (non-functional) goals to KPIs. KPIs are modelled by considering a rich set of properties which includes different types of tolerances. Unfortunately, while we regard that the connection of KPIs to other business motivation artifacts like goals is needed, the respective meta-model is poor in covering the actual measurability aspect. By carefully inspecting different contributions focusing on this meta-model from the same authors we can see that they do take into account also the metric formula while units are only covered as simple strings. There is no indication of whether metrics can be computed from other metrics as the notion of a metric seems to be missing while external information sources seem not to be covered.

A (process) metric meta-model that extends BPMN has been proposed in [26]. This meta-model seems to cover well the measurement of metrics for BPs, activities and their respective data. However, it seems to focus mainly on time-based and frequency measures. On the other hand, it is able to specify both process and instance-based metrics.The notion of KPI is indirectly covered by the introduction of the Target Definition which includes both upper and lower thresholds over the measurements of a specific (process) metric. Units are only covered in the form of enumerations highlighting the fact that new units are not allowed to be specified by the modeller. The main strength of the meta-model is its tight connection to BPMN as well as the fact that it is able to cover the complete state of both BPs and activities. In addition, it seems to be able to cover the mapping from respective metric conditions that might be violated into actions that can be performed to resolve these violations. However, the current set of actions seems to be quite limited which even map to the abortion of a BP. A major drawback of the meta-model is that it does not cover well the rest of the levels in the overall BPaaS stack.

The authors in [25] propose the Score-ML domain specific language for the modelling of indicators. This language is able both to define indicators as well as to connect them to business goals and referenced objects. However, the referenced objects seem to be poorly covered mainly at the BP level (e.g., BP itself and its resources/products). The language is able to categorise the indicators, while it includes two interesting self-relationships on them: (a) *computedFrom* indicating the one indicator can be computed from another one; (b) *similar* to express the similarity between indicators. The main issue with this modelling is that the notions of metric and indicator are intermixed. However, we advocate that this is incorrect. The metric explicates the way the respective measurements for an indicator can be produced while the indicator itself imposes particular threshold(s) on these measurements. Moreover, the way the indicator is computed is actually a relation between metrics forming a metric tree hierarchy rather than a relationship hierarchy between different indicators. Apart from this, the units of measurement are not covered while the measurement formulas for indicator metrics are not modelled. The latter represents the second major issue. Establishing relations between (indicator) metrics is not enough but there is a need to clarify how metrics can be computed from their componnt metrics. The coverage of external information also seems to be missing from this language.

In [52], an approach that covers the functional and non-functional specification of service-based processes has been proposed which considers mainly two levels: the choreography and the orchestration ons. Concerning the non-functional aspect, the well-known WSLA [78] language is exploited for the non-functional description in the choreography level, while an extension to WS-Policy, named as WS-QoSPolicy, has been proposed for the coverage of the orchestration level. Please note that WSLA seems to be a quite expressive language which however needs some extensions in order to cover the BP level. Moreover, WS-QoSPolicy seems to be very simplistic allowing just for the specification of thresholds over a specific set of QoS attributes. Obviously, the coverage of all levels is not supported by this language and especially of the infrastructure one. In addition, the use of different languages to specify the non-functional description in the two levels considered leads to an unnecessary heterogeneity which could be avoided. This is especially true in the case of WS-QoSPolicy where the notion of a metric is completely missing.

In [28], the authors propose a domain-specific language which enables high-level monitoring, measurement data collection and BP control which is complemented by a transformation approach to executable code. The

language itself comprises three main blocks: (a) the data block which defines the data to be used in the analysis and their types as well as new measurement data that are required for the analysis; (b) the event block which represents elements from the business domain to be used in the analysis rules as well as the points of connection between these rules and the BP; (c) the rule block actually connects the events to specific actions to be taken when they are violated. Concerning the definition of KPIs, these are captured via events while metrics are captured via the concept of a measure. Metrics are actually computed via the event block through the association of different events to the respective value / formula to be measured. The respective modelling approach seems to be implementation independent and has only a specific dependency over BPMN. However, we see that it covers only the process level while the notion of metric is not completely covered according to all possible measurability aspects. In this sense, it is actually difficult for a non-rule expert to be able to re-use definitions of metrics for computing a new metric. Thus, the use of the language will be restrained solely over (evaluation) experts on rule-based modelling. External information sources seem also not to be covered by this language. Correlations between KPIs may also be more difficult to specify.

The authors in [18] propose to use templates and ontology-based linguistic patterns in order to specify process performance indicators (PPIs). Templates seem to follow a table-based approach which is centered around 8 columns out of which the most important ones make a connection from a PPI to: a certain goal, to a certain type of measures, to the target value and to the context (or scope). An interesting connection is also provided with respect to the responsible for a measurement (which could be a component or a human). We can consider that the level of expressiveness for PPIs is moderate as: (a) only one target value is considered; (b) the notion of a metric is not directly modelled making hard the re-use of metrics in the specification of new ones. In fact, the authors seem to re-use PPIs for this purpose which has already been explained to be wrong; (c) the source of information and the respective data to be retrieved cannot be represented by a single linguistic term. Moreover, the authors seem to cover mainly the BP level and not the remaining ones. The nice features of this PPI modelling approach is that: (a) PPIs are do mapped to goals which can actually enable a goal-based analysis and (b) that the process scope / context is also covered enabling the definition of the number of process instances to consider or the measurement period to be employed. Ontology-based linguistic patterns (LP) are exploited in order to populate the content of the PPI templates. In this respect, the LP definition of a KPI will make a reference to the respective function to be used and the corresponding object to be measured in accordance to the type of the measure employed. For instance, an aggregated measure would enable the use of statistical functions over simpler / raw measures and on a specific object from a BPMN model, like an activity. The definition of the actual metric computation seems quite interesting as it involves a kind of restricted natural language form. This might be more user-intuitive but it is not certain whether it can really enable the specification of composite metric formulas which can be recursive in nature. A pure mathematical form might have been more appropriate. This is actually an issue that we currently explore, i.e., the appropriate formalism for the metric formula specification. A simple and restricted mathematical language could be employed in this case which would make the definition of the formula more compact and more easily understandable by the respective modeller.

In [13], the authors propose a process specification model which includes the definition of events and their correlation to respective events. Metrics can then be defined over events. However, the authors do not detail how this mapping is performed which seems to be rather simple, while not all appropriate measurement details are specified. The meta-model proposed seems also to be limited to time-based metrics which also reflects the inability to properly model metrics and their computation formulas. Obviously, other levels apart from the BP one are not also covered.

Liu et al. [43] proposes an approach where metrics are associated to process artifacts which become their context. In this respect, when actual events are produced by the system, they can be correlated with each other according to a specific artifact in order to produce its respective measurement. Compared to other approaches, this work seems to focus on resource measurement as it considers that efficient resource utilisation is a key for fulfilling KPI targets for BPs. The respective measurement meta-model proposed seems quite simple as it

includes a quite minimal number of concepts related to events and (external) data sources, monitoring contexts which include metrics, as well as situation triggers and outbound events. However, this meta-model is also coupled with an extended artifact model which covers relationships between business tasks, artifacts and resources. The authors also propose that a metric decomposition method should be followed for metrics modelling, such as Business System Design Decomposition and Analytic Hierarchy process, in order to reach the level of resource metrics that can be computed from resource and artifact content. Compared to our work, the approach does seem to cover a good number of levels, although the service level might be considered not to be appropriately covered, it does enable the exploitation of external information sources but does not cover well all the measurability aspects. KPIs are also not explicitly modelled. The metric specification model proposed is also non-semantic which does not enable incorporating advanced mechanisms of reasoning.

State-of-the-Art Evaluation. A summary of the state-of-the-art analysis in KPI modelling can be seen at Table 2. This summary relies on a respective set of criteria which attempt to evaluate a respective approach in the state-of-the-art. These criteria were either drawn from the literature or have been devised by the authors of this deliverable. They are as follows: (a) *KPI coverage*: how well the notion of a KPI is covered; (b) *metric formulas*: computation formulas are provided supporting the KPI metric measurement; (c) *measurability*: other aspects complementing metric specification are needed to cover all measurement details (e.g., units, measured objects); (d) *goal coverage*: connecting KPIs to goals enables to assess whether operational or even tactical goals are satisfied by performing goal analysis; (e) *semantics*: if the meta-model / language is semantic or allows semantic annotations. Semantics enables formal reasoning and reaching better evaluation accuracy levels; (f) *information sources*: ability of the language to exploit both internal and external information sources; (g) *measurement origin*: the language ability to cover measurements and explicate their origin (probes, sensors, or humans); (h) *level*: the levels covered ('BP' - business process ,'SE' - service, 'Inf' - infrastructure).

Criterion	[65]	[48]	[9]	[51]	[26]	[25]	[52]	[28]	[18]	[13]	[43]	Ours
KPI coverage	mode-rate	low	low	good	good	moder-ate	moderate	low	moder-ate	low	low	good
Metric formulas	yes	no	yes	yes	yes	no	no	yes	yes	no	yes	yes
Measur-ability	mode-rate	low	low	low	moder-ate	low	low	low	good	low	low	excell-ent
Goal Coverage	no	yes	no	yes	no	yes	no	no	yes	no	no	yes
Semant-ics	no	no	no	no	no	no	no	no	yes	yes	no	yes
Inf. sources	internal	intern al	interna l	intern al	internal	interna l	intern al	intern al	internal	intern al	both	both
Meas. Origin	probes	-	probes	-	probes	probes	probe s	probe s	probes	probe s	prob es	all
Level	BP, SE	BP, SE	BP	BP	BP	BP, Inf	BP, SE	BP	BP	BP	BP, Inf.	BP,SE, Inf.

Table 2 - Comparison table for the state-of-the-art in KPI Modelling

Based on the table evaluation results, we can see that only our OWL-Q KPI extension scores well over all criteria, it has better performance for almost all of them and can be considered as the most prominent. The only modelling work close to ours is the one in [18] which does not cover all levels, does not correlate measurements to human sources, exploits only internal information sources and provides a moderate KPI coverage. Furthermore, as it was already stated, it does not directly model the notion of a metric but intermixes it with that of an indicator.

2.2 Evaluation Ontology

In order to make appropriate correlations as well as perform various types of analysis over a particular system, it is critical to be able to capture the evolution of the system dependency model. Such a model reveals what are the actual components of the system and how they are interlinked. It can also highlight the dependencies between the components and especially the direction of such a dependency. Such a direction can actually indicate the way faults can be propagated in that system from lower to higher levels of abstraction. The same direction could also be involved for the propagation of measurements across different levels. The opposite direction enables performing root cause analysis: i.e., from a current, issue at a high-level component go down to the actual component in a lower-level to be blamed for this issue generation. Based on this analysis, it is essential that deployment models are specified via corresponding meta-models or languages. This specification will enable the appropriate structuring and interlinking of the respective component information which can then facilitate the component measurement, evaluation and management.

The Evaluation Ontology represents such kind of meta-model or language that aims to cover both deployment and state information about all the components involved in a BPaaS overall system (the so called BPaaS hierarchy). The fact that this meta-model is semantic also makes it appropriate for enabling formal forms of reasoning. In addition, other main benefits of the Evaluation Ontology are actually related to the information aspects being covered which make it quite extensive and suitable for the respective kinds of analysis to be supported by the BPaaS Evaluation Environment. These include:

- Capturing of the I/O parameter / variable values for tasks and workflows which can be beneficial for decision mining
- Natural linking of evaluation elements with ontology concepts that have already been used for their annotation. For instance, a task can be associated to a functional category. Such information is invaluable for semantic (process) mining.
- The capturing and linking of user and role information enables to use existing organisation mining algorithms as well as correlate particular human resource patterns with SLO / KPI status information. The latter can be quite handy for consultancy reasons for the broker or even in case for a customer which desires to exploit the BPaaS Evaluation Environment on his/her own.
- Workflow & task start and end times enable not only to compute execution times but also to recreate the workflow model via process mining. Moreover, such a recreation can also unveil discrepancies with respect to, e.g., paths modelled which are never followed in practice or paths which are frequently altered (e.g., some tasks are removed or the order of tasks is changed).
- As already stated, linkage with OWL-Q enables performing KPI, best deployment and event pattern discovery analysis, especially as the allocation knowledge is also recorded.
- Adaptations are also covered enabling to derive interesting facts, such as the degree of success of the adaptation rules modelled.
- Coverage of all levels from infrastructure up to the workflow.
- Extensibility is catered as the Evaluation ontology can be extended to cover additional information about respective concepts or additional relationships between concepts.

In the following, we shortly analyse the main concepts modelled in the ontology and the respective relationships between them. The actual ontology is depicted as a UML class diagram in Figure 1.

The ontology follows the well-known type-instance pattern. This enables us to capture both the allocation decisions made as well as the actual allocation that has been performed along with its evolution. At each level in the pattern, respective relationships are introduced to link the respective concepts involved, while inter-level relationships are also modelled to make the required linking between the types and their instances. We should also highlight that an important result of this interlinking is that also BPaaS are connected to the BPaaS instances created for the respective customers that have purchased them. This interlinkage is important in order to perform KPI analysis for a broker which involves the evaluation of metrics across all the instances of a certain BPaaS.

Similar to the new design of OWL-Q [39], the Evaluation ontology includes (generic) data properties which can be mapped to all or a subset of the concepts modelled. For instance, *name* and *id* properties can be attributed to any concept while *endpoint* / *URL* information can only be attributed to services (instances). To this end, from now on, whenever we analyse a concept, we do not always explicate generic data properties that characterise it but only specific ones.

We also need to explicate the way organisational information is covered. The *Tenant* concept actually models an organisation which is associated to a set of users. A tenant can either be a *Broker* or a *Customer*. In the latter case, a customer has a specific type which can take the values of “SME” and “START-UP”, thus covering the main customer types aimed by the CloudSocket project. Any user is associated to one or more roles that it has been assigned to by its organisation (tenant). Please note that there are actually two cases here in the actual CloudSocket prototype: (a) an organisation provides an overall user-to-role assignment; (b) the assignment is performed in the context of a particular BPaaS workflow and the roles that this workflow includes. Both cases can be covered by the Evaluation Ontology. However, we should note that (a) maps to cross-workflow roles which should hold for any workflow, while (b) to workflow-specific roles. In this sense, it might be better that for the case of (b) we design roles that are re-used across different workflows. Otherwise, in terms of actual ontology population (see Section 3.3), there can be cases where conflicts in roles might lead to wrong modelling and storage of respective triples in the respective Triple Store (Semantic Knowledge Base).

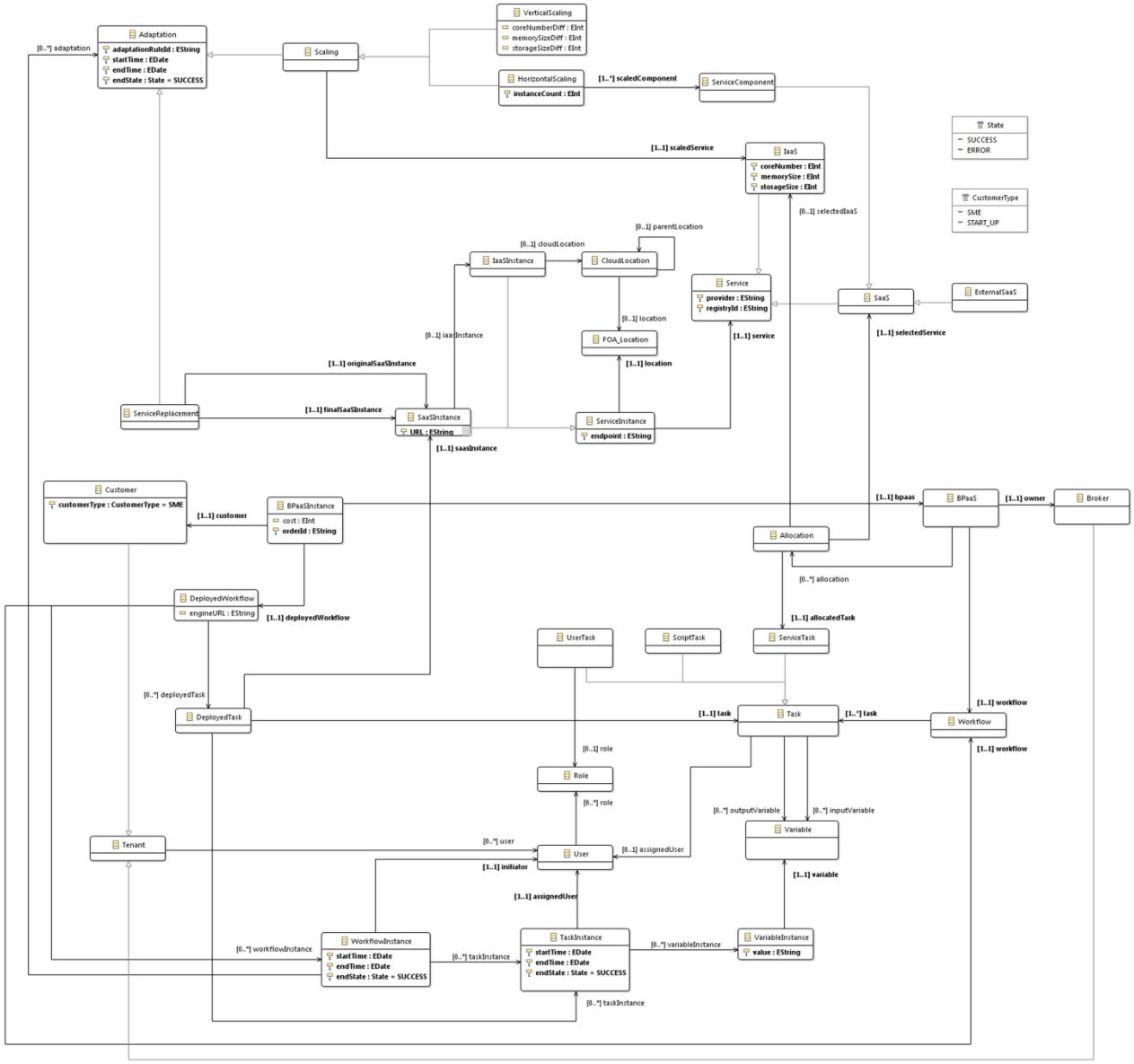

Figure 1 - The UML class diagram of the Evaluation ontology

The actual analysis of the ontology follows a top-down approach from the type to the instance level. Concerning the type level, the top concept represents the whole BPaaS, which has a specific *owner* (see Tenant concept) and is associated to the actual abstract workflow to be allocated, deployed and run in the cloud via the *workflow* object property. This abstract workflow, represented by the *Workflow* concept, is associated in turn to its main tasks. An abstract workflow is an almost final workflow for which some service tasks map to abstract and not concrete / real services. This is inline with the definition of *abstract workflow* in D5.2 [8] mapping to those workflows which are produced by the BPaaS Design Environment.

A *Variable* is an abstract representation of a variable which can map to the input or output of a workflow task. A workflow task is represented by the *Task* concept and is associated to a specific user or role that is assigned to it. We regard that both cases can happen alternatively for a task, but not both, and thus we cover them with object properties mapping to the corresponding concepts of the Evaluation ontology.

A Task can be further classified into a *ManualTask* (performed by human workers), *ScriptTask* (performed automatically by the Workflow Engine via locally running respective scripts) and *ServiceTask* (performed automatically by the Workflow Engine by calling a respective SaaS).

As one abstract workflow can be mapped to many BPaaS and thus also associated to different allocation decisions, we need to account for this variation in order to support the best deployment analysis. To this end, we have generated the *Allocation* concept which represents the different allocation decisions that have been taken for a workflow in the context of one BPaaS. As such, one workflow can be mapped to many BPaaSes and each BPaaS can be mapped to a set of unique Allocation objects (i.e., a set of allocation decisions); thus we indirectly cover different variations in allocation with respect to this workflow.

Each allocation, in the context of a certain BPaaS, associates a service task of the workflow to a *SaaS* and an *IaaS*. Depending on the type of SaaS concerned, the association to an IaaS can hold or not, thus actually enabling one allocation object to represent two related allocations. In particular, a *SaaS* can be further distinguished into an *ExternalSaaS* or an (internal)*ServiceComponent*, where both types of SaaS can be used for the realisation of the functionality of a service task. A *ServiceComponent* can then be deployed on an instance of a specific VM offering mapping to an *IaaS*. In order to allow obtaining information for both types of SaaS, the SaaS concept is also associated to the *id* of the entry in the SaaS or Software Component Registries (depending of course on its specific type). Finally, the IaaS concept is associated to specific data properties that characterise a VM offering, such as the number of cores, the main memory and storage size, while it is also mapped to the *id* of the respective entry in the IaaS Registry (from which we can obtain more information about it, if needed).

The instance level includes instances of concepts involved at the type level plus additional concepts mostly related to BPaaS adaptation. We need to stress here that the mentioning of instances here is not the same as in the ontology world where an instance is an actual object of a concept. The instance is still a concept / class which means that concepts in both type and instance level can have ontology instances. Thus, whenever we desire to talk about instances of classes, we would determine them as ontology instances.

At this level, the top concept is *BPaaSInstance*, an actual instance of a BPaaS, which is associated with an actual customer that has purchased it, its actual *cost* (both final and current[2]) and the *id* of the order / purchase at the Marketplace (in case we need to find out more information about it). This *BPaaSInstance* is also associated to the workflow that is deployed upon successful purchase by the customer. The latter workflow is represented by the *DeployedWorkflow* concept. To make the link with the *Workflow* concept at the type level, the *workflow* object property is attached to this concept. In addition, this concept is related to the respective tasks (see *DeployedTask* concept) that have been deployed and the actual *URL* of the engine in which the workflow has been deployed. The latter information is essential for distinguishing between deployments of different Workflow Engine instances. Such differentiation might enable us to perform some kind of analysis over the performance of a Workflow Engine.

The introduction of the *DeployedWorkflow* concept has been performed in order to differentiate between the actual instances of a workflow that are created on demand by the BPaaS customer from the actual deployment of the BPaaS workflow to which these instances conform to. These instances are represented by the *WorkflowInstance* concept, which is associated to the following concepts: (a) to the instances of tasks that have been created (see *TaskInstance* concept); (b) to the start and end time of the workflow instance; (c) to the resulting state of the workflow ("SUCCESS" or "ERROR"); (d) the user that has initiated this instance; (e) the adaptations that have been performed on this workflow instance to keep up with the SLOs promised.

A deployed task for a workflow has been created to represent the actual concrete allocation that has been performed for a BPaaS workflow task. This allocation is actually represented via associating a deployed task to the respective instance of a SaaS used to realise its functionality. A *SaaSInstance* is related to the specific *UR*L on which is available for invocation. We do not further classify the *SaaSInstance* as there are no further

[22] Actually, the same data property plays both roles. When the BPaaSInstance purchase period has not been ended, it can represent the current cost. When this period has been finished, then it represents the final cost.

information aspects that might be required to be captured for the respective instances of an external SaaS or an internal software component.

However, in case that a *SaaSInstance* maps to an internal service component instance, we map it to the respective *IaaSInstance* that has been used to host it. Both a SaaS and an IaaS instance are instances of a service. To this end, we have generated a respective super-class that subsumes them which is named as *ServiceInstance*. This latter class encapsulates some common characteristics for these two sub-classes which map to: (a) the actual endpoint of the service, (b) its physical location and (c) its cloud location. Physical locations are captured via the FAO[3] (United Nation's Food and Agriculture Organisation) geopolitical ontology[4]. In particular, the physical location of a service instance is mapped to the concepts of *geographical_region* and *self_governing* which are used to represent geographical regions, like continents, as well as all locations mapping to self governing countries, respectively. Cloud locations are currently modelled via the *CloudLocation* concept which is associated to its respective parent location (to cover hierarchies of cloud locations as exhibited in reality) as well as to the actual physical location this cloud location maps to. In this way, we actually cover arbitrary hierarchies of cloud locations especially as each cloud provider utilises its own taxonomy / hierarchy to model them. Please note that there is no need to cover the hierarchy of physical locations as this is already captured by the FAO ontology.

A deployed task is also associated to its respective task instances which are generated by running instances of the deployed workflow. The latter are modelled via the *TaskInstance* concept. Similarly to the case of a workflow instance, they are associated to their start and end time, their ending state ("SUCCESS" or "ERROR"), the user (if exists) assigned to and executing them and the instances of their input / output variables that have been generated. Please note that the user should, logically speaking, belong to the group assigned to the type of this task instance (i.e., a *Task*) or should be identical to the one assigned to this type. Instances of variables are represented by the *VariableInstance* concept which is associated to the actual *Variable* concerned but also carry information about the respective value obtained (after the execution of the corresponding task instance).

Finally, two types of adaptation have been modelled: (a) service replacements and (b) scaling ones, representing the most usual forms of BPaaS adaptation. Additional ones might be modelled, in case the respective need arises in the context of the CloudSocket prototype which will materialise into a respective novel adaptation capability. Any adaptation (see *Adaptation* concept) is associated to some generic characteristics, including the adaptation start and end time, its final state and the *id* of the respective adaptation rule whose triggering has led to performing this adaptation. A *ServiceReplacement* is solely associated to the service instance being substituted and the service instance used to substitute it.

