

KNOWLEDGE PROVISION FOR CLOUDSOCKET: FINDINGS AND FEEDBACK

D6.2

Editor Name	Knut Hinkelmann (FHNW), Emanuele Laurenzi (FHNW)
Submission Date	December 31, 2017
Version	1.0
State	FINAL
Confidentially Level	PU



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

EXECUTIVE SUMMARY

Research and development in CloudSocket followed an iterative approach with research and technical development running in parallel. Based on ongoing research, further development and evaluations, the first solutions were enhanced in the second and third project year.

The purpose of task 6.3 was an assessment of the innovation items, which are evaluated using the concept of an innovation shop and an innovation scorecard to document the validation.

Hence, the purpose of this deliverable is to describe the progress made. This is done by presenting and analysing the innovation scorecard and by describing the final development of the research and innovation items, together with the respective knowledge acquired and lessons learned that were achieved during the lifecycle of the CloudSocket project.

The innovation scorecard shows a high maturity of the innovation items. For nearly all innovation items there exist peer-reviewed publications and they are implemented as a prototype. A majority is already integrated in a tool by the technology partners and thus ready for exploitation.

Furthermore, the analysis of the innovation scorecard validates, that the agile project approach with the parallelisation of research (WP3) and implementation (WP4) was very efficient, because the research results (i.e. innovation items) could be integrated in the second implementation cycle, which itself was challenged by the demonstration (WP5).

For each environment of the high-level architecture, the document contains a description of the innovation items and the tools, which is then followed by a section describing the lessons learned. The latter gives an explanation on how the solutions (described in previous deliverables) were improved until the final results, and is a critical reflection of the current solutions. It thus addresses also possibilities for further research and development.

PROJECT CONTEXT

Workpackage	WP6: Validation
Task	T6.2
Dependencies	Based on D3.1, D3.2, D3.3, D3.4, D3.5,

Contributors and Reviewers

Contributors	Reviewers
Emanuele Laurenzi, Knut Hinkelmann (FHNW)	Kyriakos Kritikos (FORTH)
Damiano Falcioni (BOC)	Stefan Wesner (UULM)
Frank Griesinger, Daniel Seybold, Jörg Domaschki (UULM)	Robert Woitsch (BOC)
Kyriakos Kritikos, Chrysostomos Zeginis (FORTH)	
Joaquin Iranzo Yuste(ATOS)	
Simone Cacciatore (FHOSTER)	
Constantin Valeriu Tuguran (YMENS)	

Approved by: Robert Woitsch (BOC), as project coordinator

Version History




Version	Date	Authors	Sections Affected
0.1	September 30, 2017	Knut Hinkelmann Emanuele Laurenzi	List of Sections
0.1	October 13, 2017	Emanuele Laurenzi	Updated structure of the document
0.1	November 15, 2017	all contributors	Providing individual contributions
0.2	November 23, 2017	Emanuele Laurenzi	Merging of individual contributions into one document
0.3	December 08, 2017	Emanuele Laurenzi Knut Hinkelmann	BPaaS Design Environment Introduction
0.5	December 09, 2017	Knut Hinkelmann	Revision of all parts, Conclusion, Abstract
0.8	December 18, 2017	Knut Hinkelmann	Removing tools, restructuring
0.9	December 22, 2017	Emanuele Laurenzi Knut Hinkelmann	new innovation scorecard; further information for lessons learned, addressing partners feedback, update introduction
1.0	December 23, 2017	Knut Hinkelmann	final revision

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

1	Introduction.....	8
1.1	Innovation Items in the Innovation Shop.....	8
1.2	Innovation Scorecard.....	9
1.3	Structure of the document.....	11
2	BPaaS Design Environment.....	12
2.1	Innovation Items.....	12
2.1.1	BPaaS Semantic Modeler.....	12
2.1.2	The Context-Adaptive Questionnaire for Service Selection.....	15
2.1.3	DMN-based Ontology Query.....	24
2.1.4	DMN and BPMN 2.0 Prototype.....	25
2.2	Lessons Learned.....	26
2.2.1	BPaaS Semantic Modeler, Context-Adaptive Questionnaire and DMN-based Query Ontology.....	26
2.2.2	DMN and BPMN 2.0 Prototype.....	27
3	BPaaS Allocation Environment.....	28
3.1	Innovation Items.....	28
3.1.1	CAMEL Language.....	28
3.1.2	DMN-to-CAMEL-Mapper.....	29
3.1.3	Smart Service Discovery and Composition.....	30
3.2	Lessons Learned.....	32
3.2.1	CAMEL Lessons Learned.....	32
3.2.2	DMN-to-CAMEL-Mapper Lessons Learned.....	33
3.2.3	Smart Service Discovery and Composition Tool Lessons Learned.....	33
4	BPaaS Execution Environment.....	35
4.1	Innovation Items.....	35
4.1.1	BPaaS Orchestration.....	35
4.1.2	Cross-layer & Cross-Level Adaptation Management.....	36
4.2	Lessons Learned.....	39
4.2.1	<i>BPaaS Orchestration</i>	39
4.2.2	<i>Cross-layer Monitoring and Adaptation</i>	39
5	BPaaS Evaluation Environment.....	41
5.1	Innovation Items.....	41
5.1.1	Information Harvesting.....	41
5.1.2	Conceptual Analysis Engine.....	42
5.1.3	Hybrid Business Dashboard.....	44

5.2	Lessons Learned	44
5.2.1	Lessons learned from Information Harvesting	44
5.2.2	Lessons learned from KPI evaluation & drilldown.....	45
5.2.3	Lessons learned from Hybrid Business Dashboard.....	45
6	Summary and Conclusion.....	47
7	References	48

LIST OF FIGURES

- Figure 1- High-level architecture of the CloudSocket platform 8
- Figure 2. BPMN 2.0 extension to accommodate business process requirements..... 13
- Figure 3. BPMN 2.0 extension to accommodate cloud service and workflow descriptions 14
- Figure 4. BPaaS Ontology..... 14
- Figure 5. Hierarchical service identification 16
- Figure 6. The action selection for the functional requirements question..... 17
- Figure 7. Interface for Cloud Service Specifications..... 24
- Figure 8. Decision tables reflecting semantic rules for Availability and Backup Frequency..... 25
- Figure 9. DMN-based Ontology Query Prototype..... 25
- Figure 10 Dynamic deployment modelling..... 29
- Figure 11 High-level Cloudiator architecture. 36
- Figure 12: The combined cross-layer BPaaS monitoring framework..... 37
- Figure 13: The combined Cross-Layer BPaaS Adaptation Framework 38

LIST OF TABLES

- Table 1. Innovation Scorecard..... 9
- Table 2 Deliverables affected by evolution of innovation items and tools..... 11
- Table 3. Non-functional Specifications 18
- Table 4 Image and Region Decision Table..... 30

1 INTRODUCTION

Research and development in CloudSocket followed an iterative approach with research and technical development running in parallel. For all phases of the BPaaS approach (1. Design, 2. Allocation, 3. Execution, 4. Evaluation) a first prototypical solution was developed during the first half of the project until month 18 and described in the respective deliverables. Based on ongoing research, further development and evaluations, the solutions were enhanced in the second and third year. Subsequently, the solutions described in the earlier deliverables might not reflect the latest status of the innovation items.

The purpose of this deliverable is twofold. First, it describes the latest status of the research and innovation items. Second, it explains – based on an assessment of the first solution - the evolution of the innovation items during the lifecycle of the CloudSocket project. It thus explains the deviations in the original solutions with respect to the final ones, which are described in earlier deliverables.

1.1 Innovation Items in the Innovation Shop

The innovation items of the CloudSocket projects are made available via the innovation shop on the CloudSocket website. The innovation shop is structured according to the high-level BPaaS Reference architecture (see Figure 1), which is divided into four environments that correspond to the four phases of the established Business Process Management System (BPMS) paradigm. The BPaaS Environments are (a) BPaaS Design Environment describes business processes, business requirements and workflows, (b) BPaaS Allocation Environment linking deployable workflows with concrete services, (c) BPaaS Execution Environment that operates, executes and monitors the workflow, and (d) BPaaS Evaluation Environment that lifts key performance indicators back to business level). Additionally (e) the BPaaS Marketplace is required to enable the customer to buy the BPaaS. The architecture is specified in deliverable D4.5 [47] that defines the aforementioned BPaaS environments as loosely coupled, and hence exchangeable parts.

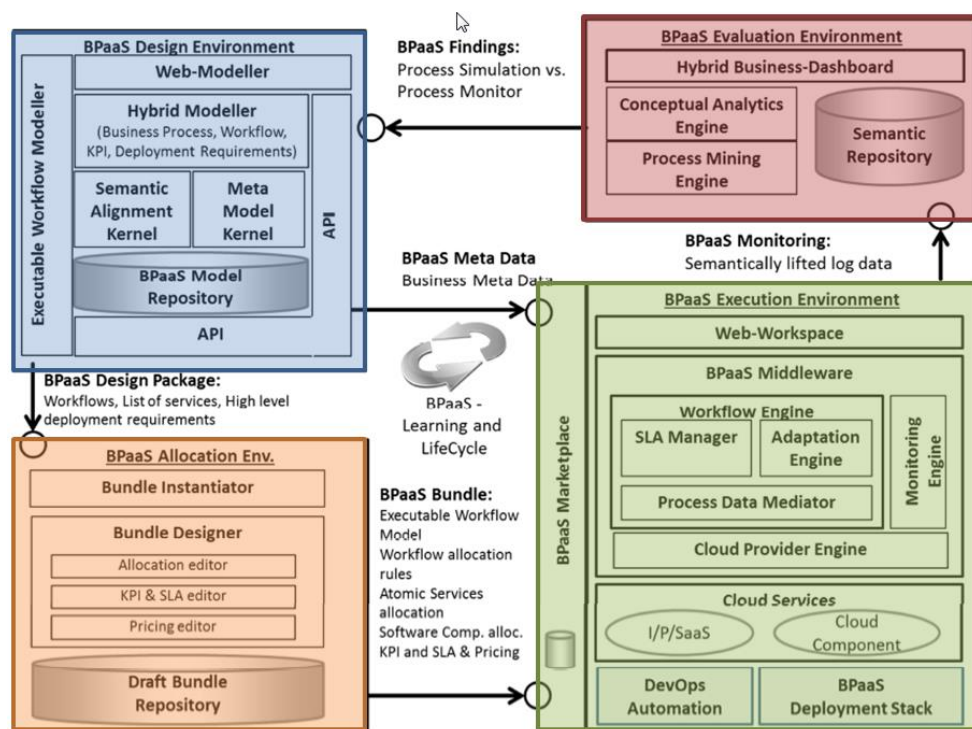


Figure 1- High-level architecture of the CloudSocket platform

1.2 Innovation Scorecard

Table 1 shows the innovation scorecard, which for each innovation item provides the assessment of the following maturity indicators:

- the BPaaS phase in which an innovation item was developed,
- publication(s) in which an innovation item appears (if any),
- whether an innovation item has a standalone prototype (the symbol “+” and “-” are used respectively) or not,
- whether an innovation item is integrated in a tool (the symbol “+” and “-” are used respectively) or is not,
- whether an innovation item was used or not used by some end users (the symbol “+” and “-” are used respectively),
- technology readiness level (TRL) of the innovation item.