On the other hand, any scaling is associated to the respective IaaS to be scaled[5]. Two main kinds of scaling are envisioned: (a) *HorizontalScaling*: here we scale the IaaS and one or more of the service components that are hosted by it, while we also specify the amount of instances of the IaaS plus service components to be generated or removed (where a positive or negative number should be provided, respectively); (b) *VerticalScaling*: here we indicate the way the IaaS should be vertically scaled by explicating the amount of respective characteristic(s) that has(have) to be increased or decreased (see properties *coreNumberDiff*, *memorySizeDiff* and *storageSizeDiff*).

As it can be seen from the above analysis, the Evaluation ontology attempts to cover the whole BPaaS hierarchy for both the type and instance levels plus respective information that complements it, such as which adaptation actions have led to its evolution and which are the respective organisations, users and roles related to it. In our view, this ontology is quite complete with respect to the current analysis tasks envisioned to be supported. It

[3] http://www.fao.org/

[4] http://aims.fao.org/aos/geopolitical.owl, http://www.fao.org/countryprofiles/geoinfo/geopolitical/resource/

[5] which covers also the case that one service component is to be hosted on multiple IaaS such that we need to know which IaaS is to be scaled along with the component

seems also to be one of its kind, by inspecting the current literature, especially as it is the first to deal with the capturing of whole deployment hierarchies over a new form of cloud services, i.e., a BPaaS. Of course, its power and potential application extensiveness even grow when combined with the OWL-Q ontology. We foresee that in the future some slight extensions might be performed in the ontology so as to suit either extensions to the current analysis types offered or new types of analysis that are incorporated in the BPaaS Evaluation Environment.

2.3 OWL-Q KPI Extension

OWL-Q [40] is a prominent ontology language for the non-functional specification of any kind of service. Its design has been carefully performed in different facets which cover all appropriate information aspects related to the capturing of measurements. By relying on semantics, it is able to support various reasoning tasks. This has been well proven by the fact that the OWL-Q ontology is accompanied by semantic algorithms which are able to support the alignment of non-functional (service) specifications according to their terms [37], the matchmaking of non-functional specifications [41] and service negotiation [12].

As reported in D3.3 [19], OWL-Q has been recently updated to become more compact. In addition, an extension of OWL-Q, called Q-SLA [39] (see again D3.3), has been proposed able to support the semantic specification of dynamic SLAs. OWL-Q is also now being supported by a greater variety of rules, also covering the SLA aspect, which enable the production of added-value facts (e.g., matchmaking of non-functional terms) as well as the semantic validation of the OWL-Q specifications.

While OWL-Q seems to cover in a complete manner the specification of QoS profiles and SLAs, it seems not be yet able to specify KPIs. However, in the context of this project, not only the modelling of KPIs should be supported but this modelling needs to be done in a semantic way so as to enable the realisation of at least the semantic KPI analysis tasks envisioned. To this end, by also considering that the coverage of metrics is more or less sufficient for the business level, it has been decided to produce a novel extension of OWL-Q towards the coverage of non-functional aspects of BPs (and thus also BPaaS), including of course the notion of KPIs. This OWL-Q extension builds upon the current OWL-Q constructs, considers the relevant new parts that need to be covered via the state-of-the-art analysis results and also takes into account the main requirements of the CloudSocket project. As such, the extension is minimal enough but sufficient to cover the new domain and still capable to cover the current use cases in the project.

One of the main challenges that had to be overcome were not related to the modelling of the KPIs themselves but to the fact that in the world of BPs, measurement data are not always automatically produced. Moreover, to complete the specification of metric formulas, external information needs to be accessed. In this respect, two particular extensions made to OWL-Q were associated to the capability to enable both the manual completion of measurements as well as the capability to exploit external information resources in metric formulas. The approach followed for the first extension was to not always relate (directly or indirectly) a measurement to a specific sensor or measurement directive but to a human resource which was able to produce that measurement. By considering that all modern information sources are available in the form of REST APIs or database endpoints, the second extension was realised, via incorporating a respective class in the ontology which is able to cover the call required for retrieving the appropriate information within a metric formula.

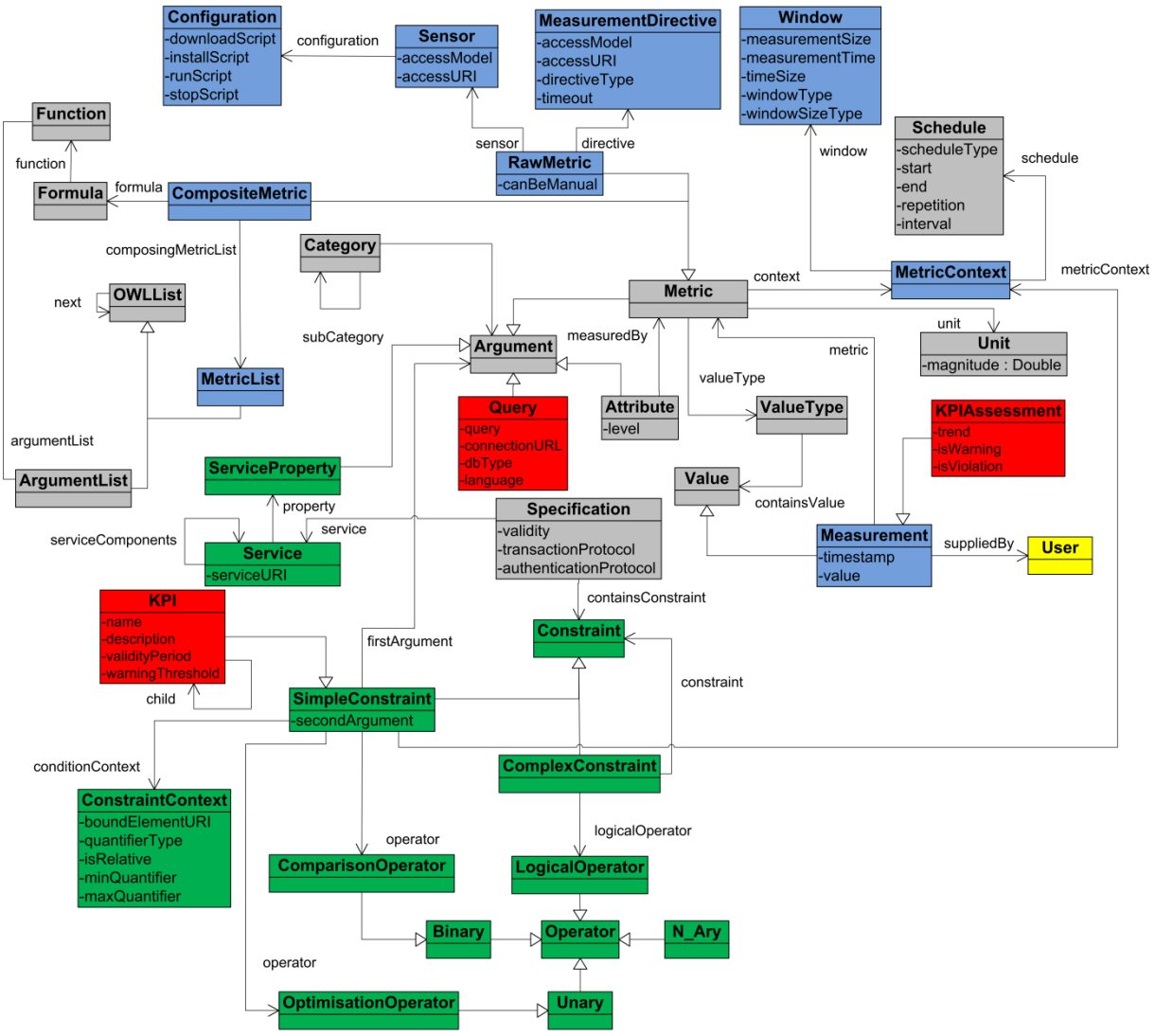

Figure 2 - The OWL-Q KPI Extension

The OWL-Q KPI extension can be seen in Figure 2, where with the grey colour you can see the generic concepts of OWL-Q mapping to the general facet, with blue the measurement-related concepts mapping to the metric facet, with green the specification-related concepts mapping to the specification facet, and with yellow the concept(s) mapping to the Evaluation ontology, while the red colour is used to denote the new concepts mapping to the actual OWL-Q extension which has resulted in the production of the KPI sub-facet of the specification facet. In the following, we proceed with the analysis of this extension by focusing on different main sub-aspects.

A Key Performance Indicator (KPI) is an indication for the performance of a specific BP (or its parts). Such an indication can signify that the performance is either satisfactory, problematic or erroneous. These three states can be captured through the introduction of two thresholds: (a) a warning and (b) a violation threshold. Performance is satisfactory when it is above the warning threshold, it is problematic when is between the warning and violation thresholds, while it is erroneous when it is below the violation threshold. This mapping of states to threshold ranges concerns the case of a performance metric which is positively monotonic. For a negatively monotonic performance metric, the order between the warning and violation thresholds is reversed and the state mapping goes on a symmetric manner (i.e., satisfactory level is below warning while erroneous level is above the violation threshold).

KPIs are represented via the *KPI* concept. As a KPI can be considered as a kind of simple constraint which also carries additional information, KPI has been made a sub-class of *SimpleConstraint*. By considering that a simple constraint already includes a reference to a respective metric and (violation) threshold, the KPI information can be completed by incorporating the specification of its name, a human-oriented description (for human consumption and comprehension), a validity period as well as the warning threshold. The validity period indicates the period of time that the KPI should hold.

A (KPI) metric is a concept already covered fully in OWL-Q by putting emphasis on aspects, such as the metric unit and value type. In addition, OWL-Q covers well the modelling of metric groups. At the BP level, this can map to having metrics classified in four widely referenced groups: (a) time, (b) quality, (c) customer satisfaction and (d) financial [35]. By considering of course the literature, such a grouping can be extended and become quite nested. This indicates that groups can have as members either metrics or other groups. However, we do need to make the following differentiation. A group hierarchy could play two different roles: (a) it could represent a hierarchical and possibly complete quality model which could be exploited to specify KPIs; (b) it could represent restrained hierarchical preference models representing the actual way measurements can be propagated from the lowest to the highest level by also providing weights to each node in the hierarchy signifying its relative importance and contribution to the higher-level quality of the parent node. Such weighting over such preference models plus their actual restricted content could be BPaaS or customer-specific and might be quite subjective. This also signifies that weights can be modified as suited at evaluation time to represent the change of (broker/customer) opinion or cater for any kind of initial misjudgement.

Before continuing, we need to explicate here that in other KPI meta-models in the literature, weights seem to be given to KPIs and not the metrics themselves. This is an alternative way for representing this nested metric structures. However, such a modelling caters for an ad hoc propagation (e.g., possibly with the on-the-fly grouping of KPIs in order to produce a certain higher-level quality value) of quality and not for a generic one. In any case, we believe that this kind of modelling is not quite suitable as the inclusion of threshold information in the nesting can be irrelevant or not actually needed.

Metrics can also be distinguished into raw and composite. Raw metrics can be immediately measured from the (service, e.g., BPaaS) instrumentation system or from sensors. On the other hand, composite metrics can be computed from other metrics via metric formulas. In the case of KPIs, it is always meaningful to refer to composite metrics. Thus, the definition of a composite metric and especially of its metric formula plays a very important role.

In the extended version of OWL-Q, a formula is associated to a specific description to explicate its main purpose and meaning for human-oriented interpretation. This is appropriate in the case of KPIs which should be understandable also by business users. In addition, as already stated, we also consider that the computation of a KPI metric can rely on exploiting different data sources: queries, formulas, metrics & attributes, data properties of I/O objects and services, and humans. In this sense, we opt for defining a generic way of specifying a metric computation formula which is also independent on the underlying implementation (as indicated in the context of the BPaaS Evaluation Environment implementation where SPARQL queries are intended to be executed over the content of a semantic repository). As such, in OWL-Q, the respective computation formula is expressed through appropriate constructs which map to these underlying data sources. Then, based on this formula, model transformation can be followed in order to produce, e.g., a respective SPARQL query to be posed on a semantic repository, as is explained in detail in Section 3.4.

OWL-Q models formulas via the *Formula* concept which is associated to a *CompositeMetric*. A formula maps to the execution of a certain *Function* which takes as input a list of *Argument*s (*ArgumentList*). A function can be numeric (e.g., PLUS, MINUS) or statistical (e.g., MEAN, MEDIAN). In fact, many functions from both kinds are already incorporated in OWL-Q. An *Argument* was originally mapped (as a super-class of) to *Metric*s, *Attribute*s,

*Value*s (constants), *ServiceProperties* (for a composite service and its service components) and formulas (to enable formula composition for expressing more composite formulas) but now it has been extended to include the other (possibly external) data sources which map to queries. The new type of data source currently captured is represented by the *Query* concept, a sub-concept of the *Argument* concept to enable the posing of queries and the retrieval of their results to be exploited in metric formulas. This concept is associated to the following information: (a) the main query string, (b) the connection information (URL) (with respect to the data source / database), (c) the database type and (d) the query language The database type can map to specific relational database management systems, like Postgres or MySQL, or to specific NoSQL databases, like MongoDB. In this way, through the above generic query-related information, we are able to actually exploit data from any kind of database, either it is semantic, relational or NoSQL.

Please note that there are two ways to exploit formula queries: (a) the query is performed in advanced and then its result is used for the evaluation of the respective computational formula in which it is incorporated; (b) the query is evaluated dynamically on-demand during the evaluation of the metric formula. The first way might be more appropriate in case of static information. As such, the query could be evaluated just once and its respective result could be incorporated as a normal value inside the metric formula. This would certainly save time during KPI evaluation/assessment. The second way is more appropriate in case that the query content is modified dynamically. To this respect, it is more suitable to run the query only at the time point it is needed in order to guarantee the most up-to-date result retrieval. Thus, based on this analysis, we can see that each way is complementary to the other. As such, to enable exploiting both of them in metric formula evaluation, another property was included in the *Query* concept in order to explicate the type of information to be retrieved.

It could be argued that human-based information is not covered at all in the above extended modelling. This is not actually the case. We do account for human input which could take different forms: (a) input provided to supporting the execution of a BPaaS (e.g., service input); (b) manual measurement for a specific metric. In the case of (a), we consider that if such an input is provided, it would then be stored in the database of the respective workflow engine which can then be retrieved via a (direct) query.

In the case of (b), we consider that the user would have to opportunity to exploit different mechanisms (e.g., provide input in a special task of a BPaaS workflow - mapping to information that could be retrieved via a query; supply a file of measurements that has to be processed and then used to populate the Semantic KB; use the Hybrid Dashboard of the BPaaS Evaluation Environment in order to provide the respective measurement). Once these measurements become available, we can re-use them in the way component metrics measurements are exploited to compute composite/high-level metric measurements. The only detail that has to be accounted for and which has been addressed in the extension of OWL-Q is the suitable marking of the respective raw metric source. In OWL-Q, only automated metrics were initially considered whose values / measurements can be computed from sensors. In the OWL-Q extension, we indicate that raw metrics can also be provided via human input by incorporating a respective boolean attribute. In addition, a measurement is now also mapped to a respective *User* from the Evaluation ontology, thus creating another connection point with the latter ontology.

A KPI should also be associated to a specific assessment period on which it should be applied. Such a period is currently captured in OWL-Q through the metric context which needs to be pointed by the respective KPI definition. Apart from the assessment period, the context also indicates other important information such as what is the measurement window and which is the (BPaaS) element that is being measured. In fact, any object within a BPaaS hierarchy could be measured residing at any level of abstraction (e.g., the whole workflow of the BPaaS or one of its tasks). Actually, OWL-Q now relies on the Evaluation ontology to signify and link to BPaaS elements. In particular, a measurement can be associated to any element in the dependency hierarchy of the Evaluation ontology. As such, by also dealing with linked data, the URI of this element suffices to make the connection.

In order to enable a drill-down of the KPI assessment from higher-level to lower-level KPIs which can assist in root-cause analysis, we associate KPIs to each other through the *child* relationship which goes from a parent KPI to the respective child. This KPI relationship should be in accordance to the respective relationship between the metrics of the involved KPIs meaning that the parent KPI's metric should be a parent metric of the child's KPI metric. For instance, a KPI for the response time of a service could be related to KPIs mapping to the execution time of the service and the corresponding network latency.

When a KPI is assessed, the respective metric value computed is checked whether it violates the respective thresholds posed. However, apart from this, we are also interested in checking some other information like the value trend with respect to the previous assessment. To also make a connection to the original concept in OWL-Q called *Measurement*, which explicates the value of the measurement and its timestamp, we have created a sub-class called *KPIAssessment* which also contains three main information items: (a) the trend; (b) whether a warning threshold violation has occurred; (c) whether an (erroneous) threshold violation has occurred. Such information can enable us to also perform different kinds of analysis over the KPI assessments in order to check, e.g., particular tensions in terms of the BPaaS performance. For instance, we could assess whether the BPaaS performance is gradually reduced from the very beginning. Such information could enable us then to perform a root cause analysis in order to find the exact problem and solve it before it is too late by obviously evolving the BPaaS as needed.

This actually ends the analysis of OWL-Q's KPI extension. To summarise, we would like to highlight that this extension is lightweight while it relies and builds on the actual base content of OWL-Q. Instead of defining a complete new facet with its own conceptualisation, we build upon the specification facet by also re-using major parts from the metric facet in order to cover the KPI domain. The main rationale for this is that OWL-Q is already rich enough to be able to cover both the non-functional specification of services, such as a BPaaS, as well as complete measurement trees mapping to the computation of any (KPI) metric. The extensions performed were driven of course by the semantics of the KPI domain but also guided by the main requirements of the CloudSocket project. The result, though, is an extension which can be exploited across different projects and use cases based on the way the KPI domain has been covered. The current application of this extension over the project use cases shows that OWL-Q is able to cover all possible KPIs that need to be posed. This constitutes a major evaluation step which validates the design of this OWL-Q extension.

2.4 Future Work

2.4.1 Evaluation Ontology

2.4.1.1 PaaS Extension

The modelling should never outpace the actual developments of a system. In this sense, one particular direction that should be followed for the Evaluation ontology would be the support to the exploitation of PaaS services and their involvement in the BPaaS hierarchy, once of course the respective realisation at the Cloud Provider Engine is finished. However, by considering the actual current content of the ontology and the minimal but sufficient information that is captured by it, the respective extension will not require great effort but will actually be straightforward.

2.4.1.2 Adaptation Coverage

Similarly to the above direction, we foresee that possibly in the final CloudSocket prototype system there will be additional adaptation capabilities realised apart from those mapping to SaaS service replacement and horizontal scaling of internal service components. We should not also forget that possibly also the exploitation of the *Data Mediation Engine*'s services might be required in some adaptation scenarios. In this sense, whenever a new capability will be planned to be made available in the prototype, the respective modelling will be extended in order

to cover all the information required to capture it. We should highlight here however that this information will need to be drawn from a particular information source which should be made available by the prototype. That information source should be the *Adaptation Engine* itself which will have the knowledge about which adaptation actions are being performed in order to adapt the behaviour of a currently running BPaaS (see also Deliverable D3.4 [20]).

2.4.2 OWL-Q KPI Extension

2.4.2.1 Other types of External Data Sources

Apart from continously validating the OWL-Q KPI extension and obtaining feedback from modellers and use case owners, we foresee one major direction for further improving this extension. This direction is related again to the exploitation of external information sources in metric formulas. In particular, apart from databases which can be heterogeneous and might require utilising different query languages, (REST) APIs can be exploited which encapsulate such databases and might provide their respective information even in a query-independent way. Moreover, we foresee that information derived in different components / environments, even outside of the CloudSocket prototype, could be also exploited. For instance, external service registries or internal workflow engines could be exploited in order to derive additional information about services or execution-related information about (BPaaS) (service-based) workflows. Such information sources might not even expose their databases, especially if they are external to the CloudSocket prototype. However, even for internal components to this prototype, it might be safer to expose an API rather than the internal database of a component. In this respect, the coverage of (REST) APIs should be considered as a necessity that has to be covered in the forthcoming evolution of the OWL-Q KPI extension. We are currently investigating how such APIs could be called and what is the kind of information that has to be modelled to cover such calls as well as the respective output produced from them. Please note here that the coverage at the modelling side should take into account even content heterogeneity which might require incorporating different mechanisms for processing different output formats (i.e., XML vs JSON). Moreover, this coverage should be complemented at metric measurement and KPI evaluation time with the capability to actually enforce the call and process / transform the respective call result. The same holds for the evaluation of queries within metric formulas. Anyway, the OWL-Q KPI extension paves the way for exploiting different information sources in metric formula computation, while the exploitation of the information modelled is another subject which might lead to developing different respective realisations.

2.4.2.2 KPI (Metric) Models

While OWL-Q's KPI extension is well-suited for the description of KPIs, it is also capable of specifying metric / quality models. Such models, which cover the description of quality terms across domains and / or even in certain domains as well as the relationships between these terms, can enable the suitable completion of KPI descriptions as: (a) they can enable the browsing and search of the relevant metrics in order to specify the way the KPI can be measured; (b) they can also drive the generation of new metrics which are suitable of measuring KPIs based on existing ones; (c) lead to a complete description of KPIs and the respective non-functional terms exploited making the KPI description documents self-contained. To this end, as an interesting line of work that we plan to follow, we will conduct a thorough study over the state-of-the-art in the pursuit of a rich quality model which will then be further enhanced by incorporating on it additional metrics and related terms, which will be both domain-independent and domain-dependent. The use cases of the project can also assist especially towards the coverage as well as the validation of the domain-dependent metrics. The resulting KPI model will be made open-source and will be exploited in the respective editors in the CloudSocket prototype that focus on the description of the KPIs.

2.4.2.3 Goal / Business Motivation Modelling

A KPI just reflects a condition over a specific metric. It does not convey additional information, such as what was the motivation that influenced the generation and imposing of a certain KPI. As such, inspired by the work in [51] as well as the work in goal modelling, we plan to further extend OWL-Q to enable it to specify goals and other kinds of motivational business elements and connect them to respective KPIs. This kind of connection will not only enable traceability but also goal-oriented evaluation which can be quite significant for business experts as they would definitely like to know which of their strategic and operational goals are currently achieved. The extension to OWL-Q will follow the minimalistic pattern of only incorporating those concepts and relationships that can sufficiently characterise the respective sub-domain. In this respect, the compactness of the language will be very slightly influenced.

3 BPAAS EVALUATION HARVESTING & ANALYSIS BLUEPRINTS

This chapter focuses on presenting the BPaaS Evaluation Environment blueprints mapping to different modules which focus on different types of BPaaS analysis. The presentation starts with the state-of-the-art analysis over related work and then continues with the description of the architecture of the BPaaS Evaluation Environment research prototype followed by the analysis of each environment module / blueprint involved in that architecture. The presentation ends with the supply of interesting research directions that might be followed in the near future per each blueprint.

3.1 State-of-the-Art

The structuring of this section is performed by respective sub-sections which focus on reviewing related work according to each type of analysis that is being supported by the BPaaS Evaluation blueprints being proposed.

3.1.1 KPI Analysis

Various KPI analysis frameworks have been proposed employing techniques focusing on supporting mainly KPI evaluation while in some cases KPI drill-down is also supported. Most of the techniques focus on appropriately structuring the underlying database to support KPI analysis. In this sense, they employ relational databases, data warehouses, semantic databases or even combinations of such databases to support their main goal. What distinguishes our approach from the literature is the fact that we employ a semantic method to perform the KPI evaluation which relies on the principles of Linked Data (LD). In this sense, it gives rise to forms of creativity in the production of new KPI metrics which is one of the main requirements in the BP domain. In particular, by relying on the appropriate linked information in the underlying Semantic KB, queries which traverse the linked data in order to discover appropriate input information items for metric computation formulas are enabled. In fact, our approach also allows exploitating even external information sources towards this task. The support to metric formula exploration is also enhanced via the use of specific means to model it which then diminishes the need that the modeller should have the knowledge of SPARQL so as to perform this exploration. In particular, the modeller can just exploit a specific language to model the formula which is then transformed into the respective SPARQL query. In our view, this is a major distinguishing feature of our approach with respect to the state-of-the-art.

We should highlight that we do not focus on evaluation performance issues as: (a) the main goal is to specify appropriate KPIs and evaluate them over long periods - in this sense, it does not matter so much if the evaluation expert obtains the respective results in one minute or five; (b) it is not the goal of the project to improve the semantic technologies but select the best possible ones and exploit them. However, please note that recent developments in this area have made SPARQL querying extremely efficient in certain Triple Stores so we are confident that in the near future the evaluation of SPARQL queries will become extremely fast. Please also consider that there is usually a trade-off between performance, accuracy and expressivity. In this sense, one could argue that Time Series Data Bases (TSDBs) could be the best technology to adopt here. However, this is not true in the context of KPI evaluation as this would destroy the capability to combine the information stored in ad hoc ways so as to support the computation of new KPI metrics.

A framework which explicates the desirable characteristics for BP performance measurement systems has been proposed in [49] and derived from an extensive literature review. The framework also provides important guidelines which need to be considered when designing BP performance measurement systems out of which a process-based approach to BP measurement system design was derived. These guidelines are separated into

two main aspects: (a) characteristics over the BP measurement system design process; (b) characteristics over the output of the design process. The proposed approach was applied in three use cases which have also led to its improvement and further refinement via observing the respective results and obtaining appropriate feedback.

A Process Analysis Framework (PAF) has been developed in [72] with the main goal to facilitate the review of the analysis capabilities of data warehouse-based BP evaluation systems. The review relies on the consideration of six main perspectives: namely functional, behavioural, organisational, informational, goal and modelling. For each perspective, a set of analysis parameters is defined which are used for the actual evaluation according to this perspective. The framework proposed seems to be generic enough and with some adjustments (e.g., modification or replacement of analysis parameters specific to data warehouse issues) could be used as a basis for evaluating any kind of BP evaluation / analysis system. Based on this framework, 11 BP analysis systems were selected and reviewed out of a certain study. The review provided specific insights about the actual capabilities of the respective systems analysed. Some of the most interesting insights are that the reviewed approaches neglect issues in data warehouse design, very few consider goals in the evaluation, while no approach is able to support analysis over all possible perspectives.

In [2], a review framework of process measurement systems comprising eight core criteria has been proposed. These criteria span the BP representation, the BP measurement, the BP lifecycle management, intra- and inter-organisational levels measurement, process decomposition, intra- & inter-process connection measurement, interorganisational coordination measurement and common interorganisation strategy. As it can be seen, the main focus was on BP interoperability. This is the main reason that many of these criteria are related to this aspect. One interesting insight derived from the review relates to the fact that most of the systems cover very few stages of the BP management lifecycle. Moreover, the degree of interoperability consideration varies between all the systems that have been considered. Finally, the authors reported the lack of coordination measures in these systems as well as the lack of an interorganisational strategy.

The iBOM business operation management platform is proposed in [10]. This platform promises to deliver: (a) the analysis and management of BPs based on business goals; (b) the definition, measurement and improvement of the performance of BPs by exploiting also prediction techniqes; (c) the proposal of BP reconfiguration to optimise BP performance with the click of a mouse. The platform seems to be able to handle BP performance measurement over both business and technical metrics. It also promises to identify root cases for KPI violations. KPI metrics are specified via metric templates which include the SQL code to be executed when the templates are instantiated according to a particular BP. The authors here also suggest the mapping of one metric to multiple templates depending on the BP type. This kind of flexibility is already provided by our approach which also enables the modelling of the metric computation formula in an implementation-independent way. While the mapping to SQL is fixed in iBOM, it also creates the dependency that someone has to know the underlying table schema in order to formulate the SQL query mapping to the metric computation formula. In our view, this is a wrong technical decision, especially in cases where the database designer is different from the BP evaluation expert. In fact, the BP evaluation experts should ideally either not know the database schema, especially as this is a quite technical information for them, or at least have a kind of abstraction mechanism that is more close to the way humans think.

In iBOM, KPI drill-down behaviour is realised via the FAPE (Factor Analysis Prediction Engine) engine which employs data mining techniques, like decision trees, in order to explain the violation of a KPI. The authors indicate that such patterns can also be exploited in performance prediction, especially when the respective pattern is partially or completely instantiated which can then potentially lead to the actual KPI violation. While the approach proposed in iBOM is indeed valid and appropriate, it actually attempts to perform some kind of root cause analysis which relies on the data monitored. However, the main issue here is the actual low level on which measurements can be produced. In our case, we go until the infrastructure level and make all suitable connections within metrics at the same or across levels. In the case of iBOM, it is indicated that the technical

layer is approached with however no certain proves about the exact level reached. Moreover, connections between metrics are not actually enforced all the way up in the metric hierarchy which of course gives rise to involving data mining techniques to infer them. Of course, our approach could be enhanced via data mining techniques in case connections between metrics are not obvious or possible. Thus, there is some kind of complementarity here between the two approaches in this aspect.

In [66], a BP monitoring and analysis framework is proposed. BP monitoring relies on a Process Metrics Definition Model (PMDM) and employs measurement directives over service instrumentation systems and events subscribed over workflow execution middleware (i.e., workflow execution engines). A centralised component takes care of metric aggregation which could involve combining the aforementioned pieces of information together to propagate measurements from the service to the BP level. Compared to our approach, the PMDM description seems to be simple and maybe insufficient as it relies solely on providing filters over BP data or the so called hooks. However, in our approach, information can be linked in various ways while external information sources can be exploited. We should also note that the infrastructure level is not covered which means that KPI and drill-down analysis will be limited on the two levels of service and BP. Analysis in the reviewed framework relies mainly on applying decision mining techniques and especially decision trees to derive the most influential factors for KPI violation analysis. Thus, the same approach as in iBOM [10] is employed here which can be useful in case metric dependencies are not completely and sufficiently captured.

The rule-based approach in [13] relies on the existence of a semantic KB on which SWRL[6] rules are checked in order to discover KPI violations. This semantic KB is fed with measurement facts for KPI metrics which are then used for the assessment of the KPI violation rules. The production of KPI metric measurements follows an event-based approach which however focuses solely on time-based metrics. While this KPI evaluation approach can be considered simple, this does not necessarily mean that it is quite performant if we consider a large semantic KB with millions of facts. It has been well proven that reasoning time exponentially increases with the size of the KB. On the other hand, the use of SPARQL can potentially have a better performance over big KBs. In addition, SPARQL is more expressive and thus can be used to address more advanced KPI measurement and evaluation scenarios.

The authors in [67] exploit semantic annotations over BPs to specify semantic KPI descriptions via a small enhancement over the meta-model proposed in [65]. These descriptions are then transformed into IT-level events that can be exploited for BP monitoring via the use of reasoning technologies.The transformation follows a two step approach: (a) the metric formula is mapped to a WSML[7] logical expression which can be executed over event data being produced by a workflow execution middleware to compute the actual metric value; (b) the KPI condition maps to a respective rule over a KB comprising facts mapping to metric measurements which are then evaluated to infer whether the KPI condition is violated, thus actually following the approach in [13]. We need to highlight here the following: (a) events seem to be generated only from one information source and not from multiple ones; (b) it is not apparent how the metric computation formula is mapped to a WSML logical expression but this really resembles our KPI metric to SPARQL transformation approach; (c) the evaluation of KPIs still relies on the evaluation of rules which might not be very scalable in practice (as indicated above). The proposed approach seems also to be restrained as it covers only two levels while the measurability aspects are not well captured. The capability to play with metric formulas seems that could be supported but this is never stressed by the authors. The main issue here would be whether the use of the specific formalism for metric formulas in that approach would be user-intuitive and provide the most suitable abstraction level.

Thomas et al. [62] have proposed a semantic and agent-based architecture for the measurement and evaluation of BPs which comprises different types of agents, each one with a different responsibility to handle. Supervisory

[6] https://www.w3.org/Submission/SWRL/
[7] https://www.w3.org/Submission/WSML/

agents are responsible for the evaluation of the BP goals and invoke monitoring agents to obtain the respective measurements needed for this evaluation. The monitoring agents themselves are responsible for the actual calculation of the measurements, supported from data gathered by node agents. The node agents themselves are responsible for the gathering and transformation of data retrieved mainly at the activity level. While the monitoring architecture seems quite simplified, it does not explicate exactly what are the main techniques and methods used to realise the functionality of each agent type. In addition, the respective ontology model via which the required information (e.g., metrics and their formulas) should be described is not provided while it is apparent that it should also have the ability to cover the dependency aspect.

A model-based framework for monitoring and evaluating the performance of BPs has been proposed in [11]. This framework takes a model-driven approach in order to construct the actual monitoring and evaluation code via model transformation. Two main models are being exploited in such a transformation which conform to respective meta-models, named as observation and data warehouse meta-model. The first model leads to the generation of observation and action code while the second to the generation of visibility code generation. The monitoring capabilities of the framework relate to the aggregation of business events and related data for the production of business metric measurements. The evaluation of such measurements relies on predefined conditions which are also mapped to a set of adaptation actions. The proposed framework seems quite sophisticated and benefits both from model-driven and data warehouse technologies to enable realising sophisticated metric formulas as well as the automatic generation of the respective code. It also relies on a specific series of tools that have been developed by IBM to support the whole transformation process. However, we see actually two main shortcomings of the proposed framework: (a) semantics are not considered at all; (b) only the BP level seems to be covered. Drill-down of metrics is not discussed also while other kinds of analysis seem not to be offered.

We should also mention monitoring approaches for service-based applications which seem to be relevant for this work. As the respective literature review has been already conducted, the interested reader should have a closer look at D3.3 [19]. We should just highlight here the main trend that can be derived from this review. This relates to the fact that the monitoring frameworks for service-based applications become quite sophisticated and can support many levels of abstraction. This is also the case for our previous work [69] which covers many of the levels that the current line of work in this project focuses on. We should note, however, that our previous work, in comparison to the others, does rely on considering metric hierarchies and not just independent metrics at each level; as such measurability gaps are easier to cover. In this respect, the work proposed in this deliverable can be considered as a continuation of our previous work, especially as it is able to build upon measurements that could be provided by the monitoring framework of our previous work as well as equivalently cover the way metrics can be defined via the use of the same semantic language, i.e., OWL-Q.

By following the same approach as in the state-of-the-art subsections of the previous chapter (Section 2.1), the following set of comparison criteria has been compiled: (a) *analysis types*: which KPI analysis kinds are supported; (b) *db type*: type of db used to store the information needed for KPI analysis; (c) *evaluation technique*: the technique used to measure KPIs; (d) *drill-down technique*: the technique used for KPI drill-down; (e) *evaluation flexibility*: system flexibility in the exploration of the possible metric space; (f) *level*: the BPaaS hierarchy levels supported.

Based on these five criteria, the following table has been produced depicting the respective comparison / evaluation results. As it can be seen, semantic dbs are do considered in more than half of the systems, signifying that their added-value is being recognised in terms of better linking information and enabling various form of reasoning. Moreover, more than half of the systems focus only on KPI evaluation. The approaches that do support KPI drill-down exploit two main techniques: (a) decision trees and (b) combination of metric & KPI hierarchies. The first technique is suitable when there are measurability gaps (disconnected metric trees) to be filled-in, while the second is suitable when connections between KPIs and metrics exist such that we can go down to more technical KPIs and then continue from there by exploring the respective metric hierarchies involved.

Approach	Analysis Types	DB Type	Evaluation Technique	Drill-Down Technique	Evaluation Flexibility	Level
[10]	all	relational	SQL queries	Decision trees	low	BP
[66]	all	relational	Formula-based aggregation	Decision trees	low	BP, SE
[13]	evaluation	semantic	Formula-based aggregation	-	low	BP
[67]	evaluation	semantic	WSML rules	-	moderate	BP, SE
[62]	evaluation	semantic	Formula-based aggregation	-	low	BP
[11]	evaluation	warehouse	OLAP	-	moderate	BP
Ours	all	semantic	SPARQL queries	Metric / KPI-based	good	all

Table 3 - Comparison table for the state-of-the-art in KPI Analysis

A variety of techniques is exploited in the approaches evaluated, spanning from SQL queries, OLAP and event-based metric formula calculation to WSML rules and SPARQL queries. We should highlight here that SPARQL queries can be more expressive, even with respect to semantic rules, as they: (a) allow different ways to link the underlying semantic information; (b) have similar grouping and aggregation capabilities with SQL queries; (c) they work on the conceptual level that is closer to human conception.

Our system seems to be one step ahead from the work in [67] and [11] with respect to evaluation flexibility as it does not only allow to map human-based formulations of metric formulas into SPARQL queries but also to experiment with the metric and condition context. By injecting into our system the respective capabilities derived from the OWL-Q KPI extension, it can also support exploiting various information sources, like metrics, service properties and external ones, thus enabling a better exploration of the metric space. In this respect, our approach is more complete and user-intuitive with respect to the other two systems.

Finally, only our system covers all levels. In fact only three our of seven systems do recognise the necessity to cover more than one level in the BPaaS hierarchy.

3.1.2 Best Deployment Discovery

Deployment reasoning is a process which involves the discovery of a deployment for a cloud application which fulfils certain non-functional requirements. As there is no connection between the usage and actual non-functional capabilities of the underlying resources, the non-functional requirements usually include only cost as it is difficult to correlate the selection of the resources with high-level application requirements at the software / component level. As such, it is not certain whether the resources selected will actually suit the user requirements. In fact, due to resource competition and underlying hardware differences between equivalent virtual resources, it has been

derived that: (a) VM offerings with equivalent capabilities map to different application performance levels; (b) in some cases, certain clouds or VM offerings, while suiting the resource requirements of the application and its components, they do not cover the high-level non-functional ones.

The above problem can have different kinds of solutions that could be enforced. One solution is through VM benchmarking [77] which can enable VMs to be classified in qualititative categories based on their actual resource capabilities. This can, e.g., lead to characterise a VM offering as medium in terms of computation capabilities and large in terms of main memory or storage capabilities. As such, the selection of VM offerings then becomes closer to reality. However, still the problem of correlating low-level to high-level capabilities is not solved.

Another solution can be related to application profiling [68] that can enable deriving correlations between metrics and service properties in different levels. However, this profiling leads to substantial testing and to the spending of great amount of resources as the potential solution space is quite large. The Daleel framework in [56] employs adaptive multi-criteria decision making (MCDM) to support the deployment of applications by considering two main optimisation criteria: IaaS service cost and application execution time. Adaptiveness in MCDM is achieved by exploiting machine learning techniques over application profiling data on one cloud. Apart from the aforementioned disadvantages of this category of approaches, the proposed framework considers only the selection of IaaS and not of other types of cloud services. The set of optimisation criteria considered by this framework is also quite limited.

A final solution which has been adopted in the context of the CloudSocket project is to learn the selection of IaaS and SaaS services based on the execution history of the applications / BPaaSes. This implies that initially some random selections can be performed that conform to the resources requirements of the BPaaS which are then however coupled with high-level non-functional requirement assessments. In this sense, by looking at the execution history, we can detect which selections have been deemed as bad, leading to a violation of the non-functional requirements, and which selections have been the most optimal, always respecting the requiements posed. The exploration of the execution history has also the benefit that it is not necessary that a BPaaS has been deployed before. In fact, if such a BPaaS is similar or equivalent to an already deployed BPaaS, then respective analysis / mining results over the latter BPaaS could be re-used for the former. This leads to saving time and resources which are otherwise needed in the other two aforementioned solutions.

As the exploration of the execution history can be considered as a kind of a learning approach, learning approaches in deployment reasoning have been proposed. In [73], a stochastic programming and learning approach has been proposed which learns from the selection history and avoids making the same mistakes for the same problem. In fact, this approach promises that in the end an optimised configuration is reached as the solution obtained for the same problem is continuously improved. One problem exhibited by the respective approach is that it still considers only the cost optimisation objective. In addition, this approach needs a series of learning cycles in order to find a suitable solution which is never guaranteed that is the optimum one. On the other hand, in the approach that we envision, even new problems can be solved via an optimal solution at once without requiring any series of solvings to achieve that, provided that they are similar or equivalent to already handled ones.

Several approaches seem to use a learning-based approach aiming at addressing different research problems: VM consolidation and VM placement. Potentially, the techniques exploited in these approaches and the respective ways that these techniques can be exploited could be modified / adjusted to apply them to the current research problem. The main issue would then be how easy would be to make the transition from one research problem to another one. In [21], a reinforcement learning agent-based approach is proposed with the main goal to optimise the energy efficiency in a data centre while guaranteeing the respective level of service delivered to users. The main scope of the approach lies in selecting VMs for migration from overloaded hosts to less loaded

ones. Within the same scope, the approach in [46] applies a fuzzy Q-learning approach for VM migration strategy selection by considering the selection problem as a dynamic decision making one. A self-adaptive learning particle swarm optimisation approach is proposed in [71], aiming at solving the problem of resource outsourcing for cloud providers, i.e., how VMs can be migrated to external cloud providers when the current cloud is overloaded by still guaranteeing the provider's profit and preserving the service level delivered to the customer. Such a problem can be considered as a special case of VM placement which expands the current capabilities of the cloud provider. An ant colony reconfiguration approach is proposed in [14] focusing on the VM placement problem. The respective technique exploited in this approach and the previous one are nature-inspired and promise to reach optimum solutions, especially after a series of rounds, by considering that ants can learn from the outcomes of the choices that they make. The main difficulty with the adoption of such techniques relates to their actual configuration which also includes respective optimisation functions that have to be appropriately set. Moreover, such techniques usually lead to the production of sub-optimal solutions or sometimes non-optimal ones, especially if the algorithm is trapped in local optima. Finally, in the current context, the authors of the latter two approaches consider just execution cost as the sole criterion to be optimised which is quite limited.

In [59], the QuARAM IaaS service recommendation system is proposed which supplies a set of IaaS services that satisfy the requirements and preferences of the user through the use of case-based reasoning. The quality of recommendations is enhanced via the consideration of user and monitoring feedback. A multi-criteria decision making approach is applied in the situation that the case base is limited while clustering is exploited to partition the huge solution space. The proposed approach is very close to our work as it attempts to consider historical feedback to improve the quality of the recommendations while it seems to consider both user and monitoring feedback while in our case only the latter form of feedback applies. However, that approach focuses on the service recommendation and not the service concretisation problem; the latter needs the derivation of mappings not only from application components to IaaS services but also from the whole application to a composition of IaaS services. In addition, that approach focuses only on IaaS service recommendation and neglects other types of cloud services.

SelCSP [74] is a constraint programming framework for deployment reasoning which attempts to optimise the risk of interaction in IaaS selection. The risk of interaction is derived from two main parameters: (a) trustworthiness, derived from user feedback based on the actual past user experience and (b) competence based on the transparency of the cloud provider's SLA guarantees. This approach seems to consider the provider reliability and actual user feedback to appropriately rate a cloud provider. However, in our case, this information is not enough as it needs to rely also on: (a) monitoring information as user feedback tends to be quite subjective; (b) correlation between high and low-level non-functional capabilities to enable reasoning over high-level requirements. In this way, while the respective criteria can be considered independent of the type of cloud service considered, the corresponding selection problem can be solved only for one type of service each time and only for external services. Internal services will still need to be addressed via some sort of application component profiling to derive the needed aforementioned correlations.

A cloud deployment option simulation framework called CDOSim was proposed in [24]. This framework employs a hybrid benchmark-oriented approach where actual measurements from the system, while running, are also taken into account. While simulation promises better deployment reasoning times, it does not guarantee that all solutions can be obtained. In addition, the actual feedback obtained from the actual application runtime is rather limited and cannot be easily exploited in a large scale to support the attainment of better deployment decisions. As also indicated by the authors, the respective metric set considered by the framework is currently limited and thus needs to be substantially expanded.

Our previous work [38] is able to discover the best deployments for cloud applications and their components. This work can also provide respective derivation facts even for applications which have never been executed, provided that they resemble ones for which execution histories have already been recorded. This can greatly enhance and

speed-up the deployment reasoning process as well as provide the potential for improving the existing deployment of an application. However, our previous work, as all the others, is not able yet to cover higher-levels of abstraction.

A comparison table depicting the evaluation of the state-of-the-art approaches considered is given below. This table comprises the following set of evaluation criteria, which were mainly devised by the authors of this deliverable: (a) *abstraction* from hardware capabilities: capacity to profile and cluster respective cloud services in a qualitative way to abstract away from corresponding hardware capabilities (e.g., number of cores); (b) *correlation* between low and high-level non-functional capabilities to enable estimating the influence of lower-lever cloud selection on the component / application performance; (c) *service type coverage*: coverage of different service types in the cloud computing stack apart from solely IaaS services; (d) *optimisation criteria number*: capability to take into consideration a great number of criteria to optimise; (e) *feedback exploitation*: capability to exploit both monitoring (previous execution history) and user feedback; (f) *technique* exploited for optimisation; (g) *optimality*: capability to produce solutions which optimally satisfy the user requirements posed.

Approach	**Abstraction**	**Correlation**	**Service Type**	**Optimisation Criteria Number**	**Feedback Exploitation**	**Technique**	**Optimality**
[56]	no	yes	IaaS	low	monitoring	MCDM, machine learning	yes
[73]	~	~	IaaS	low	monitoring	Stochastic programming & learning	sub-optimal
[21], [46]	no	~	IaaS	low	monitoring	Reinforcement learning, fuzzy Q-learning	yes
[71], [14]	no	~	IaaS	low	monitoring	Self-adaptive learning particle swarm optimisation, ant colony reconfiguration	sub-optimal
[59]	yes	no	IaaS	high	both	MCDM, clustering	yes
[74]	no	no	IaaS	low	user	Constraint programmin	yes

						g	
[24]	yes	no	IaaS	low	monitoring	Benchmarking, simulation	sub-optimal
[38]	~	~	IaaS	high	monitoring	Constraint & rule-programming	yes
Ours	~	~	IaaS, SaaS (~PaaS[8])	high	monitoring	Constraint & rule programming	yes

Table 4 - Comparison table for the state-of-the-art in Best Deployment Discovery - Cloud Service Selection

Table 4 shows that only two approaches prevail, ours and the one in [59]. However, our approach is able to cover additional service types apart from IaaS, thus catering for discovering best deployments for BPaaSes and not just single cloud application, and can potentially handle the correlation between high- and low-level capabilities by exploiting application profiling. On the other hand, it is able to only exploit user-based feedback while in [59] both types of feedback are exploited. The latter is an interesting research direction we intend to follow in the near future in order to handle within user non-functional requirements even metrics for which measurements cannot be automatically produced.

By considering each criterion individually, we can see that the first one is scarsely supported (two out of 11 approaches). This means that while benchmarking is a handy way to cluster cloud services, it has not been widely considered for IaaS selection and best BPaaS deployment. Please note, however, that some of the evaluated approaches are capable of potentially exploiting benchmarking results in order to operate over a more meaningful and suitable solution space.

The situation might seem even worse for the second criterion, where we can see that just one approach directly supports the correlation between low and high-level capabilities. However, some approaches do handle this criterion indirectly via learning, which could be a more suitable way of dealing with it when the user budget is limited and less resources can be available to support application profiling. In this sense, we actually see a continuously increasing interest in exploiting any kind of knowledge which can enable performing an informed rather than a blind IaaS selection.

It is apparent also by reviewing the respective literature that most research is focused on IaaS selection or best IaaS-based deployment of applications. This is also evident in the evaluation table where only our approach is able to handle additional types of cloud services. However, by considering that nowadays modern applications can comprise using and exploiting a variety of different types of cloud services, this major drawback of the state-of-the-art needs to be addressed accordingly. As such, we are happy that we are proposing one of the initial approaches towards addressing it.

We can also inspect both in the table (3 out of 11 in total where 2 out of 3 are own approaches) as well as in the respective literature that most of the approaches tend to focus on a rather limited set of non-functional terms to be optimised. This can be due to the respective technique employed, a limitation of the optimisation algorithm

[8] Best deployments over PaaS services will be supported once the CloudSocket prototype is able to handle the deployment of components also via PaaS services (see D3.4 [20] on this).

proposed or the corresponding domain being exploited. In any case, it is more suitable to always adopt an approach able to handle any kind of non-functional term, especially as domain-specific terms are as much as important as the domain-independent ones. Such an approach will tend to have a better applicability and practicality with respect to the rest of the approaches in the state-of-the-art.

Concerning feedback, it is actually satisfactory to see that most of the approaches exploit one type of feedback, the monitoring one. This is quite logical as usually monitoring feedback could be considered as more objective than user feedback. However, this can hold only for certain kinds of non-functional terms. Moreover, monitoring cannot always be exploited in order to provide measurements for some metrics due to, e.g., some technical limitations or the type of access over respective services. As such, it is actually endorsed that both types of feedback are exploited in a complementary manner. Unfortunately, we only see one approach moving on this direction which signifies the need to optimise related work over this.

A diversity of techniques are used by the approaches reviewed. This not only indicates the different ways that the respective research problem can be solved but also that some of the respective techniques can be complementary to each other, such that their combined use can lead to a better solution for this problem. In the end, it might be better to both use benchmarking, profiling, constraint programming and a kind of learning technique in order to more optimally address the current research problem at hand. This also represents another point of optimisation for the proposed approach, apart from the previous one (user feedback exploitation), which might be attempted to be followed in the near future. Actually, the proposed approach can be easily extended to incorporate such techniques; the main issue would be mainly to apply them accordingly over the current problem at hand.

Finally, we can actually see that many of the approaches are able to produce optimal solutions according to the techniques that they exploit. However, optimality can be sometimes restrained due to the kind of input captured by these approaches or to the way the respective optimisation problem is formulated. In this respect, the more aspects are covered, the more optimal the solution would be, also with respect to the user requirements posed. We believe that our approach behaves well over this, especially as it is able to consider different types of services simultaneously, but, as indicated above, there is still some potential for further improvement.