Table 1. Innovation Scorecard

BPaaS Phases	Innovation Item	Publication	Prototype	Integrated in a Tool	Used by End User	TRL
1 Design	<u>1.1 BPaaS Semantic Modeler</u>	[15] [16]	+	+	+	5
	<u>1.2 Context-Adaptive Questionnaire</u>	[26]	+	+	+	5
	<u>1.3 DMN-based Ontology Query</u>	- ¹	+	-	-	5
	<u>1.4 DMN and BPMN 2.0 Prototype</u>	-	+	+	-	8
	<u>1.5 BPaaS Design Environment Prototype</u>	[47][49]	+	+	-	5
	<u>1.6 Cloud Readiness Checker</u>	-	+	+	+	6
2 Allocation	<u>2.1 PaaS-SaaS support of Camel</u>	- ²	+	+	+ ³	5
	<u>2.2 SLA support in OWL-Q</u>	[31]	+	+ ⁴	-	5
	<u>2.3 DMN to CAMEL mapping</u>	[13]	+	-	-	5
	<u>2.4 Smart Service Discovery and Composition Tools</u>	[27] [30] [33] [32] [34]	+	-	+	5
	<u>2.5 SRL Extension of CAMEL</u>	[36]	+	+	+ ⁵	5

¹ Publication under submission

² Publication under submission

³ It is utilised in both the Cloud Provider Engine and the Allocation Tool

⁴ It is integrated in Smart Business Intelligence analysis Tool

⁵ It exploited by the Monitoring & Adaptation Engines and it is implemented in the Allocation Tool

3 Execution	<u>3.1 PaaS Orchestration</u>		[36]	+	+	+	4	
	<u>3.2 BPaaS Monitoring Engine</u>	3.2a Distributed and self-scalable Monitoring Engine	[6]	+	-	+	2	
		3.2b Cross-Layer Monitoring Engine	[52]	+	-	+	4	
		3.2c Synergic Cross-Layer Monitoring Framework	⁶	⁷	-	-	2	
	<u>3.3 BPaaS Adaptation Engine</u>	3.3a AXE Adaptation Framework	[6]	+	⁸	-	4	
		3.3b Cross-Layer Adaptation Framework	[51]	+	-	-	4	
		3.3c Synergic Cross-Layer Adaptation Framework	[36]	⁹	-	-	2	
	4 Evaluation	<u>4.1 Smart Business Intelligence analysis Tool</u>	4.1a Information Harvesting and Linking	¹⁰	+	+	-	5
			4.1b Conceptual Analytics Engine	[35]	+	+	+	
<u>4.2 Hybrid Business Dashboard</u>			[47][49]	+	+	-	5	

The innovation scorecard shows a high maturity of the innovation items. For nearly all innovations items there exist peer-reviewed publications or they are under review. Nearly all innovation items are implemented as a prototype and a majority is already integrated in a tool by the technology partners and thus ready for exploitation.

⁶ Publication under submission

⁷ Conceptually is ready, but the prototype is under development

⁸ Partially integrated in the Integration Environment, but not integrated in the full BPaaS life-cycle

⁹ Conceptually is ready, but the prototype is under development

¹⁰ Publication under submission

The analysis of the innovation scorecard validates the efficiency of the agile approach of the project with the parallelisation of research (WP3) and implementation (WP4), because the research results (i.e. innovation items) could be integrated in the second implementation cycle, which itself was challenged by the demonstration (WP5).

1.3 Structure of the document

The document is organised according to the high-level architecture (see Figure 1). There is a section for each environment, in which the latest findings and developments are described. Each section contains a subsection that focuses on the description of the innovation items, which is then followed by a lessons learned subsection.

The first subsection in each section contains a description of the final state of the innovation items, which have been adapted, enhanced or newly introduced based on the experiences and evaluation made with the first prototypes. These correspond to the innovation items of Table 1, which are underlined.

Table 2 gives a summary of the innovation items, for which the enhancements are described in this document, and the previous deliverables, in which they were originally described.

Deliverable	Innovation Item
D3.1 [18], D3.2 [14]	BPaaS Semantic Modeler (Section 2.1.1) Context-Adaptive Questionnaire for Service Selection (Section 2.1.2) DMN-based Ontology Query (Section 2.1.3) DMN and BPMN 2.0 Prototype (Section 2.1.4)
D3.3 [43], D3.4 [12]	PaaS-SaaS support of Camel (Section 3.1.1) SLA support in OWL-Q (Section 3.1.1) SRL Extension of CAMEL (Section 3.1.1) CAMEL Language (Section 3.1.1) DMN-to-CAMEL Mapping (Section 3.1.2) Smart Service Discovery and Composition (Section 3.1.3) BPaaS Orchestration (Section 4.1.1) BPaaS Monitoring Engine (Section 4.1.2) BPaaS Adaptation Engine (Section 4.1.2)
D3.5 [24], D3.6[25]	Information Harvesting and Linking (Section 5.1.1) Conceptual Analytics Engine (Section 5.1.2) Hybrid Business Dashboard (Section Fehler! Verweisquelle konnte nicht gefunden werden.)

Table 2 Deliverables affected by evolution of innovation items and tools

All the innovation items that were not subject to changes are not described again in this deliverable. These can be identified in Table 1 as the innovation items, which are not underlined.

The lessons learned subsections give an actual explanation about how the results differs from the preliminary solutions as described in previous deliverables and constitutes a critical reflection of the current solution. It additionally addresses possibilities for future research directions and development.

2 BPAAS DESIGN ENVIRONMENT

The BPaaS Design Environment is based on a traditional business process management tool - in this case ADONIS®. It supports the design of all parts of a Business Process as a Service. Such a design includes the domain-specific business layer and the cloud relevant technical layer, as well as the needed alignment between them. The business layer includes graphical models of business processes, organisations and documents, which are to be understood by business people. The IT layer consists of IT services and their orchestrations (i.e., graphical models of workflows), which reveal technical aspects of business processes and therefore are meant to be designed and understood by technical people. In order to support the alignment of business processes with IT services and workflows, functional and non-functional requirements can be assigned to parts of a business process. In contrast to business processes that are specified in terms of requirements, workflows are specified in terms of descriptions. There is a reference of IT service/workflow models to specification of functional and non-functional capabilities. Elements of the graphical models can be semantically annotated and mapped to concepts of the BPaaS Ontology [18] by semantic lifting [19]. This: (1) fosters consistency between human interpretable models and machine interpretable models, and (2) allows to reason on the machine interpretable models. The latter can be exploited in order to support a certain form of reasoning to suit the alignment task, which involves the matching of domain specific business process aspects with executable IT services/workflows in an automatic manner.

The BPaaS Design Environment comprises three updated resp. new innovation items

- the BPaaS Semantic Modeler – this innovation item addresses the need to support consistency between human and machine interpretation of BPaaS models;
- the Context-Adaptive Questionnaire – this innovation item addresses the need to perform a cloud service/workflow discovery in an intuitive and smart manner, i.e. Business-IT alignment in the Cloud
- DMN-based Ontology Query - this innovation item allows reasoning on and querying ontologies through a business-oriented interface
- DMN and BPMN 2.0 Prototype this innovation item allows to integrate DMN with BPMN 2.0 in the Cloud.

The research for the BPaaS Design Environment was used to enhance the BPaaS Design Environment Tool, which was first developed in the ADOxx¹¹ modelling environment, and then transferred to a web solution.

2.1 Innovation Items

2.1.1 BPaaS Semantic Modeler

As it is described in [16], the BPaaS Semantic Modeler includes a domain specific modelling language (DSML) as an extension of the OMG standard BPMN 2.0 [23]. Domain-specific conceptualization enhances the understanding of the targeted stakeholders [21] [22]. In our context, stakeholders are modellers such as a cloud brokers with business background. Hence, the extended modelling language enables modellers to design business processes as well as business process requirements through language constructs that are commonly understood by business people. Figure 2 shows the "Social Media Campaign" business process. Requirements are specified on a general level referring to the whole process, or on a more detailed level specifying requirements for a group of or single (process) activities. Requirements are represented in the form of a notebook (see bottom-left part of Figure 2) and map to two categories: functional and non-functional requirements. The former can be specified in two ways:

- (a) by assigning hierarchy categories from the APQC Process Classification Framework [1], and
- (b) by assigning an action and an object from a predefined taxonomy.

¹¹ <https://www.adoxx.org/>

The latter corresponds to the convention of BPMN to name activities by a verb and a noun [43]. The verb corresponds to the action and the noun to the object. Non-functional requirements are derived from the Cloud Service Level Agreement Standardization Guidelines [3] and were consolidated in workshops performed iteratively within the CloudSocket consortia. These business oriented requirements are grouped into five categories: *Performance*, *Data Security*, *Support Service*, *Payment*, and *Target Market*. In each group there are a number of attributes that reflect the business language, e.g., the performance category includes the media type (i.e., document, video, image and audio) and number of process executions per time frame, which are required for expressing non-functional requirements at the business level.. All requirements can be found in D3.2 [14].

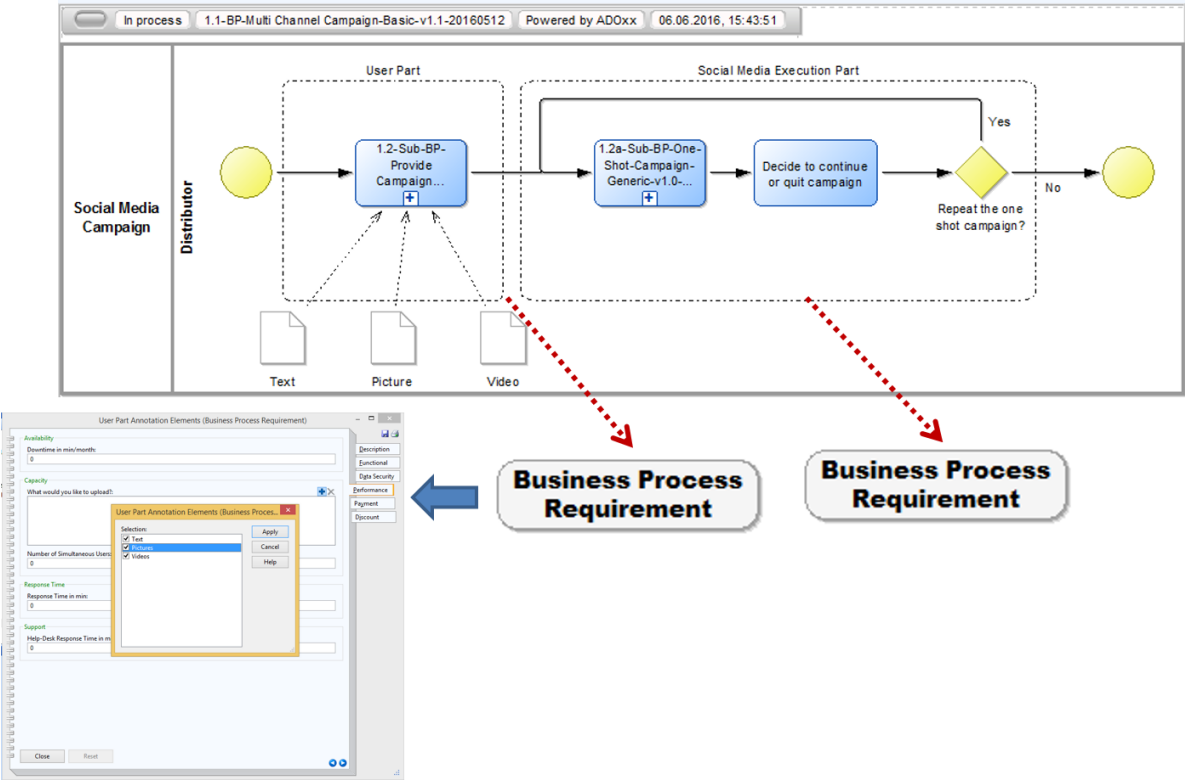


Figure 2. BPMN 2.0 extension to accommodate business process requirements

Similarly, the proposed extended modelling language supports the modelling of cloud service specifications as well as their orchestration, i.e., actual workflows. The modelling element for lanes in the BPMN 2.0 was customized to target modellers with IT background that can specify technical aspects in the cloud domain. Categories are the same as for the process requirements whereas attributes in each category change. For example, the performance category has Capacity, which spans the available data storage, simultaneous connections and service users. Figure 3 shows an excerpt from deliverable D3.2 [14], depicting the specification of cloud services and workflows on bottom, while business process requirements are specified as shown in Figure 2. Both requirements and specifications appear in the form of attributes in two notebooks, respectively.