3.1.3 Event Pattern Detection

Event pattern detection is a research field on its own which has grabbed the researchers attention lately. This is especially true as this kind of analysis can enable correlating events and discovering patterns of correlated events that can characterise a specific abnormal situation. Different techniques have been proposed to realise this kind of analysis, where almost all have been drawn from data mining. One widely used technique is association rule mining where one of the most widely adopted algorithms is the a-priori one [1]. However, such an algorithm works properly only by setting one or more thresholds, like the minimum support one (minsup), such that a satisfactory level of accuracy for the rules mined can be attained. Some of the extensions of this algorithm have focused on other aspects, including the algorithm performance [6] and the optimisation of its accuracy [31].

Another technique exploited quite often in data mining maps to decision trees. This technique has been widely applied in both BP performance and service monitoring [66]. The main rationale is to attempt to derive why a KPI violation has occurred based on correlations of this violation with any kind of data, such as a measurement, that might lead to this violation. The decision trees represent a nice mechanism as it can be easily visualised as well as understood by humans. On the other hand, it has to be performed with caution by carefully selecting the appropriate dataset on which the decision tree algorithm can operate. Major performance issues might be exhibited when the dataset becomes quite big as the potential problem space is quite large.

Another quite interesting technique is called logic-based (data) mining [58]. This technique does not require the setting of any threshold as it relies on a pure (propositional) logic-based approach to infer the (association) rules. Based on this technique, a rule can be inferred if the positive support of this rule is greater than its negative support. A positive support maps to two main rule support cases: (a) the event pattern occurrence maps to the KPI violation occurrence and (b) when the KPI is not violated, the event pattern has not occurred, while the negative support maps to the two contradiction cases for this rule: (a) event pattern does not lead to the KPI violation and (b) KPI violation is driven by a different event pattern. A similar approach on this category relies on event-calculus [5] and promises a more efficient identification of patterns which is more scalable to deal with large event streams.

In the context of event pattern detection for service-based applications that does consider all possible levels of abstraction, our previous work [70] has adopted the aforementioned logic-based approach. In that work, event correlation based on service dependency models has been followed in order to filter events and inspect only the potential sets of events which do make sense for leading to the generation of the respective critical event, such as the violation of a KPI or an SLO, and not the whole event space. This actually leads to major savings in the execution of the respective event pattern detection algorithm. Apart from this kind of optimisation, the added-value of our previous work is that apart from deriving event patterns, it can also map them in a semi-automatic manner to one or more adaptation strategies that can address the examined problematic situation. One of these strageries is then selected, based on preference-based ranking, in order to generate a respective adaptation rule that can drive the adaptive execution and provisioning of the BPaaS under examination. This added-value feature can provide significant support to the adaptation rule modellers, as they will not start from scratch by relying on the event pattern detected but actually either adopt or slightly modify an adaptation rule that is provided by the system. Based on the above advantages, it is quite natural to build over this work in order to increase its automation level as well as improve the confidence and accuracy in the generation of the adaptation rules.

3.1.4 Process Mining

Process mining is a wide field in process management which deals with the mining of execution logs of BPs in order to perform different kinds of analysis over them which can produce added-value knowledge leading to process improvement potentials. There have been different kinds of analysis that have been proposed which map to respective categories of process mining algorithms.

The first category attempts to (re-)construct the process model from the execution log to cater for two main scenarios: (a) construction of a missing process model; (b) comparison and conformance of constructed process model from the designed one. It includes approaches which are either frequency-based or genetic algorithm (GA)-based. A common base for all frequency-based algorithms is the alpha algorithm [63] which attempts to derive the relationships between pairs of process tasks by considering the individual and joint frequencies of these tasks in the process log. As noise can be involved in an execution log, extensions to this algorithm have been proposed, attempting to reduce its effect and thus produce more accurate mining results. From these algorithms, the one which is currently mostly utilised is the heuristic miner [64] from the ProM framework[9]. For the latter algorithm, specific extensions have also been proposed, which focus on attempting to resolve some of its main deficiencies. In [16], the authors focus on an extension that attempts to resolve the validity and the completeness of the mined process model. Validity is addressed via the better mining of loops, while completeness is handled via the update of the relative to best threshold and the extension of the all-tasks-connected heuristic to cover also loops. In [7], the authors extend the heuristic miner to handle time intervals which maps to a more correct and fined-grained approach to process mining, in case the respective process log covers the required information (i.e., start and end of process tasks instead of solely the end or start).

[9] www.promtools.org/

The main added-value of a GA-based mining algorithm [4], apart from its capability to handle noise, hidden activities and non-free choice constructs, is that it applies a global-based search approach instead of a local one (applied by, e.g., a heuristic mining approach). While a local-based search builds the process model in a step-by-step process by relying only on local information, a global-based search evaluates the fitness of a process model according to all traces in the process log, thus resulting in a globally optimum process model.

The second category includes algorithms which attempt to infer decision rules for decision points in the process. Such decision rules will then map to decision tasks which will substitute the manual tasks currently incorporated immediately before these decision points to cater for the automatic derivation of the values for the parameters involved in the decisions by human experts. The respective algorithms attempt to re-use algorithms and techniques from machine learning in order to derive these decisions rules, such as decision trees [54]. The main rationale shared by all of these algorithms is that they can actually abstract the process (log) model into a data mining task in which the current situation is modelled by the respective values that the process variables take directly before the actual decision point and the respective decision is the one that has to be learned from these variables. An approach that is able to discover overlapping decision rules that best fit the data available at the expense of accuracy is proposed in [45]. The proposed approach first constructs a decision tree from the process log. Then, for each leaf node, the wrongly classified instances are used to construct a new decision tree. As a result, in the end, a disjunction of rules is generated that best fit the process log. This mining approach is better in handling incomplete information in the log as well as in cases where the decision rules are conflicting, while it can also cater for cases where a non-mutually exclusive choice is employed as the respective process model decision construct. The authors show that while precision in general might be hampered slightly, the fitness is usually better and there are also cases where the fitness and precision of rules is better than a single rule-based decision mining approach.

Usually, the outcome of a decision mining algorithm is a set of branching conditions which include the comparison of a variable to a certain constant. The approach in [17] combines invariant discovery and decision tree learning techniques in order to produce a more sophisticated form of branching conditions which include atoms with linear equations or inequalities over multiple variables and arithmetic operators.

The approach in [27] employs decision tree-based learning techniques to derive relationships between the process context, the process outcomes and path decisions from which respective decision rules are derived. The novelty of the approach is that it does consider all these three different types of information to perform the inferencing, such that the situations underlying different process executions as well as they way the business goals have been satisfied in these situations are taken into account in the decision rule derivation process.

The third major category includes algorithms which attempt to derive interesting organisational information. Such information could take the form of organisation models or social networks (cooperation patterns between employees). This information can enable managers to understand and optimise the organisation model and structures of their organisation. It can also enable the creation of suitable collaboration and communication mechanisms to increase the employee cooperation and productivity. Three types of organisation mining exist [60]: (a) organisational model mining; (b) social network mining; (c) information flow mining between organisational entities. In [60], four techniques for organisation mining are proposed and one technique for social mining and information flow mining.

The organisation mining techniques focus on the discovery of task-based teams and case-based teams. Task-based teams include people with similar skills and expertise which are usually assigned the same set of tasks. On the other hand, case-based teams comprise people with complementary skills which operate over the same case. Three out of the four proposed techniques focus on team-based discovery. The simplest technique called *default mining* operates over the process log and collects all the assignments that have been performed for the process tasks. The second technique focuses on mining not individuals but also other types of organisational entities like

roles. It first calculates the profile of each individual and then calculates the distance between pair of profiles by using different alternative distance measures, like Hamming distance and Pearson correlation coefficient. When the distance between two profiles is less than a pre-defined threshold, then the two profiles are merged into the same cluster. In this way, a clustering of the individual profiles is achieved where each cluster (potentially) maps to a particular role. The third technique is more suitable to derive hierarchical organisation models rather than flat ones as in the previous two techniques, where an agglomerative clustering technique is exploited.

The technique focusing on case-based team discovery employs joint case metrics, i.e., metrics which assess how often two or more individuals participate in the same case. Based on these metrics, a respective network is constructed on which a threshold is applied to filter non-significant arcs. As such, sub-networks connecting two or more individuals will map to a respective organisational entity. Instead of threshold-based filtering, centrality-based node filtering can be employed in order to address the issue where an individual works as a hub between different networks.

The social network derivation technique relies on the use of several alternative metrics. The basic idea in this technique comprises exploring relationships between individuals within the same case based on the respective metric selected and constructing a respective network out of them provided that appropriate support for that relationship exists in the process log. For instance, two individuals might be connected with a work hand-over relationship. The respective metrics can regulate the population of the social network by considering also either direct or indirect relationships. After the construction of the network, various kind of analysis methods can be applied on it focusing on deriving its density, causality, cohesion and equivalence.

In [60], the information flow mining technique first produces a social network (see previous paragraph) and then constructs an organisational entity network by replacing individuals with the organisational entities to which they map. The weights to arcs map to the sum of weights involved in the original social network. The relative information flows can then be structured via applying a respective metric.

Orthogonal to the above classification of the process mining algorithms is the aspect of semantics. As advocated in [3], semantics can enhance the accuracy and robustness of the process mining algorithms while, in the other direction, they can be employed to generate or enhance existing ontologies from event logs. Accuracy can be enhanced by dealing with the label heterogeneity in activity names within process logs by mapping them to ontology concepts from an activity ontology. Logically speaking, the same benefits can be obtained by doing similar transformations over other information elements in the logs to support other types of mining, such as decision and organisational mining. In [75], it is also stated that semantic process mining can also enable the production of hierarchical process models with many abstraction levels which can be suitable for visualisation and summarisation purposes. In that approach, the combination of semantic process mining and semantic process planning is proposed to cater for incomplete event logs. The former can be used to construct an incomplete process model based on the process log, while the latter for the completion of that model according to the overall process goals that have been defined.

Ferreira and Thom [23] propose a semantic process mining technique which operates on semantically annotated logs in order to discover workflow activity patterns (WAPs). Such patterns represent common business functions usually occuring in BPs, such as activity execution, decision mining and approval. The respective technique employed does not exploit a normal process mining algorithm. On the contrary, it employs reasoning over a semantic KB to derive the WAPs that can be attributed to a certain BP. Then SPARQL queries are exploited to discover the patterns mapping to a BP while the execution log can be utilised also for validation purposes to check whether the order of derived WAPs in the BP model is correct.

Domain semantics are exploited in [61] to address the issues of infrequent traces and log data contributed by ad hoc changes. In that approach, a query-based data collection method is first employed to produce the event logs

and then a data cleansing method based on semantic log purging is executed which exploits domain constraints to filter incorrect log entries. Such domain constraints can also be exploited in order to examine which entries in the log violate the corresponding domain semantics.

As it can be seen from the above analysis, there are already quite capable process mining algorithms per each category reviewed. To this end, the actual goal of the research to be performed might not be to propose a new algorithm but to extend the existing ones in order to fix one or more their respective drawbacks. In particular, towards this direction, we will attempt to make some prominent algorithms semantics-aware in order to raise their accuracy levels by exploiting the respective annotations already supplied in the BPaaS bundle. We could also provide a new algorithm, only in case that this is necessary and could provide an added-value to the broker. Some initial ideas are indicated in Section 3.7 which focus on the organisational aspect.

3.2 BPaaS Evaluation Environment Architecture

The scope of this section is to analyse the overall architecture of the BPaaS Evaluation Environment research prototype without providing architectural details over main components / modules. Such details are to be provided in the next sections which focus on the respective analysis of each main component / module in the architecture in terms of the actual functionality that is being delivered by it. Please note that one component may deliver not one but many functionalities. In this case, each section will unveil only those parts of the component architecture which are relevant.

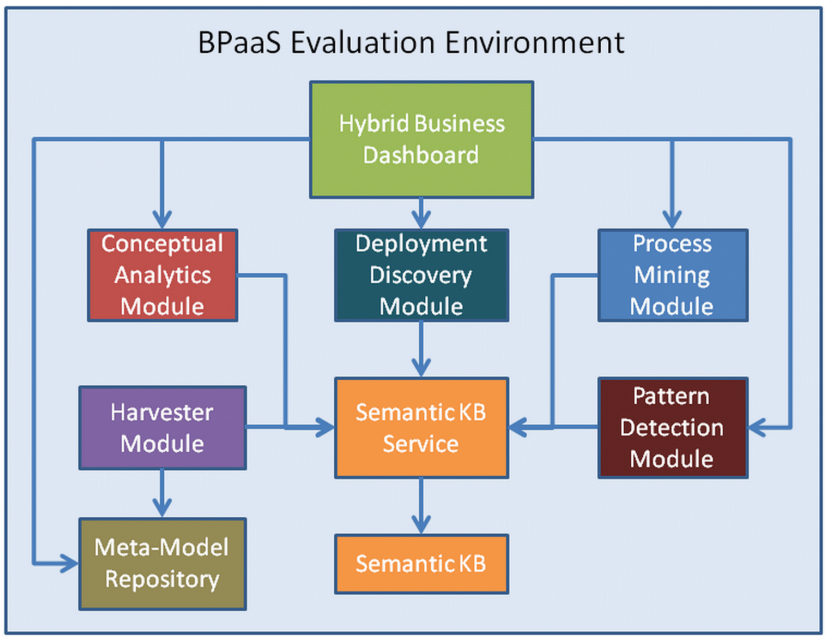

Figure 3 - Overall Architecture of the BPaaS Evaluation Environment

The current architecture of the BPaaS Evaluation Environment research prototype is depicted in Figure 3. As it can be seen, it involves the usual levels of user interface (UI), business logic and database, while it comprises 9 components (1 UI, 6 business logic and 2 database): (a) The *Hybrid Business Dashboard* (UI level) is responsible for initiating as well as visualising the results of the different kinds of analysis that are being supported by this environment. In case that multiple functionalities are needed at once, this component is also responsible for executing the orchestration of these functionalities which could be delivered by potentially different architecture components; (b) the *Conceptual Analytics Module* is responsible for delivering the functionality of KPI analysis; (c) the *Deployment Discovery Module* offers the best deployment discovery functionality; (d) the *Pattern Detection Module* is responsible for delivering the functionality of event pattern detection; (e) *Process Mining Module* is responsible for performing the different kinds of mining algorithms that will be realised or adopted by this

environment; (f) the *Harvester Module* is responsible for drawing information from various information sources within the CloudSocket prototype (i.e., other environments / components), semantically enhancing it and linking it and finally storing it in the underlying Semantic Knowledge Base (*Semantic KB*); (g) the REST interface (named as *Semantic KB Service*) to the latter KB which enables the management and querying over the information stored in this KB; (h) the *Semantic KB* itself which is realised via a semantic Triple Store; (i) the *Meta-Model Repository* from which information about BPs, workflows and KPIs can be drawn for harvesting and visualisation purposes.

3.3 Information Harvesting and Linking

Any kind of analysis can only benefit if the appropriate information exploited is already in place. If only one or a very few types of analysis are to be supported by a small and not quite distributed system, the collection of such information can be easy. However, for a big, distributed system aiming at supporting a variety of analysis tasks which span heterogenous information supplied by different information sources, then the collection, normalisation, linking and storage of such information is a quite hard task.

In the context of the CloudSocket project and the evaluation of BPaaSes in particular, different kinds of analysis are envisioned: (a) evaluation and drill-down of KPIs; (b) best BPaaS deployment discovery; (c) event pattern detection (leading to violation of KPIs/SLOs); (d) process mining. Each analysis kind requires different information to be in place in different levels of detail. For instance, KPI analysis requires the knowledge of the whole KPI metric derivation tree as well as the deployment / dependency tree for the BPaaS component actually being measured. On the other hand, process mining just requires the knowledge of which tasks of a BPaaS process have been executed, at what time and by whom as well as their semantic annotations, in case they exist. Moreover, each type of information is supplied by a different component or environment in the CloudSocket prototype architecture. Deployed workflow information can be supplied by the *Workflow Engine*, cloud service information is stored in the *Registry*, deployment and monitoring information is stored in the *Cloud Provider* and *Monitoring Engines*, complete information about users and their roles is supplied by the *Marketplace*, while BP, workflow and KPI information can be drawn from the *Meta-Model Repository*.

Fortunately, the architecture of CloudSocket prototype relies on SOA and each component / environment offers a respective (usually REST) service from which the desired information can be drawn. This is true actually for most of todays components or systems which could substitute or replace the components in the CloudSocket prototype architecture. For example, modern Workflow Engines do provide REST APIs / services via which workflow / process event and monitoring information can be retrieved. As such, this actually enables to use almost the same uniform mechanism in order to acquire the needed information. "Almost uniform" is stressed to cover cases where different service technologies (i.e., REST vs SOAP) are employed, mapping to different kinds of clients to be used for the harvesting of the underlying information offered. Again, in the context of CloudSocket, REST services are the norm, so interface uniformity in the end is guaranteed.

Interface uniformity also enables to obtain information in a uniform representation (e.g., JSON). However, interface uniformity is just one solution to a small issue in the overall information harvesting challenge. Even if one mechanism is used to obtain information from different sources, there is first the need to inspect each interface offered by the source, understand what should be the right methods to call and in what order and then fetch the required information according to that order in order to process it. By involving a variety of information sources, this is actually a quite involved task which can be hampered sometimes by the lack of (API) documentation for a particular service.

While the previous issue just relates to the effort that has to be put through in order to obtain the needed information, the main challenge is then how to process, normalise, semantically enhance and link it. The Evaluation and OWL-Q (with KPI extension) ontologies greatly facilitate the processing, normalisation and linking

tasks as they were setting a particular way via which the information should be structured and linked. Moreover, they were designed to cover the modelling of all the information required for all the kinds of analysis envisioned. Thus, any lack of information which can be modelled should reflect the inability to find such information from the available sources.

However, the semantic enhancement of the information is not an easy task. To this end, two different but even complementary directions could be followed: (a) information is already assumed to be semantically annotated; (b) the information can be semantically annotated as much as possible in an automatic manner based on the availability of respective ontologies. As an initial approach, we follow the first direction. This is due to the fact that some environments do provide support for already semantically annotating the information needed and others could be enhanced to do so. In particular, the BPaaS Design Environment already possesses the mechanisms in order to annotate BPs and workflows according to both functional and non-functional aspects. The BPaaS Allocation Environment could also be enhanced to support semantic annotation mainly in the context of technical KPI modelling. By relying then on a semantically annotated bundle, as the main product of the cooperation between the latter two environments, then the sole task of the BPaaS Execution Environment would be just to be able to appropriately link the information, whenever this is possible. For instance, concerning the BPaaS monitoring, a respective measurement should point to the name of the concerned metric from which then the semantic annotation can be obtained from the BPaaS bundle. Of course, if there is a need for deriving additional annotations for other elements in a BPaaS bundle, then the second direction could be followed. Such a need could also be apparent if the information which has the possibility to be annotated is not always annotated in the context of one or more BPaaS bundles.

The above analysis has also highlighted the need for appropriate linking of the information. Such a linking is not always straightforward, especially if we consider that each information source is independent from the other. For instance, by continuing with the measurement example, as a metric might be re-used in the context of multiple BPaaS or even in the context of multiple instances of the same BPaaS, the measurement should also be mapped to the BPaaS instance concerned. In addition, there should be a linkage also to the respective BPaaS component / object being measured. If such a linkage is not enforced, then there should be other ways in which it has to be derived. In order to assist in the task of information linking, T3.3 actually has led an initiative via which a respective identification mechanism across particular environments was enforced as well as linking requirements for particular types of information were applied. This has actually guaranteed that missing connection points between different information aspects were actually covered. This was particular true for the case of measurements, where now they map both to the id of the metric being measured as well as the id of the BPaaS object on which the measurement is performed. The latter can then be used to also identify the actually BPaaS instance being concerned, via exploiting deployment / dependency information.

In the following figure, we explicate the sources of information and how they apply to the respective evaluation ontology parts (along with small parts from OWL-Q ontology) in order to showcase how information is covered and in which way it can be linked. The colour encoding is the following: (a) red colour is used to denote information collected from the *Cloud Provider Engine*; (b) grey colour denotes information from the *Workflow Engine*; (c) blue colour denotes *Registry* information; (d) green colour signifies information from the *Marketplace*; (e) purple colour is used to denote the external ontology concept(s) from the FAO ontology.

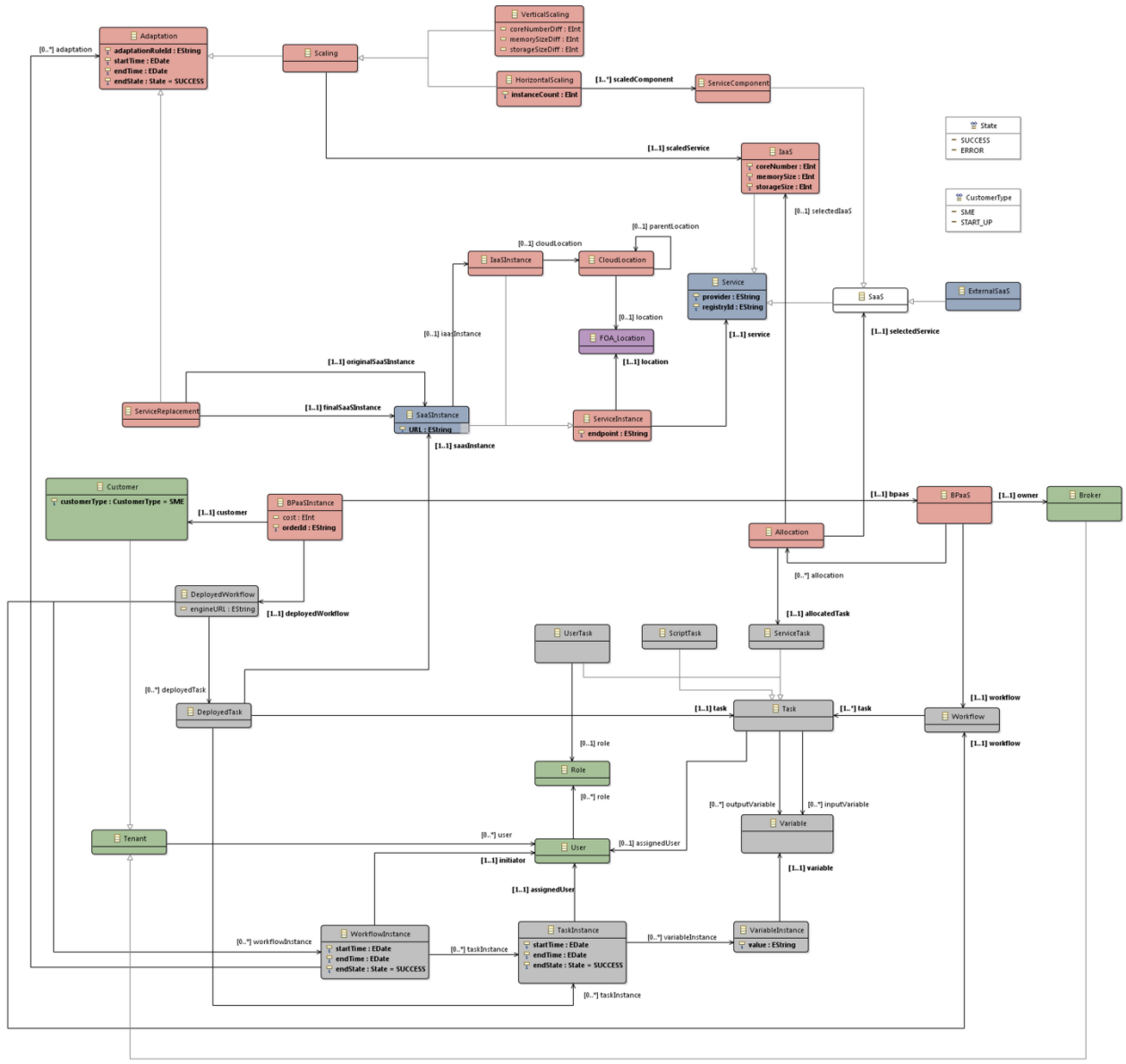

Figure 4 - Coloured UML class diagram of the Evaluation ontology

Apart from the need to link the information, another issue that had to be dealt with concerned the way the information should be exploited in the different kinds of analysis. This actually reflected the way information can be fetched from the sources and at what timing. There were three actually directions on this which could be followed: (a) information is fetched in a timely basis; (b) information is obtained via a call-back mechanism (e.g., publish-subscribe); (c) information is fetched on demand. The last direction was discarded based on the rationale that the analysis tasks would then take a greater amount of time to finish such that it was better that the information to be exploited should already be in place. From the first two directions, the first one was finally selected by considering that the second direction might lead to increased communication between the BPaaS Execution and Evaluation Environments. Moreover, as the kinds of analysis supported concern longer rather than small periods of time which do make more sense at evaluation rather than execution time, then information harvesting does not need to be very frequent. After all, if, for instance, another measurement for an instance-based metric is not fetched at the time it is generated, this will not affect greatly the respective computation of a high-level metric which is derived from a particular tree which has as a leaf this instance-based metric. Based on the above rationale, the information was finally selected to be fetched every 5 minutes.

The current realisation of information harvesting has resulted in the production of the Harvester module as a blueprint whose internal architecture is shown in Figure 5. This module is not offered as a service as is the case in the other modules of the overall BPaaS Evaluation Environment architecture (see Figure 3). This is due to the fact that the actual functionality of this module is actually preparatory for enabling the respective core capabilities of the rest of the modules in the evaluation environment's architecture. The Harvester module's architecture comprises components which map to respective fetching capabilities from different information sources. As such, the number of such components coincides witht the number of the information sources exploited. The respective components are named as follows: (a) *WF Engine Extractor*; (b) *Registry Extractor*; (c) *Deployment Extractor*; (d) *Monitoring* Extractor; (e) *Marketplace* Extractor. The *WF Engine Extractor* is able to extract modelling and execution information about an actual BPaaS workflow that has been deployed as well as for the instances of such workflow that are or have been run by the respective clients that have purchased it. The *Registry Extractor* is responsible for fetching additional information about cloud services (e.g., which is the SaaS service involved in the realisation of a BPaaS workflow task and which are its annotated input and output parameters). The *Deployment Extractor* extracts deployment information from the *Cloud Provider Engine* about which BPaaS component instances have been created and on which IaaS instances have been deployed. The *Monitoring Extractor* extracts monitoring information from the *Monitoring Engine*. *The Marketplace Extractor* extracts information about users and their roles.

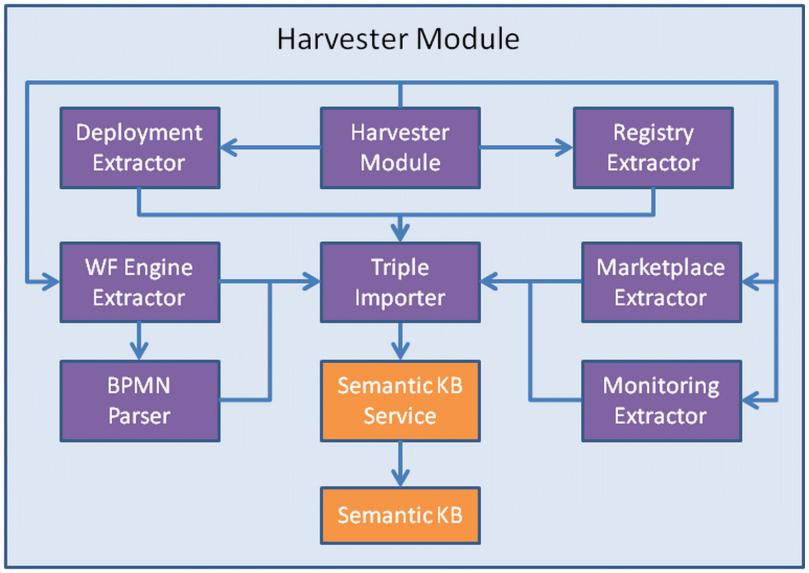

Figure 5 - The internal architecture of the Harvester Module

As workflows can be fetched from the *WF Engine Extractor*, the *BPMNParser* component is responsible for processing them. In addition, the *Triple Importer* is the actual component responsible for storing (linked) information in the underlying Triple Store (*Semantic KB*) via the use of the *Semantic KB Service*. Before being stored, the information is structured according to the respective ontology schema, while appropriate URIs are given to resources (i.e., subjects or objects) via exploiting the *Resource Naming Service* component. Finally, the *Harvester* is a thread-based component (run at the particular time interval designated in the previous paragraph) which encapsulates the core orchestration logic for information fetching and linking.

We should now highlight two major issues with respect to the information harvesting related to duplication checking and linked resource identification. Concerning duplication checking, the *Harvester* is smart enough to avoid performing unnecessary computations that may result in the production of duplicate triples. This is done in cooperation with the *Triple Importer* via performing suitable queries over the Semantic KB which involve checking whether triples about some particular BPaaS components (actually instances of classes from the Evaluation and OWL-Q ontologies) have been already stored. Please note that in the case of measurements, things are easier

as there is no need to perform any kind of query but just check the *dateTime* of the measurement. If it is greater than the *dateTime* of the latest fetching, then it should be processed and stored.

As far as resource identification is concerned, as already stated, the *Resource Naming Service* is the component responsible for producing the URIs of the information resources. This URI production is however not easy and straightforward as one would expect. This is due to the fact that the final fragment of the URI, mapping to the actual identifier of the information resource, should be always unique. As such, unique identifiers are produced according to a particular scheme. This scheme includes the following pattern: (<Class_for_Parent_Dep_Node>_X_)?<Class>_Y. This scheme reveals the following: (a) in some cases, the name of the ontology class to be instantiated along with the id of the instance (marked as Y) (id maps to the identification of this information object by the respective information source) can be unique and this will constitute the actual identifier of the information resource / class instance. For example, logically speaking, a service would always have a unique id in a registry. As such, a particular service (as an instance of the *Service* class) will be mapped to the pattern Service_Y, where Y is the registry id of this service; in other cases, the <Class>_Y pattern alone cannot guarantee that the respective identifier is unique. In that case, we need to go to a higher level in the dependency tree, i.e., the actual container of the instance, and concatenate its own pattern with that of the actual instance. For example, a task can have the same id in different workflows but this does not necessarily mean that we are dealing with the same task. In this case, we would need to refer also to the id of the workflow encompassing this task in order to make the identifier of the latter unique. In this case, the respective pattern will become: Workflow_X_Task_Y where X is the id of the workflow and Y the id of the task.

Apart from setting appropriately the id fragment for information resources, the actual namespace would also differ depending on the corresponding broker. In this sense, the following overall pattern scheme is followed in the end for the whole URI of the information resource at hand: "http://www.cloudsocket.eu/evaluation/" + <broker_name> + "#" + (<Class_for_Parent_Dep_Node>_X_)?<Class>_Y>, where <broker_name> is the name of the broker mapping to the respective BPaaS to which the information resource belongs. As such, the respective evaluation environment supports multi-tenancy as it is able to handle not just one but multiple brokers. In fact the namespace part of the URI would map to the actual (RDF) graph where all the information relevant for a broker will be stored. In this sense, after appropriately identifying the broker concerned, then the analysis in the evaluation environment will only focus on the RDF graph that belongs to this broker. This not only actually caters for information provenance with respect to who owns this information but also enables creating semantic information partitions such that the respective queries are posed only on them, thus saving valuable analysis time.

3.4 KPI Analysis

The envisioned forms of KPI analysis concern the evaluation as well as the drill-down of KPIs. KPI evaluation can concern either for the current value of the KPI metric or can be performed against the whole execution history of a BPaaS. In the latter case, the history can be shrinked down to a specific time period given by the user. The drill-down of KPIs involves providing the evaluation of KPIs on which a current KPI depends. Such form of analysis is useful when we desire to check the root causes of a specific high-level KPI violation by drilling-down until the technical level. In fact, as it can be inferred, this drill-down can take the form of an evaluation tree which corresponds to the metric dependency tree for the current KPI metric at hand.

No matter what is the KPI analysis form, a core functionality mapping to the evaluation of a KPI metric needs to be realised. In our case, we have selected a semantic approach for this realisation for two main reasons: (a) the accuracy of the evaluation will be higher; (b) semantic linking enables us to explore the actual space of possible computation formulas for KPI metrics. In fact, the second reason actually reflects the current practice in KPI

evaluation where there might be some KPI metrics which can be fixed in advance (e.g., cross-domain metrics) while many other metrics need to be computed based on the respective knowledge and expertise of the (process performance) evaluator / exprert. We should stress here that in contrast to other forms of measurement storage and computation / aggregation, the semantic linking enables us to have a rich connection between different information aspects that can be useful in the production of metric formulas. On the other hand, measurement system alternatives, such as Time Series Data Bases, require an apriori design of the measurement space and are not able to actually allow advanced forms of information linking and aggregation. In fact, some of such systems are not even able of making joins over different measurement rows.

By relying of a semantic approach and the adoption of the Linked Data (LD) technology (see representation languages exploted in Chapter 2), the most intuitive way to express metric formulas would be via SPARQL queries. The main difficulty here lies on the fact that SPARQL queries require deep knowledge about LD technology as well as great expertise in SPARQL query modelling. However, a BP performance evaluation expert might not possess such knowledge and expertise. He/she would rather prefer to specify the metric formula in a simplified mathematical language. This latter observation has led to the requirement to actually transform the specification of the KPI metric, conforming to the KPI description language (e.g., OWL-Q in our case), into a specification of a SPARQL query.

This transformation was not an easy task to perform. First of all, it relies on the kind of metric involved. We actually distinguish between two types of metrics: (a) *customer-specific*, i.e., pertaining to a particular BPaaS instance that has been purchased by the BPaaS customer; (b) *broker-specific*, i.e., pertaining to an overall view of the actual performance of the respective BPaaS offered. Customer-specific metrics have as the measurement space all the measurements that have been produced for the respective BPaaS instance of the customer. On the other hand, broker-specific metrics have a broader measurement space which includes all the measurements for all the instances of a BPaaS, spanning across all the customers that have purchased this BPaaS.

Secondly, the transformation is hardened based on the kind of dynamic evaluation enabled by the BPaaS Evaluation Environment. This actually means that the user is given freedom to experiment with formulas, types of metrics, evaluation periods (schedule & windows) and periods of history. As such, the system should not attempt to physically store the respective evaluations as this will lead to a quite large storage space which is actually heterogeneous in content also with respect to a single metric, by also considering the view that a metric actually represents just a computation while its context can vary. As such, SPARQL query evaluations are never reflected in the actual *Semantic KB*. This signifies that for the computation of KPI metrics, we should go down to the level of the *BPaaS Execution Environment* which is the level which provides measurements which have been physically stored in the *Monitoring Engine* and which have been retrieved, semantically enriched and stored in the *Semantic KB*. As such, due to this limited materialisation, it is not possible to directly exploit any other kind of existing computation, such as KPI metrics from which a high-level KPI metric is computed.

Thirdly, the mapping to respective SPARQL operators is not always straightforward. In fact, we have found that while SPARQL suffices for advanced mathematical expressions which involve well-known mathematical operators, it lacks quite significant statistical operators which are already provided by other measurement systems, like certain TSDBs (e.g., InfluxDB[10]). This has actually highlighted the need to realise there operators. Please note that this issue does not significantly impact the actual transformation process per se but actually the respective possibility for metric formula expression.

By considering the above three issues, the respective transformation approach has been incarnated in the form of a specific algorithm. This algorithm, depending on the respective input provided, attempts to construct

[10] https://influxdata.com

dynamically the SPARQL query that has to be issued in order to derive the corresponding metric measurement. As such, this algorithm is now explained in detail without of course entering low-level technical details.

The algorithm comprises two main steps: (a) metric formula expansion and (b) actual transformation to SPARQL. Formula expansion involves the recursive substitution of component metrics for which measurements are not stored in the *Semantic KB*. This substitution maps replacing each "problematic" component metric with the actual formula from which this metric is derived. Here we make the assumption that measurements for raw metrics always exist in the *Semantic KB*. An example of this transformation is the following: The average availability metric AVG_A for the whole BPaaS workflow can be computed from the formula MEAN(RAW_A) where RAW_A represents the raw / instance-based availability metric for this workflow. RAW_A is not stored in the Semantic KB. As such, it is further expanded into its respective formula UPTIME/TOTAL_OBSERVATION_TIME, where UPTIME is a raw uptime metric and TOTAL_OBSERVATION_TIME is a constant representing the observation time period. In this sense, the resulting expanded formula in the end will be: MEAN(UPTIME/TOTAL_OBSERVATION_TIME).

The actual transformation relies on transforming the following main ingredients into corresponding constructs in the SPARQL query: (a) the measurements for the metrics in the expanded formula are mapped to SPARQL variables; (b) triple patterms are generated to reflect measurement interlinking by considering the object being measured (e.g., whole workflow in the example) and the actual BPaaS instance involved; (c) the evaluation periods map to the grouping of measurements via SPARQL group by clauses; (d) history periods map to filtering criteria for the measurements over the respective *dateTime* of the measurement. The triple patterns generated for ingredient (b) explicate the following: (i) the metrics involved in the formula and the measurements being produced for them; (ii) the actual component being measured by the measurements; (iii) the actual BPaaS instance being involved which is related to the actual component being measured.

To explain the actual transformation procedure, we can continue with the previous example and make the following assumptions: (i) the availability metric should be calculated every 1 hour; (ii) the history period is 1 day; (iii) the uptime metric is calculated every second while the raw availability every 1 minute. Based on these assumptions, we can see in Figure 6 the SPARQL query generated. We will now explain this figure by especially focusing on those parts which are related to the respective information that is being mapped from the metric definition and from the BPaaS dependency model.

Lines 1-2 indicate the prefixes of the two ontologies that are being exploited, i.e., OWL-Q and Evaluation. Line 3 specifies the formula to be calculated which is the average of a division where the nominator maps to the raw uptime metric measurement value and the denominator to the observation time as a constant. Please remember that uptime is calculated every second while raw availability every 1 minute. In this sense, the observation time is 60 (1 minute maps to 60 seconds). Line 4 explicates the RDF graph from which the respective information for the query evaluation can be resolved / obtained. As it can be seen, we create individual RDF graphs per broker (BWCON in our case).

Lines 5-9 signify a set of triple patterns which link the uptime measurement to: (a) the Uptime metric; (b) its actual value which is used in the formula of Line 3; (c) the actual *dateTime* where this measurement has been produced; (d) the URI of the actual object being measured (a workflow instance in our case). While these lines guarantee that we operate over Uptime measurements, they do not provide the appropriate connections to other major information aspects, such as which BPaaS is actually concerned.

Based on the above issue, Lines 10-13 make the needed connections. Line 10 connects the current BPaaS under investigation (the well-known Christmas Cards example) to one of its instances. Line 11 links this BPaaS instance to a deployed workflow. Line 12, which is currently commented, would link this instance to a specific client that has purchased it, in case we are dealing with an customer-specific metric. Finally, Line 13 maps the

deployed workflow to the actual instance of the workflow that has been measured. As such, Lines 10-13 actually make the appropriate connection from the object being measured to both the BPaaS instance involving it and the BPaaS that is under investigation. We should highlight that Lines 11-13 can be differentiated based on the kind of object being measured. For instance, if a task instance is measured, then we need to add another triple pattern indicating that the BPaaS workflow instance includes this task instance.

Line 15, which is commented, provides a SPARQL FILTER constraint which can be used to restrain the history period under investigation. As it can be seen, the *dateTime* of the measurement is mapped to two simple constraints which indicate that this *dateTime* should be greater or equal to the low bound *dateTime* of the considered period and less than or equal to the upper bound *dateTime* of this period. The commendation of this line signifies that we are exploring the whole computation / evaluation history of the KPI metric.

Finally, Line 17 provides a grouping statement where the last sub-group is the one which maps to the evaluation period of the KPI metric (per hour). This statement groups first the results according to the month, then according to the day and finally according to the respective hour, by considering that the BPaaS under consideration has been purchased the last year (otherwise, we would need to add an additional grouping over the year).

```
1: prefix eval: <http://www.cloudsocket.eu/evaluation#>
2: prefix owlq: <http://www.ics.forth.gr/ontologies/owlq#>
3: SELECT (AVG(?uptime / (3600)) AS ?value) (MAX(?uptime_meas_ts) AS ?date)
4:      from <http://www.cloudsocket.eu/evaluation/bwcon> WHERE {
5:  ?uptime_meas a owlq:Measurement;
6:      owlq:metric owlq:Uptime;
7:      owlq:value ?uptime;
8:      owlq:timestamp ?uptime_meas_ts;
9:      owlq:boundElement ?obj.
10: eval:ChristmasCard ?bpaasInstance ?bpaasInt.
11: ?bpaasInst eval:deployedWorkflow ?depWf.
12:     #eval:customer eval:Customer1;
13:  ?depWf eval:workflowInstance ?obj.
14:
15:    #filter (?uptime_meas_ts <= "dateTime2"^xsd:dateTime && ?uptime_meas_ts >= "dateTime1"^xsd:dateTime)
16: }
17: GROUP BY (MONTH(?uptime_meas_ts) as ?month, DAY(?uptime_meas_ts) = ?day, HOUR(?uptime_meas_ts) as ?hour)
```

Figure 6 - The snapshot of the SPARQL query generated

The derivation of the actual information for KPI drill-down can then proceed by the following algorithm which is an extended version of the transformation algorithm that takes into account the whole derivation / dependency tree for the current KPI metric at hand:

- start with the metric derivation / component list for the actual metric and expand on it recursively until leaf metric nodes are reached
- check which metrics in the constructed tree are already measured and which need to be computed
- compute the needed metric values in a bottom-up manner according to the SPARQL-based transformation approach
- return back the whole, enriched with measurements tree

The issue of lack of statistics operations in SPARQL has not been resolved yet. It can involve two directions: (a) exploit the underlying capabilities of the RDF Triple Store to natively construct the operations in case these can be utilised in SPARQL queries; (b) extend SPARQL operation capabilities via exploiting a vendor-independent technology. The first direction creates a strong dependency on the underlying Triple Store, a kind of lock-in which cannot be easily overcome when this Triple Store is to be substituted (scenarios of component exchange in the CloudSocket platform). The second direction is independent of the Triple Store exploited. However, it needs to be evaluated in terms of query evaluation time as it might lead to additional overhead with respect to native

realisation of the same statistics operation. One current realisation technology for this second direction is Jena ARQ[11] which allows the specification of custom functions and custom predicates.

The main component responsible for the delivery of the KPI analysis functionality is the *Conceptual Analytics Module*. Its internal architecture focusing on this kind of analysis can be seen in Figure 7. It comprises the following sub-components:

- the *Conceptual Analytics Engine* which is the main component realising the KPI analysis functionality.
- a REST service, named as *Conceptual Analytics Service*, which is responsible for encapsulating the analysis functionality, offered by the *Conceptual Analytics Engine*, in the form of a REST API to the *Hybrid Business Dashboard*. This service includes methods which enable: (a) to obtain measurements for KPI metrics based on the history period and the respective client, if needed; (b) to provide a drill-down of a specific KPI metric; (c) to pose SPARQL queries which can enable exploration of the possible metric formula space; (d) to retrieve the customers for which measurements over a certain KPI exist (i.e., they have purchased the corresponding BPaaS and they have started executing it). As already indicated, the third functionality is more suitable for experts in LD technology. To this end, we plan to modify it via providing a respective abstraction (e.g., metric formula definition) via which the user can express the desired formula which will then be transformed into a SPARQL query. This abstraction is currently an OWL-Q specification of the metric which requires from the expert to have good knowledge of OWL-Q. On the other hand, the last functionality is suitable in case that a broker needs to inspect the performance of a BPaaS in terms of certain clients, mapping to a sort of client-based reflection. For instance, there can be cases where for one client, all KPIs are over-satisfied while for other clients, some KPIs are violated. Thus, the broker would then have the opportunity to inspect those client-specific KPIs and perform a drill-down on them to inspect the actual root causes for the respective situations.
- the *Metric Specification Extractor* which is responsible for obtaining from the name of the KPI metric, given as input to the module, its respective (OWL-Q) specification.
- the *Metric Formula Extender* which expands the formula of the KPI metric (see first transformation step above).
- *SPARQL Transformer* which transforms the expanded formula along with the other information given as input (evaluation period, customer, history period) into a SPARQL query.

Please note that the actual issuing of the SPARQL query is performed by the *Semantic KB Service* (see also general architecture of the BPaaS Evaluation Environment in Section 3.2).

To summarise, a KPI analysis functionality has been delivered which has many benefits, especially over the state-of-the-art. These benefits span: (a) the ability to configure dynamically various aspects of the metric specification, such as formulas and evaluation periods, enabling appropriate exploration of the possible metric definition space, especially for domain-specific metrics; (b) the ability to calculate both customer-based and broker-based metrics; (c) the ability to drill-down over respective metric derivation trees; (d) the ability to exploit the rich information already specified and stored in the *Semantic KB*. In the future, this latter advantage will be enhanced with the capability to exploit external information as well (actual benefit derived from LD technology plus the respective modelling capabilities of the OWL-Q KPI extension), which can be drawn from external information sources.

[11] https://jena.apache.org/documentation/query/arq-query-eval.html

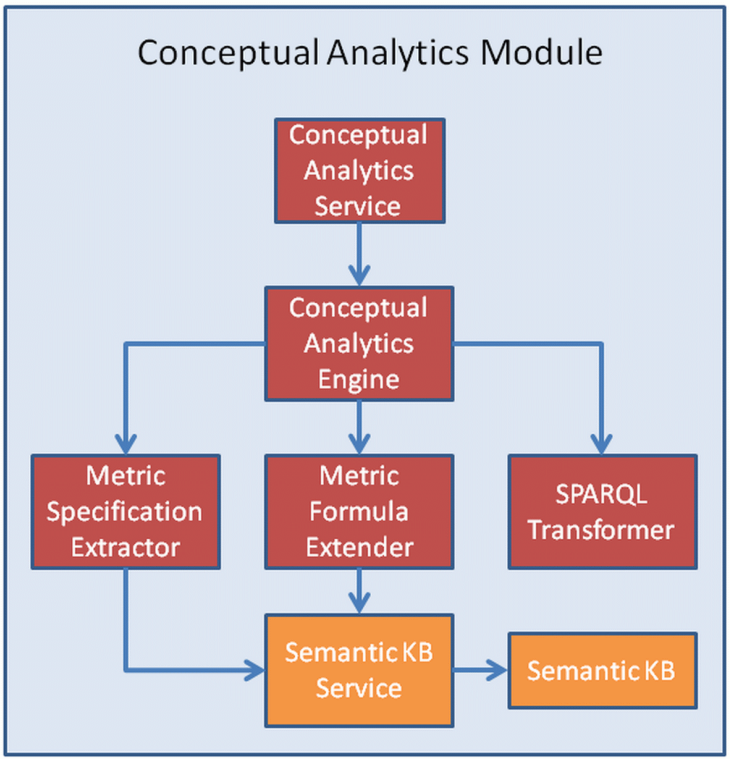

Figure 7 - The internal architecture of the Conceptual Analytics Module focusing on KPI Analysis

3.5 Best Deployment

We envision to exploit our previous work in best deployment reasoning / discovery [38] in order to make it semantics-aware and thus support a better accuracy level. This work has been conducted in the context of the PaaSage project. The following two subsections first summarise our previous work and then sketch the main idea of how to extend it, respectively.

3.5.1 Best Deployment Reasoning via a Knowledge Base based on CAMEL

Our previous work [38] relies on the CAMEL DSL [53] which is able to cover the modelling of both deployment information as well as the execution history of applications by also being able to correlate both of these two aspects. To remind the reader (see more details in D3.3 [19]), a CAMEL deployment model explicates which application component is hosted by which IaaS service / VM offering on both the type and instance level. On the other hand, execution information, which is correlated to a deployment and a requirement model explicating which requirements have led to this deployment and to the generation of this execution information, indicates which measurements have been produced, which scaling actions have been performed and which SLOs have been violated. From this view-point, CAMEL provides all the needed information to perform the reasoning. However, the main drawback is that of course this information is not annotated with ontology concepts and thus the respective semantics is not totally captured.

All the deployment and execution models of an application are stored in a model repository upon which a KnowledgeBase (KB) (realised via the Drools[12] open-source component) operates in order to derive additional, added-value knowledge. The derivation of this knowledge is guided via the execution of a certain set of rules. The knowledge that can be produced takes the form of the best deployments for applications and their components as

[12] https://www.drools.org/

well as similarity matches between applications and between application components. This knowledge is structured according to a certain domain model which follows a lightweight integration approach with the model repository by providing pointers to models from which additional information can be drawn for a certain element, such as an application component.

The main logic employed in the similarity rules signifies that component matching is inferred if the component names are equal, while application matching is inferred when applications have a great percentage of similar components which is beyond a certain threshold. This is an initial approach to application / component matching which is rather naive. On the other hand, such information can enable to discover best deployments even for applications not having execution histories already recorded. This can be quite useful in case of new BPaaSes for which the allocation needs to take place. If we know that a new BPaaS resembles an existing one, then it might be easier to discover some promising best deployments for it. Furthermore, the derivation of best deploymens for BPaaS components / parts can also be beneficial, even when there is no ability to derive relationships between whole BPaaSes, as it can lead to fixing parts of the actual deployment optimisation problem to solve, thus saving precious deployment reasoning time. Such a capability has been currently realised within a specific deployment reasoning component of the PaaSage platform called the CP Solver [38], where either a set of best application deployments are retrieved from the KB and then ranked or some deployments for particular application components are fixed.

The architecture for this previous work is depicted in Figure 8. As it can be seen, there are four main components: (a) the *Model Repository* in which all the execution and deployment information is stored; (b) the *CDO Server* which is the control access point over the *Model Repository;* (c) the *KB* which retrieves the information from the *Model Repository* upon firing of the respective rules; (d) any respective component (e.g., the *CP Solver*) which initiates the firing of rules and obtains back via queries the facts produced in order to exploit them (e.g., for deployment reasoning). The architecture includes a supporting component called *CDO Client* which is able to establish secure sessions over the *CDO Server* with the aim to support the management and especially the retrieval of the models stored.