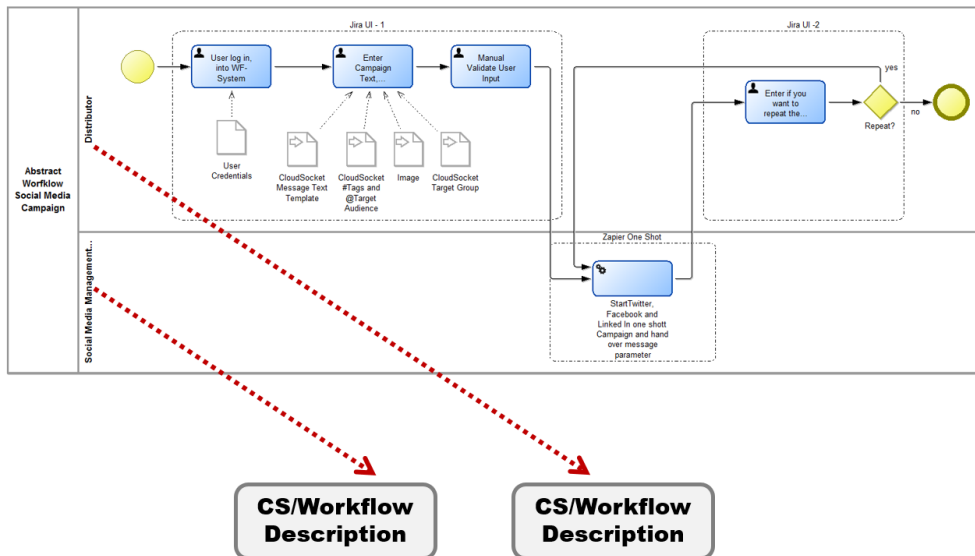


Figure 3. BPMN 2.0 extension to accommodate cloud service and workflow descriptions

Attributes can be annotated with classes and values from the BPaaS Ontology. This is called semantic lifting.

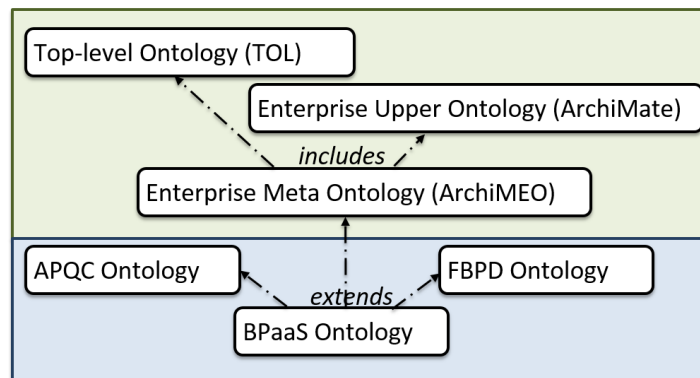


Figure 4. BPaaS Ontology

The BPaaS Ontology was first introduced in deliverable D3.1 [18], and was continuously evolved during the project. The BPaaS Ontology is part of the Semantic Modeler. The BPaaS Ontology extends two ontologies – APQC and FBPD (i.e. Functional Business Process Description). The former reflects the taxonomy of the APQC framework [2] while the latter reflects the actions and objects of the predefined taxonomy. The BPaaS Ontology is also an extension of the existing ArchiMEO ontology, which includes concepts for modelling enterprise architectures based on the ArchiMate standard [45] as well concepts for business process modelling. Hence, the ArchiMEO ontology has been extended with concepts for functional and non-functional aspects of business processes and its activities. Thus, the BPaaS Ontology extends the process requirements ontology by adding additional information when annotating the business processes and workflows.

Smart Business and IT-Cloud alignment

The BPaaS Ontology primarily enables the smart Business and IT-Cloud alignment. This smart business and IT-Cloud alignment consists of two parts:

- (1) Business-IT levels normalization (semantic rules)
- (2) Matching of BP requirements with Cloud Service specifications

The Business-IT levels normalization maps the language of the business users to IT concepts. First, it uses knowledge represented in the BPaaS Ontology to derive new facts, i.e. subsumption reasoning exploitation. For example, if a BP should run at least once a week, it subsumes also services that run on a daily basis.

Second, business-related attributes are mapped to IT-related attributes such that the most suitable workflows can be automatically retrieved for the given business process requirements. For example, while the users might prefer to require availability in maximum downtime of a service per month, the service availability of a service is typically given in percentage. As another example, we assume that a business user, while specifying the process requirements, sets the number of monthly process execution to "50" and chooses "video" as a media type to upload in the process. From the ontology the average size of videos is retrieved, which is considered as a technical aspect. It is now possible to calculate the minimum amount of storage capacity (technical aspect) that would be required to satisfy the requirements. Hence, only cloud services or workflows that have a storage capacity, which is higher than this minimum are retrieved. The calculation of the required storage capacity is performed through a semantic rule (SPARQL Construct).

Next, a semantic query (SPARQL) is used to compare business process requirements with workflow/CS descriptions.

The smart-business and IT-Cloud alignment mechanism has been implemented in a java-based prototype and described in deliverable D3.2 [14]. The prototype shows as an output the matching results, i.e., those workflows that conform to the business process requirements.

2.1.2 The Context-Adaptive Questionnaire for Service Selection

The Context-Adaptive Questionnaire aims at finding the matching cloud service(s) with the least possible number of questions. It allows specifying requirements using a domain-specific business language in a user-centric manner. The questionnaire presents a set of questions that focus first on business process functional requirements and then on non-functional requirements. Whereas questions relating to functional requirements are fixed (action, object and APQC), questions relating to non-functional requirements are displayed according to a *prioritisation algorithm*. As it is described in [26], the algorithm considers the following:

1. the user preferences in terms of categories;
2. the entropy of semantic attributes reflecting cloud service specifications, e.g., c. Namely, the higher the entropy value of an attribute is, the higher its service distinguishability degree is, and thus the higher the assigned priority of the related question becomes. This approach leads to the least possible number of questions being posed and answered, thus reducing the business service matching time.

The main idea is that the questionnaire should be applied on the whole business process first. Next, if no service can be found, the user can then move down to groups of activities (i.e., process fragments), and lookup services for each fragment. The split and search loop continues until we reach the level of activities and no services have been found for them. The top-down hierarchical service identification is shown in Figure 5; it fosters the identification of services such that they implement as many activities as possible of the business process.

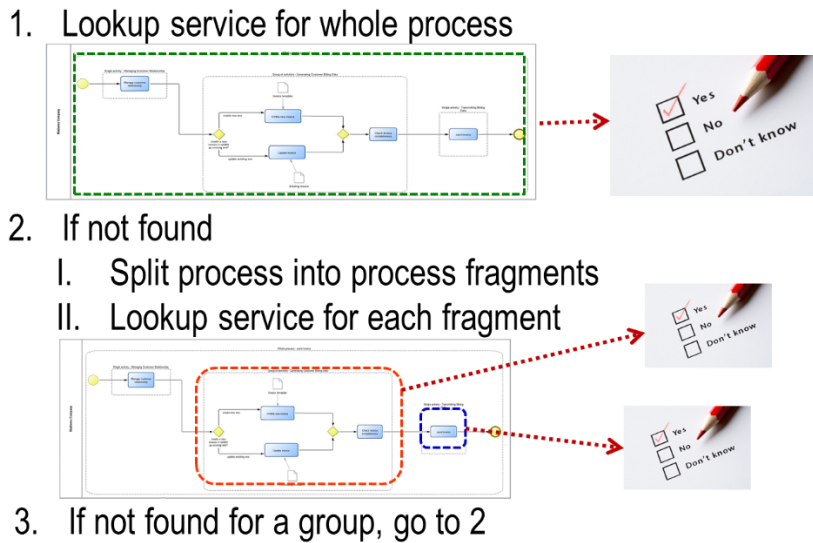


Figure 5. Hierarchical service identification

The Ontology-based Context-Adaptive Questionnaire

The Context-Adaptive Questionnaire adopts the ontology-based metamodeling approach introduced in [17]. That is, the web-based questionnaire interface is a graphical representation of concepts that exist in the metamodel, i.e., the Questionnaire ontology. The latter contains questions and answers expressed in a non-technical language and exploits the aforementioned BPaaS Ontology. In this sense, the BPaaS Ontology is extended by the Questionnaire Ontology.

The questionnaire allows specifying functional requirements in the same way as presented for the Semantic Modeler, i.e. in the form of an action-object pair and a certain APQC category. At the starting view of the questionnaire, the user can insert keywords for the object he/she is looking for, and the ontology returns the concepts matching these keywords in a list. Next, the user selects the appropriate action (e.g. manage, see Figure 6) and will continue by receiving the follow-up question about the action to be performed on that object and the respective matching APQC category.

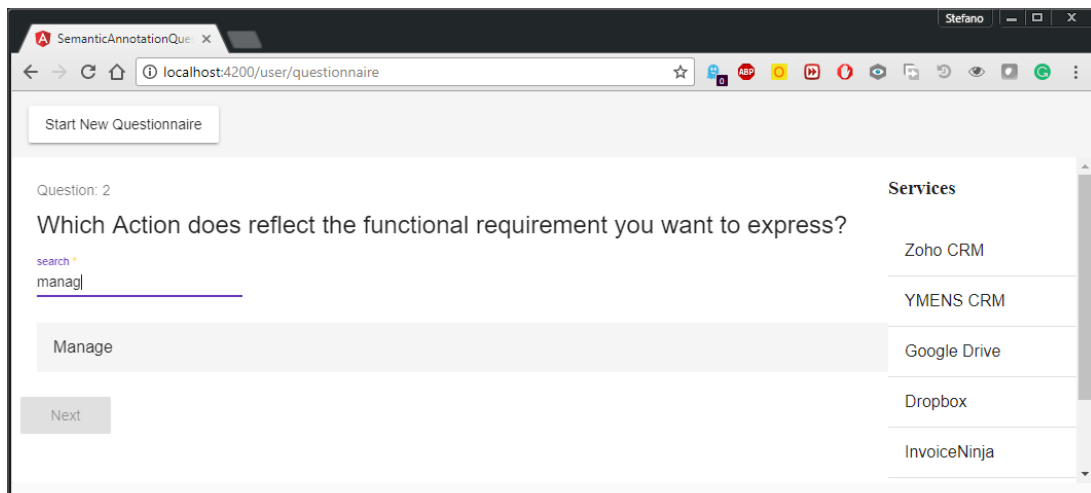


Figure 6. The action selection for the functional requirements question

Then, if the user has not identified the cloud service yet, he/she can further specify the non-functional requirements by first choosing one of the categories from the questionnaire. The non-functional requirements build on those developed in the Semantic Modeler (see section 2.1.1). Namely, they were integrated with a PhD work that consulted 46 recent research papers (i.e., from 2009 to 2018) coming from several scientific sources (i.e., IEEE, Elsevier, Google Scholar, Springer, ACM). In result, we derived the following top eight categories.

- 1) Payment
- 2) Security
- 3) Performance
- 4) Availability
- 5) Reliability
- 6) Interoperability
- 7) Support
- 8) Target Market

The Performance category, for example, includes questions like the following:

- Question: What is your preferred monthly downtime in minutes?
 - o Possible answer: 30 minutes
- Question: Should the process be executed on a daily, weekly, monthly or yearly basis?
 - o Possible answer: On a weekly basis
- Question: What is your favourite response time level?
 - o Possible answer: High, Medium or Low
- Question: How many simultaneous users should the cloud service support?
 - o Possible answer: at most 10

For each question, we have distinguished four types of answers as: (1) single-answer selection (i.e. the user selects one answers from a predefined set); (2) multi-answer selection (i.e. the user selects more answers from a predefined set); (3) search-insert; (4) value-insert. Value- and search-insert require user input. While the former enables inserting attribute values (e.g., the aforementioned downtime), the latter enables crawling predefined values from the ontology and selecting the suitable one. For instance, answers related to the first three functional requirement questions (Action, Object and APQC category) are of the search-insert type. Namely, users can insert keywords for the business process they are looking for, and the ontology returns the concepts matching these keywords. Figure 6 shows this functionality's implementation result.

Each time a question is answered, semantic rules are applied to convert implicit knowledge reflecting the business requirements into an explicit one. This prepares the ground to identify matching cloud services by applying a semantic query. For example, assume we have the following:

- Specifications from the semantic knowledge base as follows:
 - A cloud service with the execution constraint of 20 times per day.
- Requirements from the questionnaire are as follows:
 - Should the process be executed on a daily, weekly, monthly or yearly basis?
 - Answer: At least on a weekly basis.
 - How many times should the process be executed?
 - Answer: At least 10 times

Running a process at least on a weekly basis implies that can also run on a daily basis. The semantic rule, therefore, would infer the answer “On a daily basis” and insert it in the knowledge base. The semantic query then compares the derived fact with the cloud service fact related to the execution constraint. In result, the cloud service specification matches the requirement.