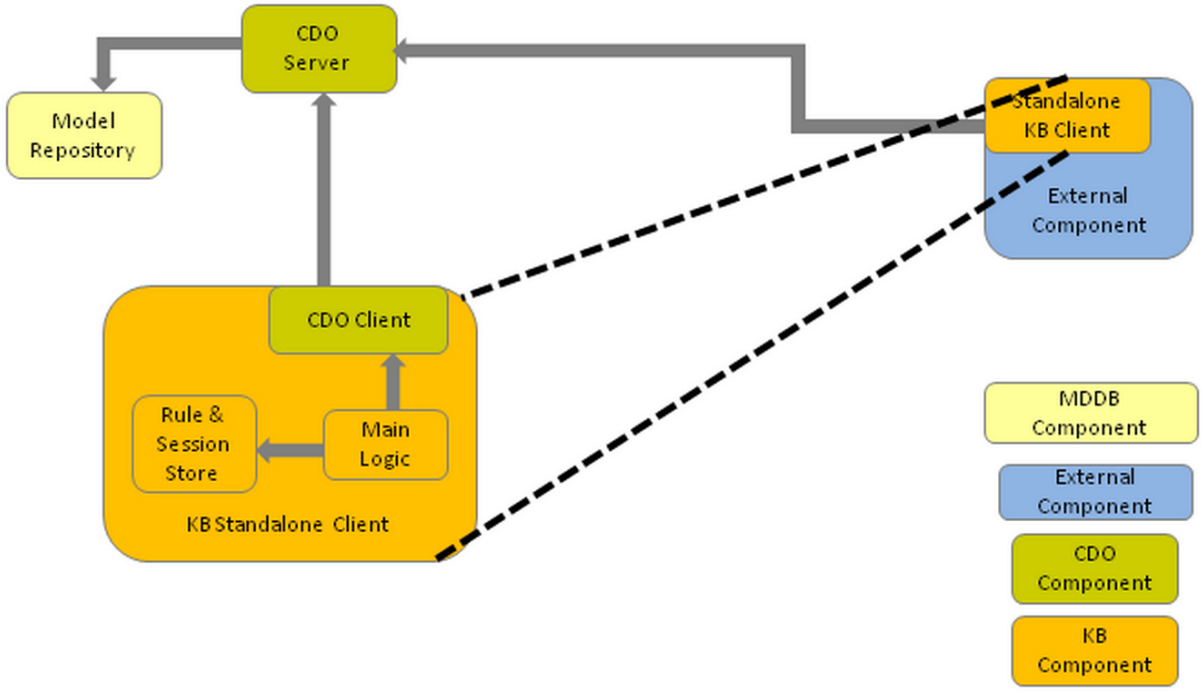

Figure 8 - The previous architecture of the KB system

3.5.2 Best Deployment Reasoning via the Semantic Knowledge Base

The main drawback of the previous work [38] in terms of application / component matching needs to be resolved via the use of semantics. As such, to facilitate this resolution, the *Model Repository* now takes the form of a *Semantic KB* in which all appropriate information has been semantically described and linked. In this way, our previous work needs to be actually extended with the capability to operate over such a *Semantic KB*. This extension should also consider that fact that the respective rules need to be modified to reflect the Evaluation & OWL-Q ontology contents as well as the new way of information accessing. We advocate that the existing technology realisation could still be exploited due to its capability to express composite rules as well as integrate external programming logic (e.g., in the form of Java code). An alternative would be the use of semantic rules which are however limited if we consider the possible practical realisations (reasoning frameworks) that rely on more limited forms of semantic rule representation.

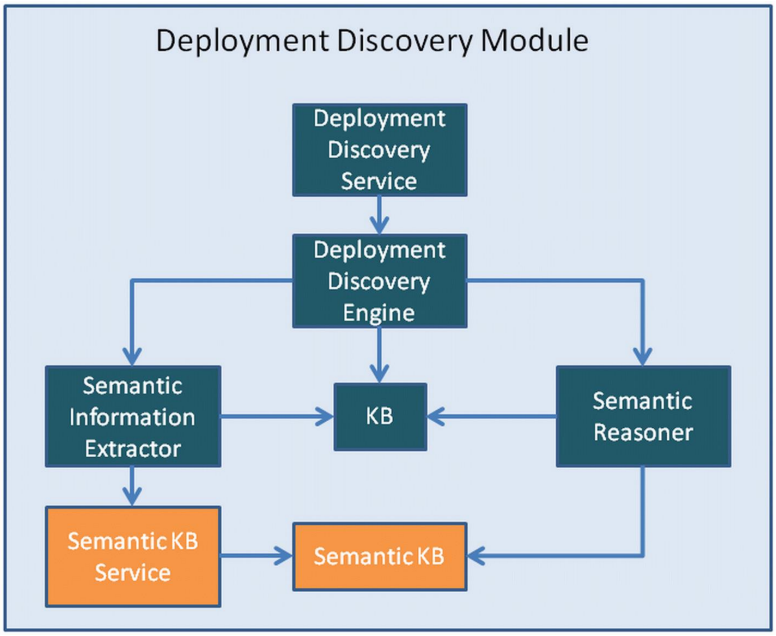

Figure 9 - The internal architecture of the Deployment Discovery Module

Based on the above rationale, the internal architecture of the Deployment Discovery Module blueprint has been designed and is now depicted in Figure 9. This architecture is an update of the previously presented one (see Figure 8) which now includes the following modifications: (a) replacement of *Model Repository* with the *Semantic KB*; (b) incorporation of the *Deployment Discovery Engine* as the main component responsible for realising the main deployment discovery functionality. This component first orchestrates the execution of other components ((d) and (e)) to guarantee the suitable population of the KB with appropriate knowledge facts and then calls the KB in order to produce the best BPaaS deployment facts and retrieve them; (c) use of a REST service, named as *Deployment Discovery Service*, on top of the *Deployment Discovery Engine* to deliver the best deployment discovery functionality; (d) incorporation of the *Semantic Information Extractor* which takes care of exploiting the *Semantic KB Service* to extract only the most relevant information and map it to respective facts into the *KB*; (e) use of *Semantic Reasoner* to infer semantic equivalence relationships between different BPaaS artifacts by operating directly over the *Semantic KB*. These facts are entered into the KB to enable discovering best deployments even for BPaaSes which do not yet possess an execution history.

Thus, as it can be seen, the existing *KB* is not replaced but just altered in order to incorporate the changes of the rules to be applied. Moreover, we introduce four new components to orchestrate the population of the *KB* and the firing of the respective rules as well as deliver the respective results via a REST service. Furthermore, the

introduction of the *Semantic Reasoner* enables a better and more accurate way of matching BPaaS artifacts as is explained in the next paragraph.

By exploiting now the semantic annotations over BPaaS workflows, tasks as well as cloud services, we will now have a better notion and correspondence to real / ideal matching as in this case two workflows will be identical if they map to the same semantic task and the same semantic I/O. The same holds for tasks and services. As such, these matching facts could be extracted and incorporated into the *KB* in order to drive the respective reasoning based on rules. Such an extraction involves the *Semantic Reasoner* which can rely on our previous service discovery work (see D3.3 [19]) possibly extended in order to operate over a *Semantic KB* instead of semantic descriptions of the corresponding service artifacts within the file system or offered as web resources to be downlaoded.

This initial idea sketched also requires slightly altering the domain model in the *KB* based on the following rationale: (a) now not model repository pointers are needed but just URIs to RDF resources; (b) the best deployments should be found not for cloud applications and respective components but for BPaaS workflows and corresponding tasks - this enables dealing with higher levels of abstraction (from SaaS to BPaaS); (c) now we have the case where not only IaaS but also other types of cloud services are being involved (PaaS & SaaS).

The above sketch of the blueprint has not been realised yet but the respective implemenation effort will start soon. It is actually convenient to rely on an existing solution and technology as this will guarantee that the least possible development time is enforced.

3.6 Event Pattern Detection

The goal of this blueprint is to discover a particular series of events that map to the occurrence of a specific SLO or KPI violation. This goal has been already realised in our previous work [70]; however, this realisation needs to be slightly extended. In that work, a logic-based event pattern detection technique is followed where a respective pattern is mapped to a corresponding violation event only if it is logically implied. This logical implication forms a better foundation for pattern detection with respect to other approaches which rely on threshold-based measures as main criteria for selecting a particular pattern or not, especially as such thresholds cannot be always computed and might depend on the respective domain.

Apart from detecting event patterns, our previous work is also able to produce respective adaptation strategies out of them which attempt to address accordingly the current problematic situation. These strategies are derived by following a semi-automatic approach where as a baseline considers the existence of manually provided mappings of single events to single adaptation actions. As such, our previous work enabled the composition of those adaptation actions by following an event-based approach in order to handle even sophisticated problematic situations that cross more than one layer of abstraction.

Different strategies for action composition have been proposed depending on the way the events are correlated to each other and the effects that each action can have on the service-based system. For instance, if two events concern a different layer and their actions are not conflicting, then they could be mapped to a composition of respective adaptation actions where the action in the lower layer is executed first followed by the action on the higher layer. In any case, an event pattern can be mapped to a multitude of action compositions, as one event can be mapped to multiple actions that can be used to resolve it. Our previous work relied on a preference-based approach to rank the composition alternatives and thus be able to select the most appropriate one. Via this selection, then a complete adaptation rule can be derived. The preference-based approach was quite simple. Each single event to single action mapping is assigned to a respective utility, where the higher is the utility, the higher is the appropriateness and thus preference over the single action. Then, a mapping of an event pattern to

a set of composition actions will be ranked via the multiplication of the utilities of the individual single event-to-action mappings from the event pattern and composition set.

In order to assist in the comprehension of our previous work, imagine the case that there are two events E1 and E2 which are mapped to two actions each, namely A11 & A12 and A21 & A22, respectively. Rule E1->A11 has a utility of 0.5 while Rule E1->A12 has a utility of 0.7. Rule E2->A21 has a utility of 0.3 and Rule E2->A22 has a utility of 0.6. Consider now that the E1 && E2 event pattern has been detected to lead to a KPI violation. As E1 and E2 concern different layers and their respective actions are not conflicting, a series of 4 composition alternatives can be produced with the following rankings: E1 E2->A11 A21: 0.15, E1 E2->A11 A22: 0.3, E1 E2->A12 A21: 0.21, E1 E2->A12 A22: 0.42. As such, in the end, the principal main mapping which is advocated to be adopted is the last one: E1 E2->A12 A22 which will correspond to a corresponding adaptation rule being proposed by the system.

The evaluation of our work has led to the conclusion that the accuracy of the event pattern detection is satisfactory. However, there is plenty of improvement potential on the way the respective adaptation actions can be selected. Our main idea of how to evolve this selection to make it more sophisticated and suitable is to associate each composition alternative to particular non-functional parameters from which then the respective score can be produced. The score / utility function should also consider the execution history of each adaptation strategy (i.e., adaption action composition) and actually employ a learning approach. The non-functional parameters to be considered could include: (a) the cost of the adaptation (strategy / action) (as, e.g., replacing one service with another results in certain cost); (b) the adaptation time, i.e., the expected time the adaptation will take to finish; (c) degree of resolution, i.e., the degree via which we expect that the problem will be resolved. The latter parameter could actually map to the existing preference-based ranking given as it indicates the respective belief of the expert that this action will actually solve the current problematic situation. It also maps to the respective uncertainty involved in the corresponding adaptation strategy selection as such a belief may not be correct or might reflect only particular cases. As such, while (a) and (b) can be objectively measured, (c) is surely a subjective measure which is the actual point of adjustment via the consideration of the execution history of adaptation strategies / rules. This means that there should be a specific way of modifying the value of the preference depending on the respective past outcome of the corresponding adaptation action or set of actions selected for the respective adaptation rule. A simple approach on this would be to introduce a fixed penalty and reward whenever the action does not or does resolve the current problematic situation. Another approach would be to introduce a more sophisticated formula via which the respective measure calculation can be performed. The overall score / utility of an adaptation strategy can be computed through a weighted sum over the respective utilities for each parameter according to the Simple Additive Weighting technique [34] where the weights can be derived by following the Analytic Hierarchy Process (AHP) [55]. Standard linear utility functions could be employed for the first two parameters which rely on a respective range over which the corresponding expected value can belong along with the direction of values that should provide a higher utility to those values that follow this direction. In particular, the following utility funtions could be used in the computation of cost and adaptation time.

$$uf_q(x) = \begin{cases} \dfrac{v_q^{max} - x}{v_q^{max} - v_q^{min}}, v_q^{min} \leq x \leq v_q^{max} \;\&\; x \downarrow \\ \dfrac{x - v_q^{min}}{v_q^{max} - v_q^{min}}, v_q^{min} \leq x \leq v_q^{max} \;\&\; x \uparrow \\ 0, \text{otherwise} \end{cases}$$

where uf_q is the utility function for non-functional parameter q, the arrows denote the direction of values (negatively and positively monotonic, respectively) and v_q^{min} and v_q^{max} denote the lower and upper bounds

improsed for the parameter. Of course, for the two non-functional parameters of adaptation time and cost, the first and third parts of the utility function apply as these parameters are negatively monotonic.

Whatever is the actual selection of the measurement formula, we face now two situations: (a) the overall utility of an adaptation strategy is not stable; (b) adaptation rules will have to be modified in the actual execution system on demand, whenever it is detected that a particular strategy has been surpassed by another one in the overall score for the particular event pattern at hand. As such, there is a need for another component in the system, at the edges of the BPaaS Execution and Evaluation Environments, which has to perform this updating. Moreover, there is a need to either constantly modify the respective bundle or decide not to reflect this modification to the bundle as this is not of a major concern for the BPaaS Customer.

Apart from the existence of the above component, there is also the need to introduce a respective method on how to compute the values for the first two parameters of adaptation cost and time. While this might be apparent in the case of cost, it is not apparent in the case of time. This relates to somehow deriving the expected time that a particular adaptation capability / action needs to be executed which can depend on the underlying infrastructure and its corresponding capabilities and characteristics. As such, we might need to have a kind of profiling for different adaptation actions which along with the respective adaptation strategy structure can enable us to derive the overall execution time for the whole adaptation strategy.

The internal architecture of the *Pattern Detection Module* responsible for delivering this kind of analysis is depicted in Figure 10. As it can be seen, the architecture comprises 5 components: (a) the *Information Extractor* is responsible for retrieving from the *Semantic KB* only the information that is relevant for the event pattern detection; (b) the *Pattern Detection Engine* operates over this information in order to derive the respective event pattern for the specific KPI / SLO violation at hand. It is also responsible for invoking the *Rule Derivator* component, when requested, in order to produce the complete adaptation rule to be delivered; (c) the *Pattern Detection Service* is a REST service responsible for encapsulating and delivering the main functionality of the *Pattern Detection Engine*; (d) the *Rule Derivator* takes as input the event pattern derived by the *Pattern Detection Engine* and attempts to produce the respective adaptation rule by considering the current content of the (Adaptation) *Rule Base* and the current content of the *Semantic KB* (spanning information about which adaptation actions have been performed, whether they have been successful, and what is their costs and expected execution time); (e) the *Rule Base* is the actual storage medium for the adaptation rules already in place. It also stores the alternative strategies per rule in order to enable the modification of each rule, whenever such a need arises. It should be noted that all parts of this architecture are already in place but some functionality or information is missing spanning: (i) the *Information Extractor* - the current implementation does not operate over the *Semantic KB* and relies on a different meta-model / language, (ii) the update of functionality of the *Rule Derivator* and (iii) information about adaptation actions and their history of execution (which is currently partially modelled and not delivered by the CloudSocket prototype).

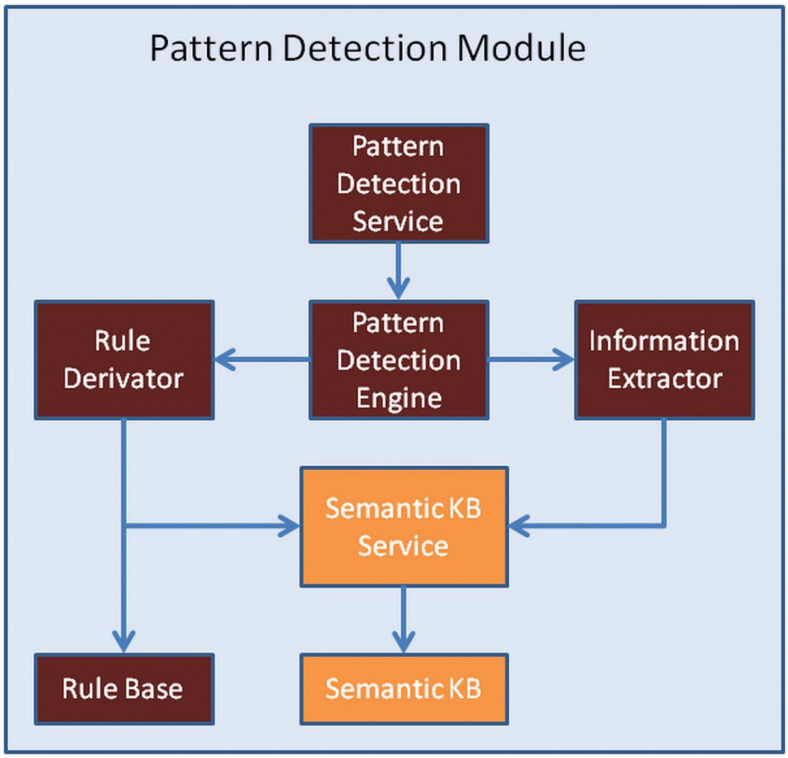

Figure 10 - The internal architecture of the Pattern Detection Module

3.7 Process Mining

Process mining is a field in Business Process Management which deals with the extraction and processining (so called mining) of process execution information with the ultimate goal of producing added-value business intelligence knowledge. Such business intelligence knowledge can take different forms where each process mining algorithm usually focuses on just one form and not multiple ones. These forms span the following:

- mining of the workflow schema
- mining of decision rules
- mining organisational / social network information
- performance mining
- service mining
- workflow recovery mining

As already stressed in the review of the state-of-the-art in process mining (see Section 3.1.4), there is also a special kind of mining which can produce results with better accuracy. This kind of mining is called *semantic process mining* and exploits the annotations provided in the execution log usually drawn from one or more certain (domain-specific) ontologies (e.g., functional / task ontology and object ontology). As the main goal is always to have the best possible level of accuracy, we certainly focus on supporting any kind of semantic mining, especially as particular types of annotation are already in place in specific parts (information aspects) of a BPaaS bundle.

From the different forms of process mining, we focus mainly on functional and organisational mining (with some correlation to the non-functional perspective for the second mining type) for the following main reasons: (a) non-functional analysis is already supported by the previously analysed blueprints such that there is a need to support only complementary forms of analysis / mining; (b) for certain kind of information (e.g., workflow recovery) there

are no logs being generated currently in the CloudSocket prototype, so the respective kinds of mining cannot be supported (e.g., workflow recovery mining); (c) service mining is similar to other forms of process mining but in our case the services realising a BPaaS workflow are already known. As such, there is no need to cover this kind of mining. To this end, we end up focusing only on the first three kinds of mining in the previously supplied list. Our main goal is to support all of them by also exploiting semantic annotations whenever they are available.

From a research perspective, it will be of course more interesting to devise new algorithms rather than exploit existing ones. However, the proposal of new algorithms should rely on: (a) whether existing algorithms have major drawbacks which have to be resolved; (b) producinng some kinds of process knowledge facts that are not currently supported by the state-of-the-art process mining algorithms while they can be definitely interesting from the CloudSocket broker perspective. Concerning (a), the state-of-the-art has shown that there are certain drawbacks which sometimes are due to the lack of semantics. To this end, we have initially taken the approach of just extending these algorithms to make them semantics-aware. In case that still the respective accuracy of the algorithms extended is problematic, then we will pursuit the derivation of new ones or the further extension of existing algorithms. The algorithms to be selected focus mainly on workflow schema and decision mining.

Concerning (b), the new forms of business intelligence knowledge which might be interesting from the broker perspective take the following two forms: (i) derivation of best and worst (human) resource usages per BPaaS; (ii) derivation of (partial) organisation model patterns per BPaaS. The first form actually resembes best deployment analysis where deployment means mapping of user tasks to respective human resources. This form is of course related to the non-functional aspect as it attempts to discover resource usage patterns against BPaaS instances from which to derive which usage patterns are the best (best satisfied all KPIs posed) or the worst (lead to violation of one or more KPIs). The second form attempts to find respective groupings of resources into roles as well as the corresponding correlations between these groups. This information can be valuable in case that the CloudSocket broker desires to provide consulting services focusing on how an organisation could be structured, depending also on its type, in order to support a certain BPaaS or to modify it to reflect such organisation structuring. This second form will only make sense in case that the respective BPaaS workflow is not actually resource-annotated, i.e., user tasks have not been mapped to certain roles or users, or that all user tasks map to a certain generic role (e.g., of a worker).

The derivation of best and worse resource usages is a business intelligence mining form that has not been realised yet. This will be the subject of the next research deliverable (D3.6) on BPaaS evaluation. The second mining form can rely on existing organisational mining algorithms, which will be extended to raise their derivations from the BPaaS instance to the BPaaS level. This form will also be subject of the D3.6 deliverable.

Implementation-wise, in this deliverable and corresponding line of work, we have focused on how to extend existing mining algorithms mainly on workflow schema derivation. Decision mining algorithms will also be extended only in case a respective issue with the Workflow Engine log extraction is resolved which does not allow us to completely capture in the right order the respective (intermediate and final) outputs that are produced for a BPaaS workflow. The extension on workflow schema mining algorithms relies on the following step-wise rationale:

- the actual semantic information related to the execution of tasks of a BPaaS workflow is extracted from the *Semantic KB*. This information takes the form of functional annotations of tasks which complements the respective information required from an execution log (e.g., when the task was initiated or finished, by which user (if applicable), etc).
- this semantic information is transformed into the respective format expected from the algorithm
- the process mining algorithm is executed based on the transformed input

Currently, we are in the process of realising the first two steps. For the second step realisation, we will rely on supporting the log format of XES[13], a XML-based standard for log representation adopted by the IEEE Task Force on Process Mining[14] and accepted by IEEE SA Standards Board[15]. This latter decision has relied also on the fact that the ProM framework[16] has been selected to be exploited which makes available a plethora of process mining algorithms of different forms. As such, via this selection, the realisation of the third step is guaranteed. Also the possible exploitation of existing decision mining algorithms, as realised in ProM, is guaranteed provided that the aforementioned issue is resolved. The first algorithm on which the respective above process will be tested is the well-known Heuristics Miner Algorithm (see Section 3.1.4).

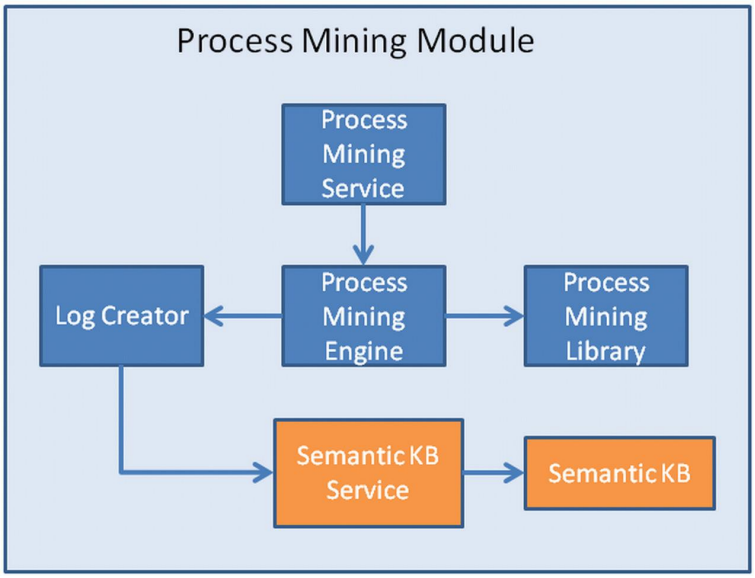

Figure 11 - The internal architecture of the Process Mining Module

The simplified architecture that is currently enforced is visualised in Figure 11.The main mining functionality is realised by the Process Mining Module. Internally, this module comprises four main components: (a) the Process *Mining Engine* responsible for orchestrating appropriately the components needed to deliver the process mining functionality. In particular, this component first guarantees that the appropriate process log is created and then invokes the process mining algorithm requested; (b) the *Process Mining Service* encapsulates the functionality of the Process Mining Engine and delivers it; (c) the Process Mining Library is a library of process mining algorithms, based on ProM, which can be invoked by the *Process Mining Engine*; (d) the *Log Creator* is responsible for creating the process log, when instructed to do so by the *Process Mining Engine*, from the *Semantic KB*.

3.8 Future Work

This section unveils some interesting research directions that could be followed in the near future for further enhancing or optimising the main functionality delivered by the BPaaS Evaluation Environment blueprints. The presentation of these directions is structured in respective sub-sections, each mapping to a certain blueprint.

[13] http://www.xes-standard.org/
[14] http://www.win.tue.nl/ieeetfpm/doku.php?id=start
[15] https://standards.ieee.org/
[16] http://www.processmining.org/prom/start

3.8.1 Information Harvesting

3.8.1.1 Unresolved Issues Addressing

Currently, most of the information covered by the Evaluation and OWL-Q ontologies can be captured by the Harvester module. However, the coverage of I/O parameters for tasks needs appropriate improvement and extension. On one hand, this is related to the way BPMN workflows are modelled, which makes it hard sometimes to identify what is the input or output of a specific task. Imagine for instance a user task. It might involve some parameters, sometimes in a Workflow Engine-specific way, for which we are not certain why they are used as input or output or both. This becomes even harder in the case of script tasks where the actual code of the script might have to be inspected in order to check whether there is a variable that is being read or written. In this sense, we have decided that the I/O of script tasks is not currently covered, while for user tasks we employ a particular heuristic in order to derive its I/O. In case that the respective variable is never used in the previous part (in the execution order) of the workflow, then the variable can be considered as output. Otherwise, the respective variable, as it has been produced by a previous task, should be considered as input to the user task. Of course, this heuristic does not cover well the case that a variable can be not only read but also written by a user task. In particular, if a variable is certainly read by a user task, we do not know, in case this variable is further exploited in the rest of the corresponding workflow, whether it is also written by the user task. This constitutes its current limitation which might be resolved in the future by entering specific vendor-specific (with respect to the Workflow Engine) extensions to, e.g., highlight explicitly whether a variable is input, output or both. Such extensions unfortunately can only be vendor-specific at the current moment, so in case that a vendor is modified, then they will have to be reinforced for the new vendor.

On the other hand, this is technically related to the current capabilities of the REST service of the respective Workflow Engine exploited. Such capabilities does not enable us to know at which time points each variable has been assigned a specific value. Maybe this limitation could be overcome in the near future but for now it is actually a stop-point for performing decision mining unless we could restrain this kind of mining on specific types of workflows.

3.8.1.2 Semantic Annotation

Semantic annotations are important especially for two types of analysis: (a) best deployment analysis where we need to know which BPaaS workflows or other components like tasks are equivalent; (b) process mining in order to produce mining results with higher accuracy. Currently, as already indicated, the direction followed is to rely on existing semantic annotations that have been incorporated in certain parts of the BPaaS bundle by the BPaaS Design and Allocation Environments. As we can imagine that, e.g., due to shortage of expertise or time, such annotations will not always be provided. Thus, it would then be highly required to infer these annotations by exploiting respective automatic annotation techniques and of course exploiting any kind of relevant ontology. In the context of service annotation, for example, we have seen some techniques from machine learning [32] which could be exploited. The main issue is that in our context, a BPaaS maps not only to services but many other kinds of information including: workflows, tasks, and metrics. If we consider that workflows and tasks can be considered as services, then respective service annotation work can be re-used and possibly expanded. If we consider, however, metrics, there is a shortage of approaches able to provide automatic annotations on such kind of non-functional terms. However, our previous work might be exploited, dedicated to non-functional (service) specification alignment [37], provided that: (a) there is a respective taxonomy of metrics which can be used to perform the annotation; (b) there is a specific way to map the current representation formalism of the metrics to be annotated to the formalism expected by the alignment algorithm.

Based on the above analysis, we expect that in short term we could inspect and re-use the (functional) service annotation algorithms. In the long term, non-functional term annotation could be a quite relevant and interesting in terms of research direction which could be followed.

3.8.2 KPI Analysis

3.8.2.1 Simplified Metric Exploration

The exploration of the metric formula space involves first changing the modelling of the OWL-Q metric before issuing the respective REST method call to obtain the respective measurements for that metric. Instead of playing with OWL-Q directly, which pre-supposes the existence of any kind of OWL-Q editor plus the actual knowledge of OWL-Q, it might be easier for the user to exploit a custom mathematical expression language (with mappings on OWL-Q) which could simplify and possibly accelerate the modification of the metric formula definition. Once the user is confident about the metric formula finalisation, then he/she can perform the modification in the respective OWL-Q specification or transformation code could be written from the custom formula expression to OWL-Q. This could be a short-term interesting direction which might be facilitated if a respective domain code API exists for OWL-Q.

3.8.2.2 Exploitation of External Sources

Apart from realising missing SPARQL operators, there is also a need to exploit external information sources, in case that the content in the *Semantic KB* is not enough (for the purposes of the current evaluator). To enable the incorporation of the respective information into metric definitions and then on respective SPARQL queries, there is a need for appropriately handling the retrieval of this information. This handling can come come with two alternative directions: (a) information is precomputed before the SPARQL query is issued; (b) some kind of technology is involved which enables the realisation of custom functionality in SPARQL that can enable the retrieval of such information. For (b), if ARQ technology is exploited in order to express the missing SPARQL operators, then this technology would actually also be the realisation medium for the respective handling as the respective hooks are offered via this technology.

3.8.2.3 OWL-Q to SPARQL Transformation Algorithm Expansion

The transformation algorithm is quite simplified by considering the discipline of keeping things as simple as possible. However, this algorithm should be appropriately validated in order to inspect whether it is sufficient enough to cover all possible cases. Otherwise, it might have to be extended in order to cover these cases. In the context of the CloudSocket project, there are enough use cases to assist in this validation, provided that they are mapped to suitable KPI metrics that need to be computed and that respective (deployment, measurement, etc.) information can be derived and semantically enhanced for them. Such validation can also enable evaluating the overall KPI analysis approach towards receiving interesting optimisation feedback.

3.8.3 Best Deployment Discovery

3.8.3.1 Human Resource-Based Best Deployment Discovery

As an alternative to organisation mining (see Sections 3.7 and 3.8.4), the human resource aspect could be considered in order to find best possible allocations for user tasks. This of course pre-supposes that the organisations exploiting the respective BPaaS instances are big enough to have the ability to incorporate different employees in the execution of the respective workflow instances which can increase the probability of existence of different allocations for the same user tasks and for the same BPaaS purchased. This also pre-supposes the existence of an appropriate monitoring framework which is able to sense appropriate non-functional metrics that measure the quality of service of the human resources. For some widely-known domain-independent metrics, this support can be easily or already realised but this is certainly not the case for domain dependent ones. If both of these assumptions hold, then the extension of the current blueprint will be quite straightfoward and easy as only particular artifacts will have to be modified slightly (i.e., the domain model plus the rule set in the KB). The respective facts derived could be used as indicators for the need of human resource management optimisation which could be beneficial in two cases: (a) the user organisation is exploiting the BPaaS Evaluation Environment

in order to produce this added-value knowledge (case of BPaaS as a consulting service - user is given BPaaS workflow and might be guided in the further use of the CloudSocket platform) or (b) the CloudSocket broker acts on behalf of the user (case of BPaaS as a product which includes the derivation of this type of knowledge) in order to derive this information and build on top of that the respective consulting services. In the second case, there is of course the issue of privacy as the broker should be entitled to just derive the respective information without exploiting it for its own internal purposes.

3.8.3.2 Single KB Approach for Best Deployment Reasoning

What has been indicated in Section 3.5.2 bares in mind the main drawback of current powerful reasoning frameworks: they are not robust and scalable enough, while when they are, they do not employ quite expressive rule languages and formalisms. However, a long research quest would be to produce such a sophisticated reasoning framework on which then very powerful types of reasoning would be built, such as the one about best deployments. As such, when such a framework becomes available, then it would be more easier to produce a respective best deployment reasoning architecture with less components involved, one KB, the semantic one, and a uniform representation of the rules, thus not requiring the need to perform any kind of extraction of information from another KB. The research partners of the CloudSocket consortium will continously observe the research results in this field of research and immediately grab the corresponding opportunity when it appears.

3.8.4 Process / Data Mining

3.8.4.1 Organisation Mining

As have been already indicated in Section 3.7, we plan to devise new organisational mining algorithms which will enable the production of added-value knowledge of two main forms that can assist the CloudSocket Broker either in providing novel consulting services or appropriately modifying the BPaaS offered (at the workflow level). In addition, upon resolution of the data instance extraction issue, we plan to exploit existing decision mining algorithms by also making them semantics-aware.

3.8.4.2 Taxonomy / Abstraction Mining

Another interesting form of knowledge that we might be interested in mining concerns the derivation of abstraction levels within processes. Such abstraction levels can enable to provide interesting overviews of BPs / workflows as well as might assist in the mapping between BPs and workflows by considering the fact that a certain BP activity could be realised by a certain BP fragment which could be abstracted on the activity level. Existing algorithms have already been proposed on this which will be explored, semantically-enhanced and possibly extended, if needed.

3.8.4.3 Mining Algorithm Composition

Sometimes, a chain of process mining algorithms might need to be executed in order to produce a series of business intelligence knowledge, where the output of one algorithm can be used as input for the next one. While it can be considered that this chain could be realised in the form of a sophisticated workflow, in practice we have seen that usually either parallel or sequential chains are exhibited. To this end, we are thinking of expanding the current research implementation towards enabling the lightweight composition of process mining algorithms. In the first place, we might no need to consider exploiting workflow technology to support the limited type of workflows needed. However, even in this case, we need to stress that there will be a need to visually specify the composition as this might not be easy to specify in a textual way. In addition, usually some configuration of the algorithms is needed which is better to be visually provided. This actually creates the need of expanding the *Hybrid Dashboard* interface to support this kind of visualisation. Apart from this, the composition component responsible for the orchestration of the process mining algorithms should be able to appropriately also check the respective preconditions of each algorithm to inspect whether they are validated or not as well as evaluate the

output produced by each algorithm which might be erroneous such that there will be no need to execute the next algorithm in order. This kind of validation and checking has to be appropriately explored and realised and constitutes another sub-direction of research mapping to the current overall direction.

3.8.4.4 Deployment Log Mining

A rule-based approach could be envisioned for enabling transactional and adaptive deployment of cloud-based applications and BPs. Such an approach is sketched as a new blueprint in D3.4 [20]. In such an approach, adaptation deployment rules drive the adaptation of the current BP deployment. However, such an approach is only able to accommodate known problematic circumstances which are mapped to a set of certain (mainly component-state-based) events. Thus, when an unanticipated situation occurs, the deployment system (e.g., *Cloud Provider Engine*) is not able to address it due to the lack of a respective rule. In this case, two main directions can be followed: (a) detection of events and automatic derivation of the respective actions to be performed according to the approach proposed by FORTH (see also Section 3.6); (b) mining of the deployment logs to discover in a more fine-grained way the respective events that have led to this situation and then exploitation of either the approach from FORTH for the automatic derivation of actions (when possible) or the deployment history for discovering actions which can be used to complete the specification of the respective rule. For this second direction, techniques from data mining could be suitable, especially those related to decision trees mining and rules mining. As this is a rather novel direction of research with significant research gaps, it would be interesting but also quite ambitious to address this challenge in the near future. The consortium will explore the respective possibilities and decide whether moving to this direction. The exploitation of course of already developed algorithms and techniques is rather obligatory in order to accelerate the research resolution process.

4 INTERACTION WITH OTHER ENVIRONMENTS

4.1 Required Input

The previous sections have analysed a particular set of blueprints which need to have appropriate input so as to function properly. Such input should come from other components or blueprints that have been developed in the context of workpackages WP3 and WP4. We should also highlight that the harvesting blueprint is the one which retrieves all the required input, links and forms it based on the BPaaS Evaluation modelling blueprints and then stores it in the *Semantic KB* in order for the core evaluation environment (analysis) blueprints to exploit it. We now explicate which WP3 & WP4 components are exploited and which information is collected from them. Then, we highlight which information is required by each blueprint of the Evaluation Environment.

Cloudiator. The *Cloud Provider Engine*, mapping to the Cloudiator blueprint in D3.3 [19] which has been already adopted in WP4, is exploited to collect deployment information in terms of which internal software components have been mapped to which cloud services (IaaS / PaaS). It will be also exploited in the future in order to retrieve logging information which can assist in the derivation of deployment rules (see Section 3.8.4.4). Such deployment rules will constitute a valuable information asset for improving the deployment of BPaaSes.

Visor. This is the current implementation of the *Monitoring Engine* in WP4. We foresee that in the future, the harvesting component could retrieve information from other *Monitoring Engine* implementations mapping possibly to other monitoring blueprints from D3.3 [19], especially in case they are adopted in the context of WP4. This will of course require adapting or extending the respective information collection mechanism that has been incorporated into the harvesting blueprint.

Workflow Engine. This is a WP4 component. It is contacted in order to retrieve information about deployed BPaaS workflows from which also the mappings of workflow tasks to services is extracted. In addition, this component also enables the retrieval of workflow execution information which maps to the workflow instances that have been created from deployed workflows and their respective task instances.

Registry. This is again a WP4 component which provides a respective registry for different types of information. Currently, this registry is exploited to obtain detailed information about the services (identifier, URL / URI, I/O parameters and their semantic annotations for SaaS services) that are being involved in BPaaS workflow tasks in order to enrich the corresponding mappings.

Marketplace. This is a WP4 component dedicated to offering a marketplace for BPaaS offerings while also incorporating the actual mechanism for user identification management. From the latter mechanism, we can retrieve information about CloudSocket brokers and their customers (including involved users and roles).

Meta-Model Repository. This component has been already realised in WP4. It is contacted to obtain BP and KPI modelling information. For the latter information type, we assume that it has already been specified in the OWL-Q KPI extension. If this is not the case, then transformation code will be implemented to transform the KPI models to OWL-Q ones before they are being fed into the *Semantic KB*.

As it has been indicated in Section 3.3, all this information has to be linked according also to the schema of the Evaluation and OWL-Q ontologies. To this end, we make the following assumptions which should hold so as to enable the appropriate linkage of all the information gathered:

- An abstract or concrete BPaaS workflow always refers to services that are involved in the (service) Registry such that we can obtain additional information about them;

- A deployed workflow should be linked to the actual BPaaS instance concerned. Please note that a deployed workflow maps to a BPaaS which has been purchased by the client; as such, it can be considered as a BPaaS instance of the BPaaS actually offered. Fortunately, the *Cloud Provider Engine* offers a specific REST API from which this correlation can be obtained.
- A BPaaS workflow should refer to valid users and roles. Users should be mapped to the *Marketplace* identification system from which we can obtain additional information about them. Of course, roles can be either generic or specific to a certain BPaaS workflow. In the former case, generic roles can be retrieved from the *Marketplace* identification system; for the latter case, the specific roles can be retrieved from the *Workflow Engine* itself.
- An internal service component (instance) should be mapped to a respective cloud (IaaS / PaaS) service (instance). Both types of mapping (type & instance level) should be drawn from the *Cloud Provider Engine*. Otherwise, the mapping at the type level can be derived from the BPaaS bundle.
- A measurement should be mapped to three pieces of information: (a) the id of the metric on which its computation has relied; (b) the id of the object (any kind of BPaaS (system) component) being measured; (c) the id of the BPaaS instance concerned. Such information is actually exploited for the evaluation of KPI metrics.

All the gathered and linked information suits different types of analysis. In the following, we explicate which information is required for each type of analysis:

- *KPI Analysis*: Requires to know which KPIs and respective metrics have been modelled for a BPaaS and how they connect to the respective BPaaS components and their deployment dependencies. It also requires to know which measurements are already in place for which KPI metrics. Such information, as already been explicated, is drawn from the BPaaS bundle as well as the monitoring and deployment log from Cloudiator / Visor.
- *Best Deployment Discovery*: Requires the same information as the KPI analysis. Deployment information is required in order to create the deployment exploration space while the mapping of this information to respective measurements can enable evaluating which deployment has achieved the best possible service level.
- *Event Pattern Detection*: Again the same information as in the previous analysis types is needed. The measurement information enables the generation of the events while the deployment information enables correlating only those events which should be related and considered in the analysis according to the BPaaS (dependency) hierarchy.
- *Process Mining*: Here we mainly focus on the functional and organisational aspect. As such, we are interested more on workflow and organisational information. Both types of information is actually drawn from the *Workflow Engine*, while some information about CloudSocket brokers and customers is drawn from the *Marketplace*.

4.2 Exploitable Output

As it has been explicated in the previous section, we foresee four main types of knowledge which can be exploited to optimise BPaaS design and deployment. In addition, we also envision that the whole *Semantic KB* could be another major output to be exploited either internally by CloudSocket components or externally from external components to the current CloudSocket prototype. In the following, we shortly analyse each type of knowledge that can be derived by explicating which components can benefit from it for which reasons.

KPI Analysis. This type of analysis enables to assess whether KPI targets are reached or not. In the latter case, via KPI drill-down we can go down till a specific component in the lower layers is found which is the one to be blamed for. As such, this type of analysis can be exploited to discover potential discrepancies in the quality / performance of a BPaaS. Such discrepancies can then outline possible optimisation alternatives for the

CloudSocket Broker which could be enforced in the BPaaS Design and Allocation Environments. Such alternatives could for instance indicate that a certain cloud service is underperforming such that it is replaced. They could also highlight that, e.g., a specific region in a BPaaS workflow is underperforming such that the respective tasks may have to be re-arranged.

Best Deployment Discovery. The discovery of best deployments in the context of certain KPIs enables to revise deployment decisions that have been undertaken within the BPaaS Allocation Environment by the CloudSocket Broker (or a respective external collaborator assigned to the BPaaS allocation task). The opposite direction might also be useful in some cases as it can be used as a guideline for certain deployments that should definitely be avoided by the CloudSocket Broker. We should stress here that the CloudSocket also benefits from obtaining best deployment suggestions even for new BPaaSes, provided that they are equivalent or similar to existing ones or at least have some components which are also exploited in existing BPaaSes.

Event Pattern Detection. This type of analysis maps to the discovery of a series of events which lead to the violation of a KPI or an SLO. Such information can be valuable mainly for adapting the runtime behaviour of the BPaaS as it can be exploited to produce respective adaptation rules which attempt to resolve such problematic situations accordingly during BPaaS execution. Such resolution can enable actually to pro-actively address this situation even before it actually occurs, thus supporting pro-active BPaaS adaptation. The respective event pattern detection insights can be either represented in the form of a decision table or via specific rules specified in SRL / CAMEL [76] (extension highlighted in [D3.4]), in case they are automatically completed in the last part of the analysis (depending also on the CloudSocket Broker preferences). Such an output can then be exploited mainly by the BPaaS Allocation Environment to update the respective BPaaS bundle, either by extending the information from the decision tables and transforming it into SRL adaptation rules or modifying the adaptation rules suggested, depending on the respective expertise of the CloudSocket broker (or his/her external collaborator).

Apart from deriving adaptation rules, this type of analysis might also be useful to show some potential discrepancies (e.g., cloud service underperformance) which always lead to certain KPI or SLO violations. Then, in case such discrepancies occur quite frequently, this can be an indication for appropriately modifying the BPaaS in order to permanently resolve them.

Process Mining. Different forms of process mining are envisioned to be supported which lead to the production of different types of knowledge. Here we will concentrate mainly on: (a) the re-creation of workflows which can unveil whether particular paths in the original BPaaS workflow are never followed or that discrepancies withing existing paths have been found (e.g., tasks that have been skipped or change of task ordering); (b) the discovery of decisions which can enable the replacement of manual tasks with automated decision ones; (c) the derivation of best / worst resource usages; (d) the discovery of organisational (model) patterns. These four types of knowledge are mainly exploitable by the BPaaS Design Environment which can actually modify the BPaaS workflow to reflect them.

Finally, we should highlight that another form of exploitable output which might be produced by the BPaaS Evaluation Environment maps to the deployment adaptation rules which will be derived via mining deployment logs retrieved from the Cloudiator / Cloud Provider Engine (see Section 3.8.4.4). Such information will be valuable during the deployment of a BPaaS in case any kind of problematic situation occurs. This information will also need to be structured accordingly. In this case, we foresee that probably SRL / CAMEL will be exploited as well. In any case, such deployment adaptation rules should be imprinted in the BPaaS bundle after being possibly revised by the BPaaS Allocation Environment. As in the case of event pattern detection, such rules might also unveil that some deployments always or frequently lead to errors such that they need to be avoided by being replaced via exploiting alternative cloud services. They can also unveil that certain component configurations are problematic such that they can be replaced by alternative ones.

5 SUMMARY: RESEARCH SHOWROOM

This chapter first summarises the main research assets / blueprints that are offered by the BPaaS Evaluation environment in the form of a research showroom which explicates also some important aspects related to their adoption, like their maturity level. Then, it explains in short the blueprint handover process. Finally, it summarises this deliverable and paves the way for future work to be incarnated in the next and last deliverable of T3.3, D3.6 - BPaaS Monitoring and Evaluation Prototypes.

5.1 Research assets

The presentation of the research blueprints follows the one in D3.3 [19] and is carried out in a table-based way. The columns in the tables span the following information aspects:

- the name of the blueprint
- short description of its main functionality
- the dependencies that this blueprint may have with other blueprints or components in the CloudSocket architecture.
- short explanation of the dependencies per each component on which the asset depends
- the type of the blueprint / asset signifying whether it is an extension of an existing asset or a new asset
- the research state of the asset which maps to its maturity level, while it also explicates the expected time of its delivery; the following states have been considered along with their respective expected delivery times (EDT): (a) *idea* mapping to 12 months EDT; (b) *concept* mapping to 9 months EDT; (c) *in progress* mapping to 6 months EDT; (d) *alpha* mapping to 3 months EDT.
- the licence involved in the offering of each blueprint

Thus, please find below the respective table-based summarisation of the blueprints according to the two main categories: BPaaS (evaluation) modelling and analysis.

	BPaaS Evaluation & Monitoring Modelling Blueprints
Name	**1. Evaluation Ontology (cf. section 2.2)**
Summary	Enables to capture whole BPaaS (deployment) hierarchies.
Dependencies	-
Dependency Explanation	-
Asset Type	New asset
Research State	Alpha version
License	Open-Source
Name	**2. OWL-Q KPI Extension (cf. section 2.3)**
Summary	Extend OWL-Q [40] to cover additional information aspects for supporting the computation of KPI metrics, the evaluation of KPIs and the KPI drill-down analysis.

Dependencies	-
Dependency Explanation	-
Asset Type	Extension of OWL-Q
Research State	Alpha version
License	Open-Source

	BPaaS Evaluation Harvesting & Analysis Blueprints
Name	**3. Information Harvesting and Linking (cf. section 3.3)**
Summary	Enables to collect information from different information sources, to semantically enhance and link it and to store it in the Semantic KB
Dependencies	*Workflow Engine*, *Cloud Provider Engine*, *Marketplace*, *Registry*, *Meta-Model Repository*, *Semantic KB Service*
Dependency Explanation	Depends on the *Workflow Engine* in order to obtain information about workflows and their execution, on the *Cloud Provider Engine* in order to retrieve information about the deployment of internal service components and the monitoring of BPaaS components, on the *Registry* for obtaining extra information about cloud services, on the *Marketplace* for retrieving organisational information, on the Meta-Model repository for obtaining information about BPs and KPIs, and on the *Semantic KB Service* for the storage of the semantically-enhanced and linked information
Asset Type	New asset
Research State	Alpha version
License	Open-Source
Name	**4. KPI Analysis (cf. section 2.3)**
Summary	Features the computation of KPI metrics, the evaluation of KPIs and the KPI drill-down analysis
Dependencies	*Harvester*, OWL-Q KPI Extension, Evaluation Ontology, *Semantic KB Service*
Dependency Explanation	Depends on the *Harvester* as without this component the *Semantic KB* will be empty, on the two ontology blueprints to have a complete account on the way the semantic information can be accessed, and on the *Semantic KB Service* to have the capability to pose SPARQL queries over the *Semantic KB*.
Asset Type	New asset
Research State	Alpha version
License	Open-Source

Name	**5. Best Deployment Discovery (cf. section 3.5.2)**
Summary	Supports the discovery of best deployments for BPaaSes
Dependencies	*Harvester*, OWL-Q KPI Extension, Evaluation Ontology, *Semantic KB Service*
Dependency Explanation	Depends on the *Harvester* as without this component the *Semantic KB* will be empty, on the two ontology blueprints to have a complete account on the way the semantic information can be accessed, and on the *Semantic KB Service* to have the capability to pose SPARQL queries over the *Semantic KB* in order to populate the respective internal KB.
Asset Type	Extension of existing asset produced in the PaaSage project
Research State	In progress
License	Open-Source
Name	**6. Event Pattern Detection (cf. section 3.6)**
Summary	Enables the discovery of event patterns leading to the violation of KPIs / SLOs. Can also derive adaptation rules as mappings between event patterns and respective adaptation strategies that can be employed to resolve the respective KPI / SLO violations under examination.
Dependencies	*Harvester*, OWL-Q KPI Extension, Evaluation Ontology, *Semantic KB Service*
Dependency Explanation	Depends on the *Harvester* as without this component the *Semantic KB* will be empty, on the two ontology blueprints to have a complete account on the way the semantic information can be accessed, and on the *Semantic KB Service* to have the capability to pose SPARQL queries over the *Semantic KB* to retrieve the information required for the actual analysis.
Asset Type	Extension of previous work [70]
Research State	In progress
License	Open-Source
Name	**7. Process Mining (cf. section 3.7)**
Summary	Currently supports the execution of existing process mining algorithms. Will enable to make them semantics-aware. Might incorporate extensions of existing as well as new process mining algorithms.
Dependencies	*Harvester*, OWL-Q KPI Extension, Evaluation Ontology, *Semantic KB Service*
Dependency Explanation	Depends on the *Harvester* as without this component the *Semantic KB* will be empty, on the two ontology blueprints to have a complete account on the way the semantic information can be accessed, and on the *Semantic KB Service* to have the capability to pose SPARQL queries over the *Semantic KB* in order to retrieve the information required for the actual (process) mining.
Asset Type	New asset (but relies on existing process mining algorithms)

Research State	In progress
License	Open-Source

From the above analysis, it can be easily seen that 4 assets are already in alpha state and 3 are in progress. By considering that in progress means that around 6 months are needed to finish the actual implementation, we believe that we are in a good position to deliver the latter 3 components. Concerning the first 4, they have or will be soon adopted in the current implementation of the BPaaS Evaluation Environment in WP4.