Accordingly, non-functional specifications have been re-laborated, too. We screened the service descriptions of four marketplaces, i.e., Ymens¹², IBM¹³, Also¹⁴ and UK digital marketplace¹⁵. As a result, Table 3 was developed, which includes the above mentioned top 8 categories and their sub-categories. In turns, predefined values were entered for most of the sub-categories (see column the most right column of Table 3).

Table 3. Non-functional Specifications

Non-Functional Specifications		
Top 8 Categories	Sub-categories	Values
Payment	Payment Plan	Customizable Plan Free of Charge Fixed Subscription Per-terabyte Per-instance Per-user Per-day Per-hour Initial Base Fee Per-Item Utility Pay-as-you-go Monthly Fee None Prepaid Annual Plan Try Free First Other Not specified
	Additional costs	yes no not specified

¹² <http://www.ymens.ro/en/frontpage>

¹³ www.bluemix.net

¹⁴ www.alsocloud.ch

¹⁵ <https://www.digitalmarketplace.service.gov.uk/g-cloud>

Security	Encryption type	AES TLS VPN IP Filtering SSL (Secure Sockets Layer) IAAS Ipsec TLS PSN SOX HIPAA FDA FIPS ISO:27001 SSH HIPAA and HITECH PCI DSS Privacy Shield Other Not specified
	Stored data location	EU-US Privacy Shield agreement locations EES European Union United Kingdom The Netherlands European Economic Area (EEA) Africa Asia China Israel Europe Austria French Germany Italy Romania Switzerland North America USA South America not defined Brazil Other
	Is there a security standard in place?	yes no not specified

	Is there an automatic password management in place?	yes no not defined
Performance	Is a performance management system in place?	yes no not specified
	Are different performance plans available?	yes no
	Response Time (in ms)	value
	Is the computing processing power scalable?	yes no
	Is the Data Storage scalable?	yes no
	Data Storage in GB	value
	Simultaneous Users	value
Availability	Availability in Percentage	value
	Access Log Availability (in Months)	value
	Access Log Retention Period (in Months)	value
	Audit Log Availability (in Months)	value
	Audit Log Retention Period (in Months)	value
Reliability	Backup Frequency	every 3 hours defined by customer monthly weekly yearly daily hourly not specified
	Backup Retention Time	up to 1 day longer than 1 year up to 1 week up to 1 month up to 1 year up to 6 month not specified
Interoperability	Data Import Format	video_mp4 pdf ppt mp3
	Data Export Format	csv Other
	Can data migration be performed independently from the provider?	yes no
	Application Programming Interface (API) Integration	yes no

Support	Service Support Responsiveness	at_most_1_working_days at_most_2_working_days at_most_3_working_days at_most_5_working_days at_most_4_working_days at_most_30_working_days at_most_1.5_hours at_most_1_hour at_most_4_hours at_most_2_hours at_most_5_hours at_most_8_hours at_most_12_hours at_most_120_hours at_most_3_hours at_most_15_minutes Not specified
	Service Support	Mon-Fri Mon-Sat Mon-Sun 24-7 7 days a week 24-5 9am-5pm Not specified
	What are the offered support channels	Phone Mail On-line Ticketing On Site Support Social Media Other
Target Market	Target Market	Business Publishers Culture/Archeology Justice sector Social Care Development Agencies Health Care Government Institutions Public Sector Social Sector Web-Developers App Developers Education No Target

Question Prioritisation Algorithm

The posing of the non-functional requirement question follows a question prioritisation algorithm. This enables identifying the matching cloud services by asking as few questions as possible. Answers to the questions, along with previous ones, are used to display the follow-up question. The algorithm considers the following:

- Dependencies between the user's preferences and non-functional attributes. Namely, all the non-functional attributes that belong to the selected categories are considered. For instance, if the user selects "Availability" and "Security", only questions that belong to these two categories will be displayed.
- Entropy expressing the distribution over the matching and not matching services. The entropy is 1 if there are equal numbers of matching and not matching services, while the entropy is 0 if all services are either matching or all services are not matching. The objective is to have an overall entropy of 0, which means that either all the remaining services are matching or there are no matching services.

A straightforward idea would be to consider the information gain of each non-functional attribute to decide on the attribute for the follow-up question. Information gain is used in decision tree learning to identify the attribute that contributes most to decision making. The formula for information gain is as follows:

$$IG = E - EV$$

where IG is the Information Gain, E is the Entropy of a classification and EV is the Expectation Value of the attribute. The information gain is the expected reduction in entropy caused by partitioning the examples according to a given attribute. This, however, requires knowledge about the expected result in advance in order to calculate the expectation value. In our case this value is unknown as we do not know the matching cloud services in advance.

Instead, we calculate the entropy over each single attribute. The attribute with the highest value is picked as we can assume that it subtracts the most from the overall entropy (remember that the goal is to achieve an entropy of 0). Subsequently, the related question is displayed to the user.

For example, if all cloud services in the repository have the same percentage of monthly availability, the entropy value for the attribute availability will be 0. As such, the question related to the availability will get the lowest priority as it won't filter out any services from the matching set. Hence, the question related to the preferred availability will not be asked.

The entropy of each attribute is calculated with

$$Entropy(attr_i) = - \sum_{j=1}^J (p_{ij} * \log_2(p_{ij}))$$

where J is the total number of attribute values and p_{ij} is the probability that a certain attribute value val_{ij} of attribute $attr_i$ appears in a specific cloud service. By considering that this probability is independent and uniform across all attribute values, then p_{ij} can be expressed as:

$$p_{ij} = \frac{[CS]_{csval_{ik}=val_{ij}}}{[CS]}$$

where the nominator denotes the number of cloud services that exhibit the respective attribute value ($csval_{ik}$ denotes the value of $attr_i$ for cloud service k) and the denominator the number of all cloud services.

The prioritisation algorithm's signature and main logic is as follows:

Input.

- 1) Already stated variables: $attr_i$; CS ; val ; $csval$.
- 2) The set of non-functional categories $C = \{Data\ Security, Payment, Performance, Service\ support, Target\ Market\}$.
- 3) Set of tuples $\langle attr_i; Q_l \rangle$ where Q is the set of questions and Q_l is a certain question where $1 \leq l \leq [Q]$. So, each tuple maps 1 attribute to 1 question.

Output. The filtered set of cloud services CS that match with the content of the questionnaire, i.e., questions and answers.

Business Logic.

1. IF the number of categories left is positive ($|C| > 0$), select a category c_n ,
ELSE exit.
2. IF c_n has a positive number of semantic attributes left, i.e., $|attr_i.s.t\ attr_i.cat = c_n| > 0$,
THEN calculate the entropy of all the selected category's attributes
ELSE remove the current category c_n from C and go to (1).
3. Select attribute $attr_i$ with highest entropy.
4. Display question Q_l that is mapped with the $attr_i$.
5. Get user answer mapping to a value val_{ij} of attribute $attr_i$.
6. Filter services in CS which do not satisfy the condition: $csval_{ik} = val_{ij}$.
7. Remove the semantic attribute $attr_i$ from the category c_n and go to (2).
8. Exit.

Interface for Cloud Service Specifications

Cloud providers can insert cloud services and related specifications in the ontology repository in an easy manner through a web-based form. This solution also follows an ontology-based metamodeling approach. Besides the smart Business-IT alignment, an ontology-based metamodeling approach supports the continuous evolution of requirements. In case new concepts (e.g., service, service specification category, question or answer) have to be added or existing ones should be changed or deleted, this can be quickly done in the ontology. Changes will then be propagated to the web-based interfaces (i.e., cloud service forms) automatically, without programming interventions.

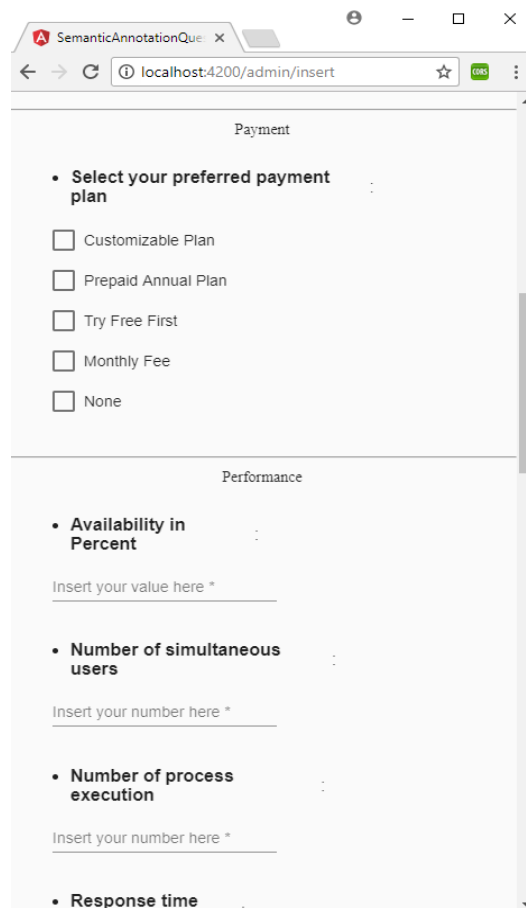


Figure 7. Interface for Cloud Service Specifications

2.1.3 DMN-based Ontology Query

The DMN-based Ontology Query enables the business user like the cloud broker to use DMN decision tables as a user-friendly interface for reasoning on ontologies. The latter is only possible through the specification of semantic rules, accordingly. This task requires ontology expertise. This innovation item aims at overcoming this challenge by allowing the broker to create DMN decision tables, in the ADOxx modelling environment. Figure 8 shows two DMN decision tables: one transforms the more business term monthly downtime in minute into the more technical term monthly availability in percentage. The second decision table infer the backup frequency granularity of cloud services from a lower (e.g. daily) to higher one (e.g. weekly, monthly and yearly). That is, if a cloud service has a daily backup frequency implies also weekly, monthly and yearly.

Availability		
U	BusinessP...	CloudServi...
1	>=0	(100 - (?val...

Backup Frequency		
R	CloudServi...	CloudServi...
1	daily	weekly
2	daily	monthly
3	daily	yearly
4	weekly	monthly
5	weekly	yearly
6	monthly	yearly

Figure 8. Decision tables reflecting semantic rules for Availability and Backup Frequency

Decision tables can then be imported in the prototype shown in Figure 9. In order to create semantic rules out of DMN decision tables, the user should: (a) load decision tables in xml format, and (b) load the ontology in .ttl format. By clicking on the dedicated button “Execute Decision Tables” the semantic rules will be created in the destination directory. As the right side of Figure 9 shows, files needed for actions (a) and (b) can also be retrieved from the web.

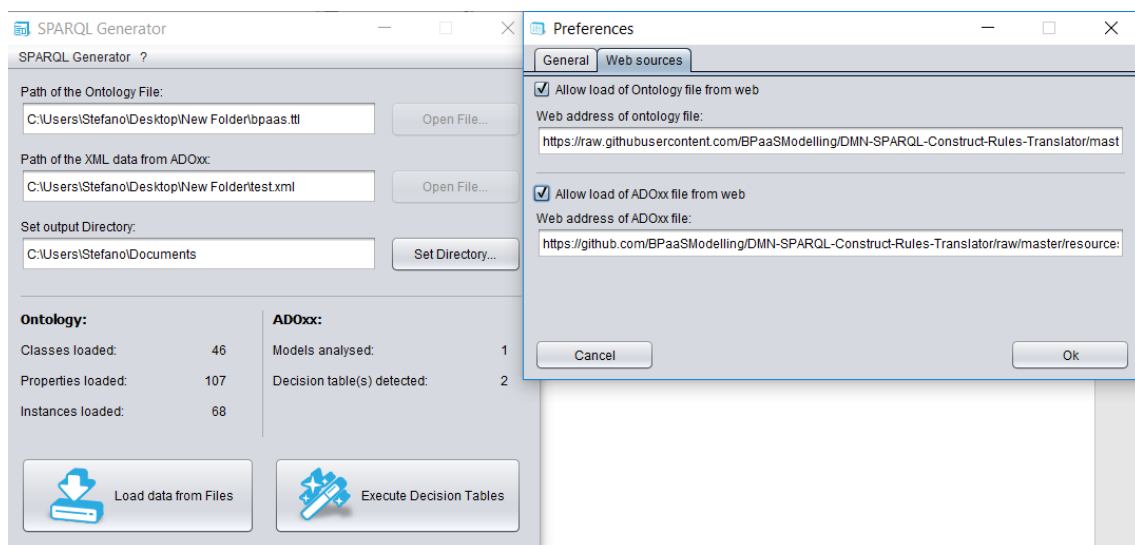


Figure 9. DMN-based Ontology Query Prototype

2.1.4 DMN and BPMN 2.0 Prototype

Initial prototype that integrates Decision Modeling Notation (DMN) into the Business Process Model Notation (BPMN). No further cloud computing or CloudSocket specific contribution, but preparation of initial exploitation item to link BPMN – for modelling business processes and workflows – with DMN – for modelling annotation and deployment decisions. The metamodel created defines a minimal subset of BPMN 2.0 that can be used to define business processes that can be linked to a decision model. The subset is selected based on practical experience of the types of concepts used for modelling such procedures. The decision model gives you the possibility to design your business decisions. The aim is to enable business users (e.g. analysts, technical developers) to comprehend the decisions that have been defined.

2.2 Lessons Learned

2.2.1 BPaaS Semantic Modeler, Context-Adaptive Questionnaire and DMN-based Query Ontology

The Semantic Modeler described in deliverable D3.2 [14] was developed in the first year and subsequently validated with respect to two use cases developed within the consortia, i.e., the Christmas greetings BPaaS and the send invoice BPaaS. Next, we provided access of the prototype to the CloudSocket partners to collect feedback.

The main drawbacks were the following:

- 1) Too much effort in specifying the business process requirements.
- 2) Difficult for non-ontology experts to insert cloud service specifications in the triple store.
- 3) Matching only considered the whole workflow. This prevents identifying cloud services that would implement group of activities or single activities.
- 4) Difficult for non-ontology experts to formulate the semantic rules and semantic queries.

In the second year of the project, we aimed at improving the Semantic Modeler by addressing the above listed drawbacks.

We addressed the first drawback by developing the innovation item “Context-Adaptive Questionnaire” introduced in sub-section 2.1.2. Namely, from the business perspective, process requirements have been expressed in the form of a questionnaire. Answering meaningful and business-related questions made the approach more (business) user centric.

Additional evaluation was done with external partners. In collaboration with a project at FHNW, we analysed the attributes for the service specification and the context-adaptive questionnaire. In particular, in cooperation with the CLIMB research project [10] and with hotelleriesuisse, the Swiss Hotel Association, we validated the attributes for the cloud service specification and adapted it to the categories listed in section 2.1.2.

While the context-adaptive questionnaire allows for easier business requirements specification, adding service specifications required ontology expertise. This was recognized as a too difficult task for non-ontology experts. Therefore, the web interface for cloud service specifications (see

Figure 7) was developed to address this challenge. The web interface allows translating the entered specifications into an ontology. In turns, the information entered for a cloud service can be used in the context-adaptive questionnaire for service matching/identification purposes. Additionally, the creation of this web-service led to a further improvement on the way semantic rules are executed. Namely, instead of executing semantic rules for each answered question, they are executed only once when inserting the cloud service specifications in the form of an ontology. This allows to keep low the computing power that is needed to execute semantic rules.

To address the third drawback, we proposed the hierarchical service identification. As shown in Figure 5, the questionnaire is applied on the whole business process first. Next, if no service can be found, the questionnaire is successively applied on groups of activities (i.e. process fragments) and, if necessary, until single activities.

In order to perform the Business-to-IT alignment, semantic rules need to be applied before a semantic query can be performed. Namely, semantic rules are applied to (1) deriving new facts, e.g., if a BP should run at least once a week implies it could run on a daily basis too; (2) transforming the business to the IT view, e.g., from the non-functional attribute of downtime to the one of availability. Then, business process requirements can be compared with cloud service specifications. For non-ontology experts, it is difficult to express those semantic rules and semantic queries as deep know-how about ontology languages and SPARQL is required. Therefore, we adapted

the DMN modelling language such that business users can model semantic rules and queries in DMN decision tables [38]. DMN is intended as a decision modelling language suitable for communication with business users. The DMN-based Ontology Query Prototype (introduced in subsection 3.1.1.3) can parse: (a) the created DMN decision tables and (b) the ontology against which the rules are performed. In result, semantic rules are created and ready to be inserted in the Context-adaptive Questionnaire prototype.

2.2.2 DMN and BPMN 2.0 Prototype

The definition of this prototype started from the results of the LearnPAd European Project ¹⁶and was the first effort required to integrate DMN in the commercial product Adonis¹⁷.

In this prototype, we have seen that a graphical improvement for the DMN decision table was required. In particular is better to visualize the whole table as graphic image in order to avoid to open the element notebook all the time. This will not match the architecture of the ADOxx environment and this change had required lot of effort. This has been implemented to become a Release Candidate in the commercial version of Adonis.

¹⁶ <http://www.learnpad.eu/>

¹⁷ <https://us.boc-group.com/adonis/>

3 BPAAS ALLOCATION ENVIRONMENT

The goal of the BPaaS Allocation Environment is to make the executable BPaaS workflows, as handed over by the BPaaS Design Environment, deployable in the cloud. This is mainly achieved in two ways: (a) the discovery and the selection of SaaS services which can realise the functionality of the BPaaS workflows tasks; (b) the discovery and selection of IaaS services which can support internal SaaS components of the BPaaS workflow. The main outcome of the BPaaS allocation phase is the production of the BPaaS bundle, a container that includes the following information: (i) the content of the respective BPaaS design package (BP, workflow, DMN & KPI models); (ii) a CAMEL model spanning both deployment, monitoring and adaptation information; (iii) business-oriented information including pricing and Service Level Agreement (SLA).

The main research conducted in the context of the project and this environment focused on providing automatic support for the two aforementioned functionalities that are required to make BPaaS executable workflows deployable in the Cloud. This automatic support has come with the delivery of 3 main innovation items:

- (a) extensions to the CAMEL language focusing on specifying the whole allocation dependencies within the BPaaS hierarchy as well as the modelling and exploitation of PaaS services;
- (b) the smart service discovery and composition tool focusing on providing automatic support to the discovery and selection of cloud services;
- (c) the DMN-to-CAMEL mapper focusing on supporting the mapping of DMN rules to the respective allocation/deployment specification in the CAMEL language.

Independently of the conducted research, a sophisticated BPaaS allocation tool was also developed which does support the manual allocation of the BPaaS workflows and acts as a proprietary editor for the specification of the required BPaaS bundle. Such a tool already supports the CAMEL extensions while it can be easily adapted in order to support the integration of the innovation items in the pursuit of automating as much as possible the BPaaS allocation phase for the BPaaS broker.

In the sequel, we focus on the analysis of both the innovation items and the allocation tool in separate sub-sections. This analysis also involves knowledge derived from the lessons learned during the development of all these artefacts/assets.

3.1 Innovation Items

3.1.1 CAMEL Language

Initially, the CAMEL language supported the specification of deployment information mapping to the allocation of SaaS components at the IaaS level. However, by considering that PaaS services also enable the allocation of software components and come with the benefits of:

- (a) rapid deployment and
- (b) the capability to allocate the application component without providing deployment details at the infrastructure level,

it was decided that CAMEL should be extended (see D3.3 [43] and D3.4 [12]) in order to allow the specification of PaaS requirements and capabilities and thus the exploitation of this information to support the deployment of application components.

The specification of PaaS requirements involved the posing of certain constraints over the characteristics of the needed environment to deploy the application components as well as of the PaaS provider along with restrictions over the needed infrastructure, which were already covered by CAMEL. Symmetrically, the specification of PaaS

capabilities involved the modelling of all the possible offerings over the characteristics of such an environment. The specification of the actual allocation, both happening at the type and instance level, involves the hosting of application components by PaaS nodes on which the PaaS requirements are posed as well as the hosting of application component instances by PaaS instances, which satisfy the requirements posed at the type level.

By considering that a BPaaS hierarchy can involve also an additional level, the SaaS one, the next CAMEL extension (see D3.3 [43] and D3.4 [12]) focused on the specification of two kinds of SaaS services:

- (a) internal SaaS services which map to a certain kind of internal (application/BPaaS) components;
- (b) external SaaS services which map to SaaS services provided by SaaS providers.

At the type level, for both types of services the information provided is more or less the same and involves mainly the specification of the IDs of the tasks of the BPaaS workflow, which are realised by the SaaS service. However, the way allocation is specified differs. As internal SaaS services are internal application components, they can be hosted by IaaS or PaaS nodes. On the other hand, an external SaaS service is obviously already hosted but also out of control of the BPaaS management system. Similarly, at the instance level, an instance of an internal SaaS service is hosted by an instance of an IaaS or PaaS service. On the other hand, for an instance of an external SaaS service we only specify information which mainly concerns the actual endpoint and location of this instance, by considering that one SaaS service might be offered in different cloud locations.

In contrast to the next two innovation items, the extensions performed on the CAMEL language have been already integrated in the final CloudSocket platform implementation. In particular, the extended CAMEL language is fully supported by the BPaaS allocation tool (see Section **Fehler! Verweisquelle konnte nicht gefunden werden.**) while it is also exploited by most of the components of the BPaaS Execution Environment.

3.1.2 DMN-to-CAMEL-Mapper

As the benefits of cloud computing come along with an additional technical depth, Model-Driven Engineering (MDE) tries to reduce this complexity by providing domain specific languages (DSL) for the cloud computing domain, such as Tosca [38] or CAMEL [41].

These DSLs ease the cloud adoption by enabling to specify a cloud application deployment model on a higher level, abstracting away from quite technical specificities, which can be executed by cloud orchestration tools (COTs), such as Cloudiator [3]. Still, the specification of a deployment model requires a certain degree of technical knowledge and the deployment model is static in nature. Current modelling approaches do not reflect the dynamics in changing business requirements that impact an implemented deployment model at run-time. As shown in the first lane of Figure 10, any requirement change leads to the re-modelling of the deployment model, which is error-prone and cost-intensive.

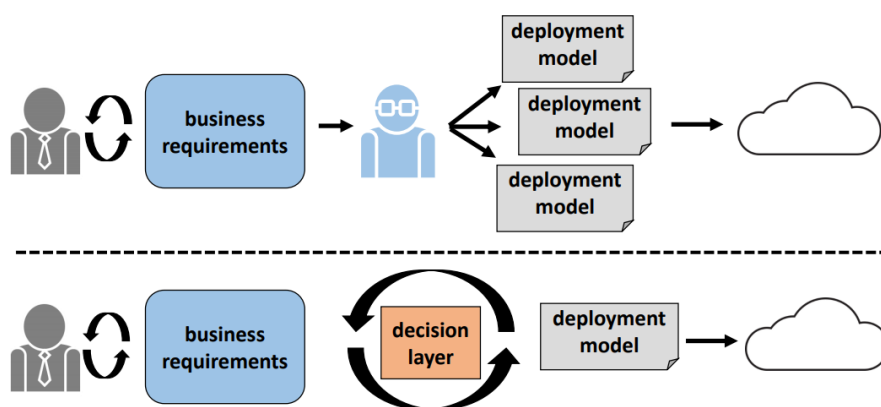


Figure 10 Dynamic deployment modelling.

In order to enable support for (i) dynamicity and (ii) reusability of cloud application models, CloudSocket introduces a simple decision layer on top of the DSL modelling, enabling the transformation of business requirements into a technical deployment model at both design-time and run-time, as shown in the second lane of Figure 10. In the following, we present the technical realisation of this decision layer in CloudSocket, namely the DMN-to-CAMEL-Mapper, while a more detailed study of the dynamicity and reusability challenges in cloud modelling can be found in [13].