5.2 Blueprint handover process

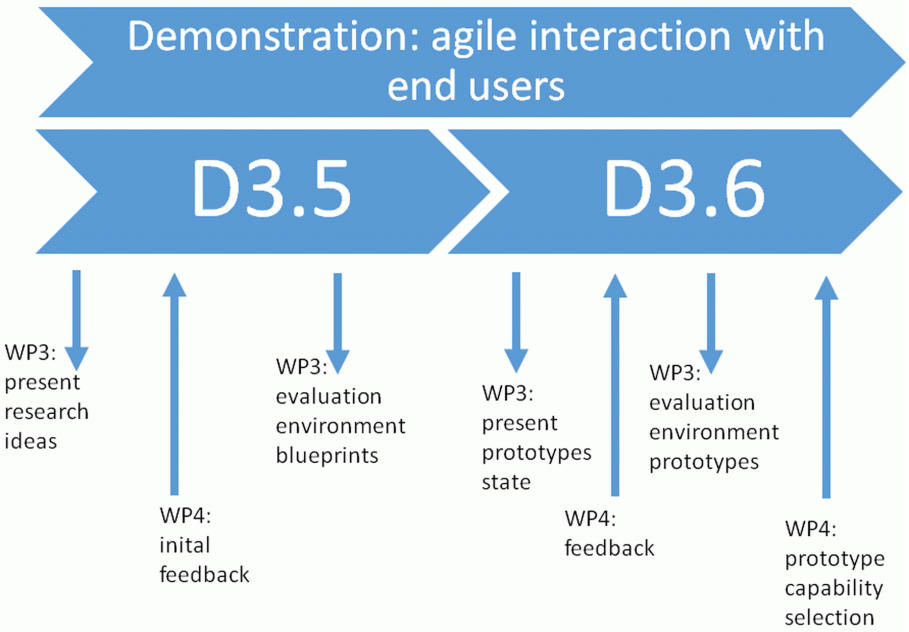

Figure 12 - The blueprint handover process

The blueprint handover process is depicted in Figure 12 and is similar to the process that has been adopted by T3.2. The similarities concern that the two processes involve presentation of the research ideas and the subsequent prototypes state (in GA meetings, first one has already been done in Olten about the research ideas) in order to obtain feedback from WP4 and the end users as well as the production of the respective blueprints and prototypes. The main difference concerns the fact that there is a perfect match between blueprints and respective components in the implementation of the BPaaS Evaluation Environment in WP4. In this respect, there is no need for any prioritisation. In fact, the final decision to be made by WP4 would be not the selection of a prototype but of the respective capabilities that this prototype will offer. This will also depend on whether all capabilities have been realised in the end or not. For instance, concerning the Process Mining module, WP4 might select only the capability to execute a process mining algorithm, the capability to also operate over semantic process logs or the whole set of capabilities that might include extensions of existing or new process mining algorithms. However, before performing any kind of selection, WP4 will have all the information needed to make this selection, including the current state of the prototypes / blueprints and of their respective capabilities. To assist in this final selection, the next deliverable, D3.6 will provide a more advanced form of summarisation which will focus on the individual capabilities of each (research) prototype / blueprint.

5.3 Summary and Future Work

This deliverable is the first in the series of deliverables involved in T3.3 that deal with the final production of research prototypes for BPaaS monitoring and evaluation. In this respect, it has presented some initial ideas, concepts and work in progress in the form of blueprints that have the potential to become actual research prototypes. The final respective prototypes will of course be reported in the next and last deliverable in this series named as D3.6 - BPaaS Monitoring and Evaluation Prototypes.

The current blueprints that have been proposed enable the modelling of deployment and monitoring information about a BPaaS as well as conducting different kinds of analysis results over this information. The current analysis kinds to be supported include the capabilities to evaluate KPIs and support their drill-down, to discover best deployments, to detect event patterns leading to the violation of KPIs / SLOs and to perform process mining to highlight particular BPaaS process discrepancies. Some of these blueprints are in progress while others have already reached an alpha state. This means that the quality and amount of work has been substantial. This also promises that in the end the in-progress blueprints will eventually be realised according to the time plan of T3.3. The blueprint delivery process that has been specified in the previous section guarantees this by also enabling obtaining fruitful feedback that will certainly enable optimising existing capabilities, accelerating the realisation of concept-based blueprints and also focusing on only capabilities that can have a major impact to WP4 and especially the end users.

Apart from transforming the blueprints into actual prototypes, this deliverable has also paved for some work directions that might be followed in the near future. Some directions seem to be promising while they do not require significant effort. Other directions need some exploration in order to reach the state of concept which could then be realised. In any case, even if the CloudSocket brokers are presented with the current set of ready or nearly ready capabilities, they are already equipped with a significant set of tools that enable them to produce different kinds of business intelligence knowledge that supports the completion of the BPaaS lifecycle and the respective BPaaS optimisation.

6 REFERENCES

[1] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB. pp. 487–499 (1994).

[2] Juan Jose Alfaro, Raul Rodriguez-Rodriguez, María José Verdecho, and Angel Ortiz. 2009. Business process interoperability and collaborative performance measurement. *International Journal of Computer Integrated Manufacturing* 22, 9, 877-889.

[3] A. K. Alves De Medeiros, C. Pedrinaci, W. M. P. van der Aalst, J. Domingue, M. Song, A. Rozinat, B. Norton, and L. Cabral. 2007. An outlook on semantic business process mining and monitoring. In *Proceedings of the 2007 OTM Confederated international conference on On the move to meaningful internet systems - Volume Part II (OTM'07)*, Robert Meersman, Zahir Tari, and Pilar Herrero (Eds.), Vol. Part II. Springer-Verlag, Berlin, Heidelberg, 1244-1255.

[4] Alves De Medeiros, A.K., Weijters, A.J.M.M., Aalst, W.M.P. van der (2007). Genetic process mining : a basic approach and its challenges. Data Mining and Knowledge Discovery, 14(2), 245-304, 2007.

[5] Artikis, A., Sergot, M.J., Paliouras, G.: Run-time composite event recognition. In: DEBS. pp. 69–80. ACM (2012)

[6] Bettini, C., Wang, X.S., Jajodia, S., Lin, J.L.: Discovering frequent event patterns with multiple granularities in time sequences. IEEE Trans. Knowl. Data Eng. 10(2), 222–237 (1998).

[7] Andrea Burattin, and Alessandro Sperduti. Heuristics Miner for Time Intervals. 2010. In European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges, Belgium.

[8] S. Cacciatore, S. Cuomo, K. Hinkelmann, J. Iranzo Yuste, J. Jähnert, B. Lammel, Y. Liang, P. Martos, S. Naldini, and R. Woitsch, 'D5.2 – BPaaS Reference Models', CloudSocket project deliverable, 2016.

[9] Fabio Casati, Malu Castellanos, Umeshwar Dayal, and Ming-Chien Shan. 2006. A metric definition, computation, and reporting model for business operation analysis. In *Proceedings of the 10th international conference on Advances in Database Technology (EDBT'06)*, Yannis Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, and Mike Hatzopoulos (Eds.). Springer-Verlag, Berlin, Heidelberg, 1079-1083.

[10] Malu Castellanos, Fabio Casati, Ming-Chien Shan, and Umesh Dayal. 2005. iBOM: A Platform for Intelligent Business Operation Management. In *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*. IEEE Computer Society, Washington, DC, USA, 1084-1095.

[11] Chowdhary, K. Bhaskaran, N. S. Caswell, H. Chang, T. Chao, S.-K. Chen, M. Dikun, H. Lei, J.-J. Jeng, S. Kapoor, C. A. Lang, G. Mihaila, I. Stanoi, and L. Zeng. 2006. Model driven development for business performance management. *IBM Syst. J.* 45, 3 (July 2006), 587-605.

[12] Marco Comuzzi, Kyriakos Kritikos, and Pierluigi Plebani. 2009. A Semantic Based Framework for Supporting Negotiation in Service Oriented Architectures. In *Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing (CEC '09)*. IEEE Computer Society, Washington, DC, USA, 137-145.

[13] Claire Costello and Owen Malloy. 2008. Building a Process Performance Model for Business Activity Monitoring. *Information Systems Development - Challenges in Practice, Theory, and Education*, Wita Wojtkowski, Gregory Wojtkowski, Michael Lang, Kieran Conboy, and Chris Barry (Eds). Springer-Verlag, 237-248.

[14] Mate J. Csorba, Hein Meling, and Poul E. Heegaard. 2010. Ant system for service deployment in private and public clouds. In *Proceedings of the 2nd workshop on Bio-inspired algorithms for distributed systems (BADS '10)*. ACM, New York, NY, USA, 19-28.

[15] Yi Cui and Klara Nahrstedt. 2001. QoS-Aware Dependency Management for Component-Based Systems. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC '01)*. IEEE Computer Society, Washington, DC, USA.

[16] Sofie De Cnudde, Jan Claes, and Geert Poels. 2014. Improving the quality of the Heuristics Miner in ProM 6.2. *Expert Syst. Appl.* 41, 17 (December 2014), 7678-7690.

[17] Massimiliano de Leoni, Marlon Dumas, and Luciano García-Bañuelos. 2013. Discovering branching conditions from business process execution logs. In *Proceedings of the 16th international conference on Fundamental Approaches to Software Engineering (FASE'13)*, Vittorio Cortellessa and Dániel Varró (Eds.). Springer-Verlag, Berlin, Heidelberg, 114-129.

[18] Adela del-Río-Ortega, Manuel Resinas, Amador Durán, and Antonio Ruiz-Cortés. 2016. Using templates and linguistic patterns to define process performance indicators. Enterp. *Inf. Syst.* 10, 2 (February 2016), 159-192.
[19] J. Domaschka, F. Griesinger, J. Iranzo Yuste, K. Kritikos, D. Seybold, R. Sosa, and C. Zeginis, 'D3.3 – BPaaS Allocation and Execution Environment Blueprints', CloudSocket project deliverable, 2016.
[20] J. Domaschka, F. Griesinger, J. Iranzo Yuste, K. Kritikos, D. Seybold, R. Sosa, and C. Zeginis, 'D3.4 – BPaaS Allocation and Execution Environment Prototypes', CloudSocket project deliverable, 2016.
[21] Martin Duggan, Kieran Flesk, Jim Duggan, Enda Howley, and Enda Barrett. 2016. A Reinforcement Learning Approach for Dynamic Selection of Virtual Machines in Cloud Data Centres. In *Sixth International Conference on Innovative Computing Technology (INTECH 2016)*. Dublin, Ireland.
[22] Christian Ensel. Automated generation of dependency models for service management. 1999. In *Workshop of the Open View University Association*. OVUA.
[23] Diogo R. Ferreira and Lucinéia H. Thom. 2012. A Semantic Approach to the Discovery of Workflow Activity Patterns in Event Logs. *Int. J. of Business Process Integration and Management* 6, 1, 4-17.
[24] Fittkau, F., Frey, S. and Hasselbring, W. (2012) CDOSim: Simulating Cloud Deployment Options for Software Migration Support In: *Proceedings of the 6th IEEE International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA 2012)*, 24. Sep. 2012, Riva del Garda, Italy.
[25] Ulrich Frank, David Heise, Heiko Kattenstroth, and Hanno Schauer. 2008. Designing and utilising business indicator systems within enterprise models: Outline of a method. In: *Modellierung betrieblicher Informationssysteme - MobIS 2008 : Modellierung zwischen SOA und Compliance Management*. Saarbrücken, Germany, 27. - 28.
[26] Jan-Philipp Friedenstab, Christian Janiesch, Martin Matzner, and Oliver Muller. 2012. Extending BPMN for Business Activity Monitoring. In *Proceedings of the 2012 45th Hawaii International Conference on System Sciences (HICSS '12)*. IEEE Computer Society, Washington, DC, USA, 4158-4167.
[27] Johny Ghattas, Pnina Soffer, and Mor Peleg. 2014. Improving business process decision making based on past experience. *Decis. Support Syst.* 59, 93-107.
[28] Oscar González, Rubby Casallas, and Dirk Deridder. 2009. MMC-BPM: A Domain-Specific Language for Business Processes Analysis. In *Proceedings of the 12th International Conference on Business Information Systems (BIS)*, Witold Abramowicz (Ed.). Springer, Poznan, Poland, 157-168.
[29] Boris Gruschke. Integrated Event Management: Event Correlation Using Dependency Graps. 1998. In *DSOM*.
[30] Hasselmeyer, Peer (2001) Managing Dynamic Service Dependencies. In: *12th International Workshop on Distributed Systems: Operations & Management.*, 15 October 2001 - 17 October 2001, Nancy, France.
[31] Hellerstein, J.L., Ma, S., Perng, C.S.: Discovering actionable patterns in event data. IBM Systems Journal 41(3), 475–493 (2002).
[32] Andreas Heß, Eddie Johnston, and Nicholas Kushmerick. 2004. ASSAM: a tool for semi-automatically annotating semantic web services. In *Proceedings of the 3rd International Conference on Semantic Web Conference (ISWC)*, Sheila A. McIlraith, Dimitris Plexousakis, and Frank Van Harmelen (Eds.). Springer-Verlag, Berlin, Heidelberg, 320-334.
[33] K. Hinkelmann, S. Kurjakovic, B. Lammel, E. Laurenzi, and R. Woitsch, 'D3.2 – Modelling Prototypes for BPaaS', CloudSocket project deliverable, 2016.
[34] C. Hwang and K. Yoon, "Multiple Criteria Decision Making," Lect. Notes Econ. Math., 1981.
[35] Kaplan R. S., Norton D. P., "The Balanced Scorecard – Measures that Drive Performance", Harvard Business Review, n°1, January-February.
[36] Karagiannis, D., 1995. BPMS: Business Process Management Systems: Concepts, Methods and Technologies, SIGOIS Special Issue, SIGOIS Bulletin 10-13.
[37] K. E. Kritikos, 'QoS-Based Web Service Description and Discovery', PhD Thesis, University of Crete, 2008.
[38] K. Kritikos, K. Magoutis, and D. Plexousakis, `Towards Knowledge-Based Assisted IaaS Selection', in *CloudCom*, IEEE, 2016.
[39] K. Kritikos, P. Plebani, and D. Plexousakis, 'Semantic SLAs for Services with Q-SLA', in *Cloud Forward*, 2016.
[40] K. Kritikos, and D. Plexousakis, 'Semantic QoS Metric Matching', in *ECOWS*, 2006.
[41] K. Kritikos and D. Plexousakis, 'Novel Optimal and Scalable Nonfunctional Service Matchmaking Techniques', *Serv. Comput. IEEE Trans. On*, vol. 7, no. 4, pp. 614–627, 2014.

[42] Beate List and Birgit Korherr. 2006. An evaluation of conceptual business process modelling languages. In *Proceedings of the 2006 ACM symposium on Applied computing (SAC '06)*. ACM, New York, NY, USA, 1532-1539.
[43] Rong Liu, Anil Nigam, Jun-Jang Jeng, Chian-Rou Shieh, and Frederick Y Wu. 2010. Integrated Modeling of Performance Monitoring with Business Artifacts. In *IEEE 7th International Conference on e-Business Engineering (ICEBE)*. IEEE, 64--71.
[44] Azeem Lodhi, Veit Küppen, and Gunter Saake. 2011. An Extension of BPMN Meta-model for Evaluation of Business Processes. *Sci. J. Riga Tech. Univ.* 43, 1 (January 2011), 27-34.
[45] Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, and Wil M. P. van der Aalst. 2016. In *28th International Conference on Advanced Information Systems Engineering* (CAiSE 2016). Springer-Verlag, Berlin, Heidelberg, 377-392.
[46] Seyed-Saeid Masoumzadeh and Helmut Hlavacs. 2013. Integrating VM Selection Criteria in Distributed Dynamic VM Consolidation Using Fuzzy Q-Learning. In *7th International DMTF Academic Alliance Workshop on Systems and Virtualization Management: Standards and the Cloud (SVM13)*. Zurich, Switzerland.
[47] Motta G., Longo A., HIGO, Misurare le prestazioni dei processi gestionali, Marchio registrato 2 Aprile 2006 [REGISTERED trademark], LE2006 e 000114, 2006; Motag:, Pignatelli G., “Disegnare processi performanti”, *Sviluppo e Organizzazione, n 221*, maggio-giugno 2007.
[48] Gianmario Motta, Giovanni Pignatelli, and Manuela Florio. 2007. Performing Business Process Knowledge Base. In *First Internation Workshop and Summer School on Service Science*. Heraklion, Crete.
[49] Andy Neely, John Mills, Ken Platts, Huw Richards, Mike Gregory, Mike Bourne and Mike Kennerley. 2000. Performance measurement system design: developing and testing a process-based approach. *International Journal of Operations & Production Management* 20, 10, 1119-1145.
[50] D. Palma, M. Rutkowski, and T. Spatzier, *TOSCA Simple Profile in YAML Version 1.0*. 2015.
[51] Alfonso Pierantonio, Gianni Rosa, Darius Silingas, Barbara Thönssen, and Robert Woitsch. 2015. Metamodeling Architectures for Business Processes in Organizations. In *Proceedings of the Projects Showcase, part of the Software Technologies: Applications and Foundations 2015 federation of conferences (STAF 2015)*. CEUR, L'Aquila, Italy, 27-35.
[52] Florian Rosenberg, Anton Michlmayr, and Schahram Dustdar. 2008. Top-down business process development and execution using quality of service aspects. *Enterp. Inf. Syst.* 2, 4 (November 2008), 459-475.
[53] A. Rossini, K. Kritikos, N. Nikolov, J. Domaschka, F. Griesinger, D. Seybold, D. Romero, D2.1.3 –CloudML Implementation Documentation (Final version), PaaSage project deliverable, 2015.
[54] A. Rozinat and W. M. P. van der Aalst. 2006. Decision mining in prom. In *Proceedings of the 4th international conference on Business Process Management (BPM'06)*, Schahram Dustdar, José Luiz Fiadeiro, and Amit P. Sheth (Eds.). Springer-Verlag, Berlin, Heidelberg, 420-425.
[55] T. L. Saaty, *The Analytic Hierarchy Process, Planning, Piority Setting, Resource Allocation*. New york: McGraw-Hill, 1980.
[56] F. Samreen, Y. EL Khatib, M. C. Rowe, G. S. Blair, "Daleel: Simplifying Cloud Instance Selection Using Machine Learning," *IEEE/IFIP Network Operations and Management Symposium*, 2015.
[57] Stefan Seedorf and Martin Schader. 2011. Towards an Enterprise Software Component Ontology. In *AMCIS*. Detroit, Michigan, USA.
[58] Sim, A.T.H., Indrawan, M., Zutshi, S., Srinivasan, B.: Logic-based pattern discovery. IEEE Trans. Knowl. Data Eng. 22(6), 798–811 (2010).
[59] Sima Soltani, Patrick Martin, Khalid Elgazzar, "QuARAMRecommender: Case-Based Reasoning for IaaS Service Selection,” *The International Conference on Cloud and Autonomic Computing (CAC 2014)*, Imperial College, London, September 8-12, 2014.
[60] Minseok Song and Wil M. P. van der Aalst. 2008. Towards comprehensive support for organizational mining. *Decis. Support Syst.* 46, 1 (December 2008), 300-317.
[61] Linh Thao Ly, Conrad Indiono, Jürgen Mangler, and Stefanie Rinderle-Ma. 2012. Data transformation and semantic log purging for process mining. In *Proceedings of the 24th international conference on Advanced Information Systems Engineering (CAiSE'12)*, Jolita Ralyté, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza (Eds.). Springer-Verlag, Berlin, Heidelberg, 238-253.
[62] Manoj Thomas, Richard Redmond, Victoria Yoon, and Rahul Singh. 2005. A semantic approach to monitor business process. *Commun. ACM* 48, 12 (December 2005), 55-59.
[63] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, 16(9):1128–1142, 2004.

[64] A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros. Process Mining with the Heuristics Miner-algorithm. BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006.
[65] Branimir Wetzstein, Dimka Karastoyanova, and Frank Leymann. 2008. Towards Management of SLA-Aware Business Processes Based on Key Performance Indicators. In *BPMDS*. Montpellier, France.
[66] Branimir Wetzstein, Philipp Leitner, Florian Rosenberg, Ivona Brandic, Schahram Dustdar, and Frank Leymann. 2009. Monitoring and analyzing influential factors of business process performance. In *Proceedings of the 13th IEEE international conference on Enterprise Distributed Object Computing* (EDOC'09). IEEE Press, Piscataway, NJ, USA, 118-127.
[67] Branimir Wetzstein, Zhilei Ma, and Frank Leymann. 2008. Towards Measuring Key Performance Indicators of Semantic Business Processes. In *Proceedings of 11th International Conference on Business Information Systems (BIS 2008)*. Springer-Verlag, Innsbruck, Austria, page 227--238.
[68] P. Xiong, C. Pu, X. Zhu, and R. Griffith, "vperfguard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments," in *ICPE*. New York, NY, USA: ACM, 2013, pp. 271–282.
[69] C. Zeginis, K. Kritikos, P. Garefalakis, K. Konsolaki, K. Magoutis, and D. Plexousakis, 'Towards cross-layer monitoring of multi-cloud service-based applications', in *Service-Oriented and Cloud Computing*, Springer, 2013, pp. 188–195.
[70] Zeginis, C., Kritikos, K., Plexousakis, D. (2014). Event Pattern Discovery for Cross-Layer Adaptation of Multi-cloud Applications. In *ESOCC*, 138-147.
[71] Xingquan Zuo, Guoxiang Zhang, and Wei Tan. 2014. Self-adaptive learning PSO based deadline constrained task scheduling for hybrid IaaS cloud. *IEEE Trans Autom Sci Eng* 11, 2, 564 - 573.
[72] Khurram Shahzad and Paul Johannesson. 2009. An evaluation of process warehousing approaches for business process analysis. In *Proceedings of the International Workshop on Enterprises & Organizational Modeling and Simulation* (EOMAS '09). ACM, New York, NY, USA.
[73] G. Horn, "A vision for a stochastic reasoner for autonomic cloud deployment," in *Second Nordic Symposium on Cloud Computing & Internet Technologies (NordiCloud '13)*. ACM, September 2013, pp. 46–53. [Online]. Available: http://doi.acm.org/10.1145/2513534.2513543
[74] Nirnay Ghosh, Soumya K. Ghosh, and Sajal K. Das. 2015. SelCSP: A Framework to Facilitate Selection of Cloud Service Providers. *IEEE Trans. Cloud Computing* 3, 1 (Jan.-March 2015), 66-79.
[75] Florian Lautenbacher, Bernhard Bauer and Sebastian Förg. 2009. Process Mining for Semantic Business Process Modeling. In *Second International Workshop on Dynamic and Declarative Business Processes (DDBP) at EDOC Conference*. IEEE Computer Society, Auckland, New Zealand.
[76] K. Kritikos, J. Domaschka, and A. Rossini, 'SRL: A Scalability Rule Language for Multi-Cloud Environments', in *6th International Conference on Cloud Computing Technology and Science (CloudCom)*, IEEE, 2014, pp. 1–9.
[77] A. Papaioannou, 'QoS Design and Implementation of a System for Evaluating Distributed Application Deployments in Multi-Clouds', PhD Thesis, University of Crete, 2013.
[78] Keller, A. and Ludwig, H. (2003). The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. Journal of Network and Systems Management, 11(1):57–81.

ANNEX A: LIST OF ABBREVIATIONS

List of abbreviation used into the document:

- API: Application Programming Interface
- BPaaS: Business Process as a Service
- BPEL: Business Process Execution Language
- BPMN: Business Process Model and Notation
- CAMEL: Cloud Application Execution Modelling Language
- CEP: Complex Event Processing
- DMN: Decision Model and Notation
- DSL: Domain Specific Lanugage
- IaaS: Infrastrucutre as a Service
- JVM: Java Virtual Machine
- KPI: Key Performance Indicator
- QoS: Quality of Service
- PaaS: Platform as a Service
- RDBMS : Relational Database Management Systems
- REST: Representational State Transfer
- SaaS: Software as a Service
- SLA: Service Level Agreement
- SLO: Service Level Objective
- SOA: Service Oriented Architecture
- SRL: Scalability Rule Language
- SWRL: Semantic Web Rule Language
- WSML: Web Service Modelling Language
- TSDB: Time Series Database
- VM: Virtual Machine
- WSDL: Web Service Description Language
- WAR: Web application ARchive