The DMN-to-CAMEL-Mapper prototype is based on the Decision Model and Notation (DMN) [38] as decision layer and CAMEL [41] as the cloud modelling DSL. The DMN standard provides a human-readable common notation for modelling and automating decisions. We have chosen decision tables to represent decisions as these are well known to business experts. An example of a decision is shown in Table 4.

Hit Policy	Input		Output	
	Privacy Level <String>	Provider <String>	VM Image <String>	Region <String>
C				
1	low	Provider X	Image X	US
2	low	Provider Y	Image Y	Europe
...

Table 4 Image and Region Decision Table

A decision table consists of three column types: (i) a hit policy, (ii) an input variable set, and (iii) an output variable set. The hit policy defines the selection over overlapping decisions with policies like {U}nique, i.e., only a single decision is applicable in a case or {C}ollect, i.e., several decisions can be selected. Each input variable can potentially map to a respective output variable of a sub-decision table. Hence, there is a possible cascade of decisions leading to hierarchical decision tables. Any decision table is associated with a business knowledge model (BKM) defining the decision logic, i.e., the mapping between the input and output parameters. DMN is chosen as it is an impact-gaining standard and it is already well adopted on the business level.

We propose a realisation based on a hierarchical set of decision tables to rearrange CAMEL model artefacts as described in [13]. Both languages, CAMEL and DMN, are integrated into the meta-modelling platform ADOxx¹⁸, providing a modelling tool for dynamically generating CAMEL models via DMN.

3.1.3 Smart Service Discovery and Composition

This innovation item maps to the production of a service-based prototype tool, which supports both the semantic discovery as well as the selection of cloud services. Such a tool can be really exploited by a graphical BPaaS allocation tool, like the one developed in CloudSocket, in order to provide automatic support to the selection of the right cloud services that best satisfy the requirements of the broker.

Concerning service discovery, the tool includes a configurable sub-system [32] that supports both the semantic functional and non-functional discovery of services. Configurability mainly comes with the capabilities to: (a) select the right non-functional service matchmaking algorithm from those that have been developed by FORTH; (b) select the way functional and non-functional service matchmaking can be performed in order to produce a final and uniform service matchmaking outcome.

¹⁸ <https://www.adoxx.org/>

The first capability has been mapped to extensive research work which has been conducted in order to find both a smart way via which service advertisements can be organised as well as how they can be matched with user requests. As far as the matchmaking technique is concerned, two main directions were followed:

- (a) use constraint programming techniques in order to support the actual matchmaking between service requests and advertisements;
- (b) use ontology reasoning to support this.

For the first direction, our previous work [29] has been mainly considered which led to the development of 4 non-functional matchmaking algorithms that map to the mixed category of non-functional service matchmaking approaches [29]. In that category, first any non-functional service specification is described via an ontology language, OWL-Q [28] in our case, and then all specifications are aligned based on their non-functional terms, mainly non-functional metrics (e.g., *raw response time*). The aligned specifications are then matched based on certain matching metrics and especially the conformance one [4] (i.e., indicating that the solution space of the service advertisement should be included in the solution space of the service request). For the second direction, ontology-based subsumption [33] was employed to support the matchmaking by considering that all non-functional specifications are described by a certain ontology language. This maps to the ontology-based approach category [29] in the state-of-the-art. The proposed approach by FORTH [33] advanced the state-of-the-art in that category by correcting some issues with existing approaches as well as speeding up the non-functional matchmaking process.

Such a speed up was also researched by considering the way the service advertisement space can be smartly organised. This led to two novel smart ways for this organisation:

- (a) Exploitation of the *subsumes* relationship between non-functional specifications: here an hierarchical organisation of the advertisement space is constructed by exploiting this relationship where the parent node in the hierarchy subsumes not only its children but all its descendants. Such an organisation enables to immediately find all advertisements that are subsumed by a request if that request subsumes their root ancestor node;
- (b) Exploitation of the *subsumedBy* relationship: the advertisement space here is organised according to the opposite direction. The main rationale is that when the percentage of matched offers is low, it is better to organise the advertisement space such that the non-matching of a root node leads to not requiring the visiting and matching of its descendant nodes.

Concerning the way functional and non-functional matchmaking can be performed, 4 possible ways [32] have been explored in our research and can be configured to function in the developed tool:

- (a) *sequential*: here first it makes sense to execute functional service matchmaking and, in case it succeeds (results are produced), then also non-functional matchmaking;
- (b) *parallel*: the two aspect-specific matchmaking processes are executed in parallel and then their results are joined/concatenated;
- (c) *subsumes*: the service advertisement space is organised according to functional and non-functional subsumption and matchmaking is performed as explained in previous paragraph;
- (d) *subsumedBy*: same as in previous paragraph where organisation of the advertisement space is performed via an opposite relation than *subsumes*.

Experiments have shown [32] that the parallel combination of the two aspect-specific matchmaking processes leads to the best possible performance.

Service selection is another major issue in BPaaS workflow allocation as it involves various levels, which depend on each other. For instance, it is easy to derive that the selection of services at the IaaS level has an influence on

the quality of the internal BPaaS service components, which then has an impact on the selection of services at the SaaS level. In this respect, we have developed a cross-level service selection algorithm [30], which does consider such dependencies by also exploiting our semantic service matchmaking sub-system in order to accurately reason over semantic service offerings. This algorithm relies on the mapping of the BPaaS workflow, its main functional and non-functional technical requirements and the respective service advertisement space into a constraint optimisation problem, which is solved in order to find the best possible (allocation) solution that optimally satisfies the requirements posed. The main features of this algorithm are the following:

- (a) it is able to produce a solution even when the user requirements are over-constrained;
- (b) it is able to handle non-linear constraints and optimisation objectives;
- (c) it is able to handle any kind of variable (integer, real, set-based) (which could be mapped to a non-functional term like a metric);
- (d) it can take into consideration the mapping of dependencies between levels via the use of respective functions that derive the non-functional capabilities of a service (e.g., internal SaaS component) at a higher-level from the non-functional capabilities of another service at the lower-level that supports the former service (e.g., IaaS service that hosts the internal SaaS component);
- (e) dependencies even within the same level can also be covered via the use of functions;
- (f) there is an ability to assign parts of the problem to best deployment knowledge which is derived via a rule-based approach [27] over the execution history of the application/BPaaS.

In this way, the solution space is greatly reduced and the solution time is quite accelerated.

3.2 Lessons Learned

3.2.1 CAMEL Lessons Learned

CAMEL is constantly maintained and extended due to its exploitation in various currently running projects. Through its lifetime, CAMEL has evolved from a very simple language for specifying distributed multi-cloud applications in a provider agnostic way to an extremely powerful DSL that captures huge parts of all aspects of cloud applications, cloud providers, and cloud infrastructures. A distinguishing factor for CAMEL is the extensive coverage of multiple domains, but also of the interdependencies between these different domains. This enables consistency checking already during development time (comparable to compile errors when programming) in contrast to run-time checking (comparable to run-time errors) as used by all competitors.

During the process of working with CAMEL, the extensive domain coverage addressed by CAMEL has also unveiled drawbacks. In particular, users of CAMEL should be able to understand all its concepts covering all different domains related to cross-cloud application/BPaaS management. This requires first to be better acquainted with this language before moving ahead to start specifying its models. Fortunately, CAMEL comes with an extensive documentation. This documentation has also been recently complemented with a table-based document¹⁹ which explains the semantics of all CAMEL concepts. However, we do understand that: (a) it takes some time to be acquainted with this language; (b) it might be difficult to use CAMEL even if the semantics of all its elements is clarified. To this end, it is planned to produce a set of tutorials for CAMEL in order to better explain its usage. Such tutorials could, for instance, exemplify what has to be specified in order to support the deployment and adaptive provisioning of simple cloud applications, cross-clouds ones as well as BPaaSes. Please note that the domains covered by CAMEL map to the sphere of expertise of devops users, which are the main target users of CAMEL. In this respect, once this extensive material about CAMEL becomes available, then this kind of users will be able to fully exploit it.

¹⁹ http://camel-dsl.org/wp-content/uploads/CAMEL_Semantics_v1.1.pdf
Copyright © 2017 FHNW and other members of the CloudSocket Consortium
www.cloudsocket.eu

Another issue with CAMEL that needs to be appropriately handled comes with respect to its extensions. In particular, different projects are currently working on CAMEL and have developed their own extensions on this language. In this respect, there should be a certain mechanism in place which should enable the integration of all those extensions that increase the added-value of the language. This mechanism should enable integrating the extension by guaranteeing that: (a) it follows the same modelling style/patterns and a more or less similar level of specification detail; (b) it is accompanied with appropriate OCL²⁰ rules which cover the semantics of the respective domain as well as cater for the cross-model consistency of the different kinds of sub-models that can be specified with CAMEL. Such mechanism should also include the update of CAMEL documentation material once the respective extension is integrated. Finally, it should trigger the updating of the various kinds of CAMEL editors that can be exploited by the devops users in order to edit CAMEL models, such as the Allocation Tool developed in the context of this project.

CAMEL has been developed by humans so it is not a perfect language. In this respect, there is a real need to thoroughly evaluate the coverage and usability of this language. Coverage can be assessed through the use of domain experts which can inspect which parts of CAMEL cover well a certain domain and which need further improvement. On the other hand, a user study over CAMEL could unveil possible improvements in the way information is modelled focusing mainly on enhancing the user experience as well as the rapid specification of CAMEL models.

While CAMEL is already extensive enough, we believe that it could be further enhanced through the coverage of additional domains spanning the following aspects: (a) specification of SLAs: CAMEL is currently able only to specify service level objectives (SLOs) in the form of requirements posed by the broker/user. However, it should be able to be extended in order to fully specify SLAs as such SLAs can also drive the monitoring and adaptation of a BPaaS according to the mutual agreement reached between the BPaaS provider and requester. What should be covered in the modelling of SLAs has been indicated in [31] which could be considered as a guideline on how to perform this extension; (b) CAMEL currently supports the specification of medium-in-expressivity mathematical expressions which can be exploited for the definition of metric formulas. We believe that this should be modified in order to have the best possible expressivity. This should consider incorporating additional mathematical operators apart from those already supported as well as conditional/piece-wise linear mathematical functions. The latter could really assist in the definition of the utility of the values of a certain metric that could then drive the optimal selection of cloud services.

3.2.2 DMN-to-CAMEL-Mapper Lessons Learned

The aim of the DMN-to-CAMEL-Mapper prototype was to reduce the technical complexity of the software component allocation by mapping high-level business requirements to the low-level cloud-specific description. We used completely separated tools to (i) create, and (ii) evaluate the DMN tables. We found that it is useful to let the consultants use the tools they are accustomed to exploit (e.g., ADOxx or Camunda DMN editor), but what is currently missing is to integrate the REST server of the DMN-to-CAMEL-Mapper into such tools to not move from one environment to another. What is also still open is an appropriate format and repository for existing CAMEL artefacts and respective DMN tables. In this respect, it might be worth to extend the XSD (XML Schema Definition) for the DMN editors to allow for meta-data with correlation to CAMEL fragments.

3.2.3 Smart Service Discovery and Composition Tool Lessons Learned

Recently, we have proposed a certain approach [26], which is able to also consider the message compatibility between services during service selection. This kind of compatibility enables the production of allocation solutions, which are operable, meaning that the service-based workflow realised by them can properly function. This approach needs now to be fully implemented in order to really provide added-value to the service selection algorithm. Apart

²⁰ www.omg.org/spec/OCL/About-OCL/

from this, our work will focus on how to derive the functions that connect services/components at both the same or different levels. Parametric model learning [50] could be one of the possible techniques that could be used for this purpose. The derivation of best knowledge should also be optimised through the consideration of semantic relationships between the different components of a certain BPaaS hierarchy. This would enable the suggestion of solely accurate best deployments for BPaaS components which is a kind of knowledge that is exploited for further improving the service selection time. Finally, while we have produced a demonstration that shows the main benefits of the proposed tool via the use of a certain simplified user interface (UI), we plan to either develop a more sophisticated one or better to integrate this tool with the BPaaS Allocation tool of the CloudSocket project. The latter would really enable the enrichment of the respective tool with an add-on capability, which would enhance its automation and added-value, thus making it capable to automatically produce allocation solutions as well as better enable the user to browse the solution space with the potential modification of the respective requirements posed on the fly and the respective subsequent visualisation of the service selection result. This latter kind of interaction would really enable the broker to find the best possible configuration that can really make him/her profitable and optimise the respective utility expected from the supply of the corresponding BPaaS service.

There is complementarity between the smart service discovery and composition tool and the DMN-to-CAMEL transformer. In particular, the former can enable to produce a certain allocation solution while the latter can support the transformation of that solution to an allocation specification defined via CAMEL. As such, this cooperation facilitates the broker who needs currently to manually check the solution and integrate its various parts into a whole CAMEL agglomeration. While this kind of integration requires significant effort, it can also lead to the production of wrong CAMEL definitions as humans are usually error-prone sources of information. However, a more automated approach could enable producing correct overall CAMEL definitions, provided that the individual CAMEL models pertaining to the description of a BPaaS component are accurate. It would also be interesting to also include a deployment testing facility as an extra add-on to enable to test the respective overall CAMEL models produced for the whole BPaaS in order to support the respective troubleshooting and adjustment of these models for guaranteeing the proper deployment of the BPaaS.

The semantic service discovery and selection tool relies on the existence of semantic descriptions of (cloud) services. Such descriptions are not currently available for the majority of the services available in the market. In this respect, we foresee the following directions for realising a semantic service repository: (a) use a form-based approach like the one mentioned in the previous chapter to allow cloud providers to specify the description of their services in a user-intuitive manner which is then transformed to a semantic form and stored in a semantic service repository. Such an approach requires providing the right incentives to cloud providers for supplying the information needed; (b) in case the cloud providers are unwilling to provide such information, it can then be derived through a crawling approach: known marketplaces and cloud service offering pages can be crawled in order to acquire and then semantically enrich (e.g., via (semi-)automatic annotation algorithms) the information collected in order to store it in the semantic repository.

Some of the above directions would really create further added-value, but it was not possible to deal with them under the limited resources under which a certain project operates. However, we believe that the supply of the current innovation items will really enable the community (academic, industrial) to utilise, exploit and possibly expand them having the aforementioned directions as guidelines. This should definitely then make a nice sustainability plan for them catering also for the respective evolution.

4 BPAAS EXECUTION ENVIRONMENT

With the BPaaS Marketplace as a direct interface to the customer, the BPaaS Execution Environment deploys and executes a BPaaS bundle after it was purchased. Thus, this environment actually takes care of:

- (a) deploying the BPaaS according to the deployment plan included in the bundle,
- (b) deploying the monitoring infrastructure to be used for monitoring the BPaaS and
- (c) importing the respective executable workflow model from the BPaaS bundle into a workflow engine in order to enable its execution by the customer that has purchased it.

Another goal of this environment is to support the monitoring and evaluation of the BPaaS according to the KPIs and SLOs that have been defined for it. In case of a violation of an SLO, particular adaptation plans are executed, which are triggered via the adaptation rules that have been already defined in the BPaaS bundle.

Finally, the BPaaS Execution Environment exposes a set of APIs which enable other CloudSocket environments or BPaaS customers to:

- (a) deploy a BPaaS (Marketplace),
- (b) create, execute and manage instances of the BPaaS workflow (BPaaS Customers) and
- (c) produce as well as support the retrieval of BPaaS monitoring and assessment results for later evaluation purposes in the BPaaS Evaluation Environment.

4.1 Innovation Items

4.1.1 BPaaS Orchestration

For the cloud orchestration, we not only investigate the capabilities of provider-independence on the IaaS layer, but we integrate the same capability for the PaaS layer. And even more, we target an orchestration across those layers. The work done here is divided in two aspects:

- (i) specification, and
- (ii) execution of the BPaaS bundle.

For the *specification*, we extended CAMEL to allow the description of PaaS-based application components with the following focus: (a) description of requirements on PaaS services, (b) description of PaaS types and instances mapping to certain PaaS capabilities, and (c) capabilities to configure the lifecycle of a component via a PaaS API. The respective model enhancements can be found in deliverables D3.3 [43] and D3.4 [12].

On the *execution* side, we enhanced the so-called PaaS Unified Layer (PUL). The PUL provides an API that abstracts the different PaaS providers to simple management operations, which are employed via a REST API, as well as offers a Java-based client. The compatibility to multiple PaaS providers was achieved as an enhancement to PUL.

The PUL has been integrated into Clodiator [3]. The Clodiator framework serves as Cloud Orchestration Tool and empowers the Cloud Provider Engine. As seen in Figure 11, it consists of Colosseum, which offers a RESTful interface to its model, which makes use of registries, an application repository, and workers that handle the internal jobs, such as application deployment or resource provisioning.

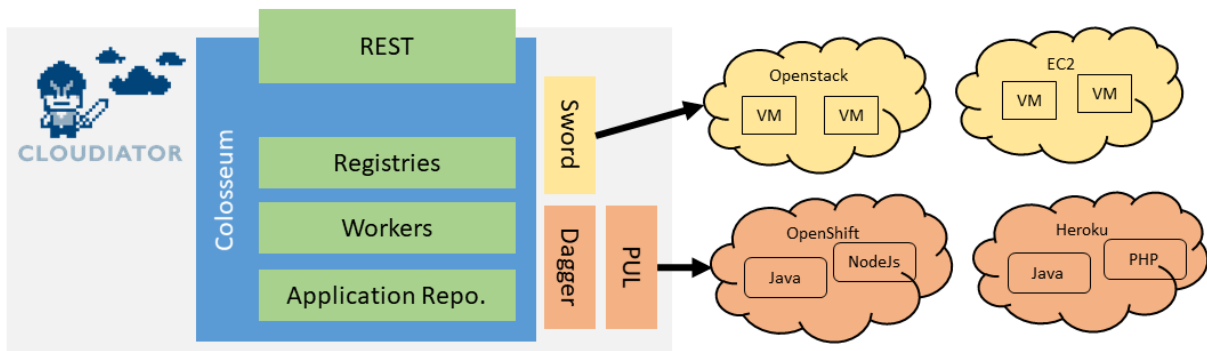


Figure 11 High-level Cloudiator architecture.

Further, Cloudiator provides abstraction layers for the Cloud providers on different levels: (i) Sword abstracts the IaaS Clouds, and (ii) Dagger abstracts the PaaS Clouds by utilising PUL.

Therefore, Cloudiator is able to handle PaaS- and IaaS-based services simultaneously and interchangeably. The management of services was designed independently from their Cloud level. All features that map to the origin IaaS orchestration were also mapped to the PaaS layer. For example, the monitoring system [6] was in turn unified to work across Cloud providers and levels. Also the application description was extended during the CloudSocket project [7].

4.1.2 Cross-layer & Cross-Level Adaptation Management

Identical to the aforementioned Cloud orchestration efforts, the work concerning cross-layer adaptation management was split into (i) specification, and (ii) execution. The sub-model SRL (Scalability Rule Language) [23] of CAMEL was extended in its scope to:

- *cover additional cloud levels*: this means that when defining behavioral rules in CAMEL, the user can now define additional adaptation actions, such as migration, service replacement, cloud burst, or task modifications, in addition to the well-known scaling actions, thus covering additional levels of abstraction.
- *cover adaptation workflows*: this means that the consequent part of adaptation rules can now express more advanced forms of adaptation strategies in the form of adaptation action workflows. For instance, this means that we could specify an adaptation workflow which first migrates an internal BPaaS component to a new VM and then updates the BPaaS workflow to change the service endpoint of that component.

Through this extension, then CAMEL, by also being capable of covering the detection of event patterns, enables the specification of sophisticated adaptation rules which can more completely address both simple and quite advanced adaptation scenarios. A more detailed description for this CAMEL extension can be found in [36].

A BPaaS system involves multiple levels of abstraction, which need to be handled along with their dependencies. In this way, in order to better deal with failures or bottlenecks during BPaaS provisioning, there is a need to both monitor and adapt a BPaaS across all possible levels. However, by checking the current state-of-the-art, we can really see some major deficiencies: (a) not all levels are covered; (b) when catering for more than one level, only level-specific monitoring and adaptation mechanisms are incorporated. This leads to the independent detection and addressing of a certain, complicated situation which can lead to a vicious adaptation cycle [51] where one adaptation mechanism at one level diminishes the results of another mechanism in another level; (c) monitoring and adaptation work for the Cloud has been quite limited.

In this respect, we have opted in the project for the development of a cross-level BPaaS monitoring and adaptation framework (see D3.4 [12]), which capitalises over existing work that has been already conducted by FORTH [51]

and UULM [6]. This framework is actually capable of exploiting the adaptation rules specified in CAMEL in order to “execute” them, i.e., be able to monitor the BPaaS in order to generate events that map to event patterns whose detection can then be used in order to enact the adaptation workflows specified in these rules.

The cross-level BPaaS monitoring part of the framework (see Figure 12) relies on a publish-subscribe mechanism that enables the propagation of monitoring information from one level to another as well as the reporting of that information to an event processing engine catering for the generation of events that lead to the execution of adaptation actions/strategies/workflows. It also relies on the use of the CAMEL language, which enables the specification of precise metric formulas that support the coverage of measurability gaps through the definition of metric (computation) hierarchies. Such formulas can also act as a guide to which metric components need to be collected from the same or lower abstraction level in order to support the computation of a high-level metric. Each level-specific portion of the monitoring sub-framework maps to the work of one from the two aforementioned partners.

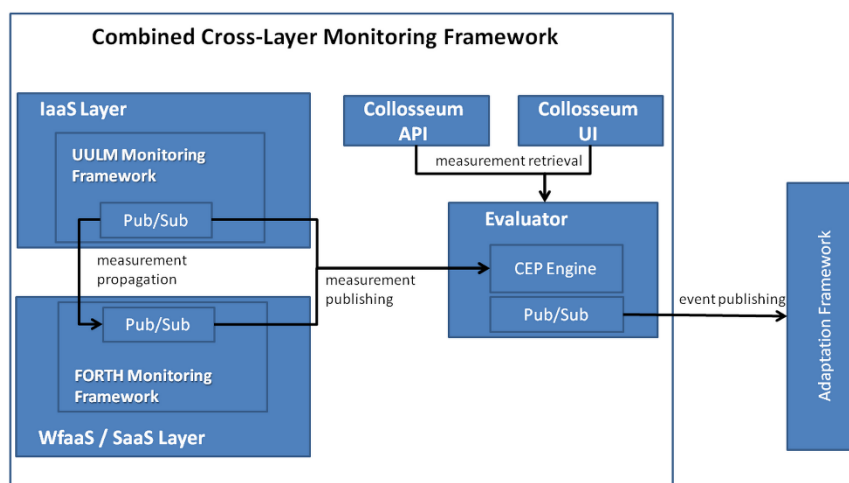


Figure 12: The combined cross-layer BPaaS monitoring framework

The work from FORTH covers the SaaS (composition) and workflow levels while the work from UULM covers the infrastructure and platform levels. Internally in each level-specific portion, basic building blocks are available and common to both monitoring approaches exploited which include:

- a time series database (TSDB), which stores the measurement information,
- an aggregator which takes care of aggregating the measurements produced at a specific level, and
- the respective monitoring sensors which populate the content of the TSDB.

The event processing component is a Complex Event Processing (CEP) engine which is able to retrieve all the relevant measurements from the different levels and then detect the event patterns that lead to the triggering of adaptation rules. Such a triggering is enabled again via a publish-subscribe mechanism which allows a certain component, like an adaptation engine/framework to listen to the event patterns of interest so as to subsequently enact the appropriate adaptation rules. This logical architecture can have multiple realisations at the physical level enabling to cover the measurement of both cross-cloud and single-cloud metrics as well as the production of the respective event patterns. This monitoring framework is already integrated with the SLA engine, thus enabling the follow up of the status of SLAs while a BPaaS is running.

The rationale for the realisation of the adaptation functionality was quite similar. We have relied on the existing adaptation capabilities exhibited by the work of FORTH [51] and UULM [6], which are offered in the form of adaptation services, in order to construct a certain cross-level BPaaS adaptation framework (see Figure 13). This framework takes the view that adaptation rules are mappings from event patterns to adaptation workflows. In this

sense, the enactment of an adaptation rule would just need the deployment and execution of a certain adaptation workflow within an Adaptation Engine. However, in our case, such enactment is not performed via the static deployment of fixed adaptation workflows. On the contrary, the adaptation workflows mapping to the adaptation rules are defined in an abstract manner and then concretised into a form of a concrete workflow, which is deployable and executable by a workflow engine, based on the current adaptation capabilities of the system/framework as well as the broker preferences concerning the time and cost of the adaptation. Another major feature of the adaptation framework is that it exploits the previous work of FORTH [51] in the context of event pattern discovery for the semi-automatic derivation of the adaptation rules that can drive the adaptive behaviour of the BPaaS. Such rules can then be inspected via the use of a certain UI in order to be adjusted, if needed, by the respective expert. The latter expert has also the capability via this framework to specify new adaptation rules and incorporate them in the framework. Such a capability could be beneficial for the rapid addressing of novel or unforeseen situations, which can obviously not be handled by the current configuration of the adaptation framework.

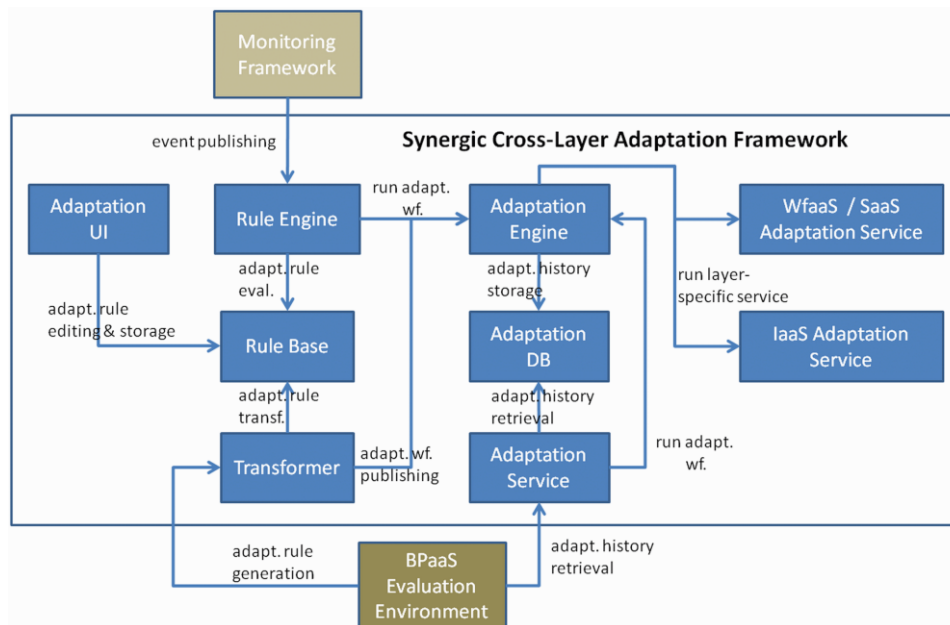


Figure 13: The combined Cross-Layer BPaaS Adaptation Framework

The detection of the adaptation rule to be triggered is performed via a Knowledge Base, which takes into account the events (patterns) that are submitted by the cross-level monitoring sub-framework in order to find out the right rule to enact. This Knowledge Base then passes the adaptation part of the rule, i.e., the (abstract) adaptation workflow, to the Transformer component, which takes care of the both the concretisation of the adaptation workflow as well as its transformation into a specific language (e.g., BPMN) that is supported by the framework's workflow engine. This workflow, as it can be easily understood, includes the execution of certain service-based tasks, which map to specific methods that are offered by the level-specific adaptation mechanisms of the framework (originating from the previous work of the two partners involved). The workflow engine exploited has also the capability to store the adaptation actions performed in the system, which can enable us to support various types of analysis over them. The current adaptation mechanisms supported include mainly adaptation actions at the infrastructure and service/SaaS level, such as scale out/in and service replacement.

The cross-layer adaptation framework was initially designed as a conceptualisation of the complete way the adaptation of BPaaS can be performed across different layers. However, the great integration and implementation effort involved in its construction did not match the available resources at the two partner sides. To this end, it was decided to go for a simplified version of this framework in which only a subset of all possible components can be realised. To this end, a respective implementation was realised as an extension of the Cloudiator framework. In this implementation, the already existing integration abstractions regarding the cloud providers and layers allowed the

seamless integration of the adaptation actions. As the adaptations are described in workflows, a workflow processing component to realize the execution of the aforementioned adaptation workflows (see deliverable D3.4 [12]) was also developed.

To realize a migration scenario (see also above example when explicating the adaptation workflows extension in CAMEL), the broker or an optimization tool would define an adaptation workflow with four steps: (i) instance creation, (ii) state copy, (iii) endpoint replacement and (iv) instance deletion. In the first step a new instance of a component is created in the target cloud of the migration. The state copy can be user-defined scripts that copy data to a new location. The third step updates the endpoint in the BPaaS workflow for the migrated instance. This means that as Cloudiator already caters for the wiring of components, the new instance is registered in the service chain (i.e., BPaaS workflow) as defined by the broker. The final step cleans up the old instance and frees the respective resources.

4.2 Lessons Learned

4.2.1 BPaaS Orchestration

The use of a cross-cloud orchestration tool has proven extremely useful in the project and has significantly increased the flexibility of the Allocation Environment in picking up the best-possible provider. The added support to the PaaS level proved that IaaS and PaaS can be served from the same abstraction layer. Yet, the step of adding PaaS support also has unveiled that the diversity of APIs and concepts used by PaaS operators is even higher than in the IaaS case. This is not so problematic on the user-side of the system, but rather on the back-end where the interaction with the various providers and their APIs takes place. Here, we could see that the coverage of cloud providers offered by abstraction layers, such as PUL, is not high enough. Other libraries cover a different sub-set of the landscape so that it would have been very attractive to apply multiple of them in order to increase the coverage.

During the project we worked on two separate (meta-) models of which the instantiations had to be synchronized: CAMEL and the Cloud Provider Engine of Cloudiator. This was due to the different target groups of the models. CAMEL needs to be used by modellers and thus provides support to the design of application models. Cloudiator, on the other side, offers a model that captures many technical details in a way that allows for rapid processing, including historical changes of the model and replication of some information. Furthermore, it requires a fine-grained REST-interface to cater for several components that use this interface.

Concerning this, a stronger synchronisation between CAMEL and the Cloud Provider Engine of Cloudiator would be useful. However, we would still argue against a common model for all purposes, as this would most likely end in many compromises that lower the usage of the model.

4.2.2 Cross-layer Monitoring and Adaptation

As described in Deliverable D3.4, we have developed a research prototype (Adaptation Management, AM) that runs independently of the Cloud Provider Engine (CPE) and which introduces (i) a new meta-model, and (ii) another synchronization method. The AM comprises a model that is very specific to adaptation actions and complies with the aforementioned CAMEL extensions. This means that the complex adaptation actions are not part of the CPE model but are rather outsourced to the AM. This has some advantages in respect to the focus and simplified usage of the AM and its model. However, it comes with the burden of synchronising between the AM and CPE. Finally, we found that the single adaptation actions are better to be directly handled by the CPE. At the time of the prototype development, this was hampered by the very monolithic architecture of the CPE with one central model. This was also a driver to decide for a more distributed architecture of the CPE, including a shared, non-centric model. This

will decouple the components of the CPE and help for an easier extension. This architecture refactoring is currently work in progress, and will continue in other research projects.

Another approach that is currently the focus of a feasibility study is the usage of a modelling language to define the processes (also the adaptation actions) that are executed inside the CPE. This integration of, e.g., BPMN, obviously would cause a large amount of development effort and therefore has to be checked carefully. The idea for this also arose from the fact that sometimes it is hard to know which processes inside the CPE conflict and cause specific errors, as the CPE comprises many concurrent processes with states that are hard to capture on any given point of time. This is where a clearly defined process management component (including process state machines) could help enormously.

Capitalising on the aforementioned issue, there is also the need to adapt the deployment workflow when the respective need arises. This was actually experienced various time during the deployment of BPaaSes. The remedy that was performed was to manually inspect the log file of the deployment and then modify the respective deployment part of the BPaaS bundle in order to adjust it accordingly. This is a rather cumbersome and time-consuming process. In this respect, it would be much better if this is replaced via a more automated approach. Such an approach could come via the application of log mining which could enable to derive the actual root cause of a certain deployment issue and then address it on the fly by constructing a certain deployment adaptation workflow based on the current adaptation capabilities of the system. A certain example of how this could be achieved would be to have a certain deployment workflow with underspecified exception handling for each workflow task which could be realised on the fly through the construction and execution of a corresponding adaptation workflow when the respective need arises, i.e., when the execution of the corresponding deployment task fails.

Concerning the cross-level monitoring and adaptation framework, we foresee the following extensions or improvements:

- (a) implementation of the rest of the framework components
- (b) thorough evaluation of the framework and especially its physical realisation alternatives in order to discover its better possible configuration within the (BPaaS management) cloud infrastructure;
- (c) capability to inject new adaptation functionality on the fly without disrupting the execution of the adaptation framework;
- (d) realisation of a nice UI for the editing and adjustment of adaptation workflows;
- (e) extension of the semi-automatic adaptation rule derivation approach with the capability to dynamically modify the adaptation rules inferred based on their actual performance at runtime and their level of success;
- (f) proposal of adaptation history analysis algorithms which can provide an excellent insight not only of the adaptation rule performance but also derive suitable knowledge that can be used for the further improvement of the BPaaS services managed;
- (g) capability to support the dynamic reconfiguration of the monitoring infrastructure to handle additional, unanticipated monitoring load;
- (h) capability to support monitoring infrastructure robustness including mechanisms for the back up of "fresh" and thus useful monitoring information.

Due to the lack of time and resources, the cross-level monitoring and adaptation framework was not fully implemented and integrated into the final CloudSocket platform version. We foresee that this can be realised in the near future, possibly in the context of a forthcoming European project. This is essential in order to support a quite extensive list of adaptation scenarios and possibilities for a BPaaS. This would certainly assist a broker in better managing a BPaaS and avoiding undesired situations from happening.

5 BPAAS EVALUATION ENVIRONMENT

A current design and allocation solution for a BPaaS is not always optimal. Furthermore, an organisation can evolve, making an optimal design and allocation solution sub-optimal or not optimal at all. In this respect, the goal of the BPaaS evaluation phase is to discover those problematic places in the design, allocation and provisioning of a BPaaS which can then lead to its adaptation and possible evolution, thus enabling to optimise the BPaaS and achieve the respective goals of the broker (such as profit maximisation).

The discovery of the aforementioned problematic places can take the form of various types of knowledge, which are collectively known as business intelligence knowledge. These types include the following:

- KPI evaluation: capability to evaluate the performance of a KPI through the assessment of KPIs
- KPI drill-down: capability to perform a hierarchical assessment of a KPI hierarchy/tree which can enable the detection of root causes of a certain problem, such as the violation of a high-level KPI.
- Best BPaaS deployment discovery: capability to dig into the execution & monitoring history of a BPaaS in order to derive the best possible deployments of the BPaaS at hand
- Event Pattern detection: capability to infer event patterns, which lead to the violation of KPIs. Such event patterns can lead to the semi-automatic production of BPaaS adaptation rules.
- Process mining: capability to mine the BPaaS log in order to derive new knowledge (e.g., recreation of process models, ability to detect execution paths that are never executed, etc.)

Most of the above types have been realised in the form of a BPaaS Evaluation Environment prototype which supports the derivation of business intelligence knowledge. Such an environment has the capability to connect to a BPaaS Design Environment in order to hand-over the knowledge derived to support the optimisation of the BPaaS.

5.1 Innovation Items

5.1.1 Information Harvesting

The derivation of the business intelligence knowledge requires the storage and maintenance of various type of information as well as its linking. Such information covers both execution/monitoring as well as allocation information for a BPaaS. The linking of information is important, however, in order to support a more sophisticated form of business intelligence knowledge derivation which spans multiple, interconnected pieces of information. Such a linking, if performed in a semantic manner, would also enable the production of mostly accurate knowledge, in contrast to other approaches in the state-of-the-art.

Based on the above rationale, FORTH has developed a Harvesting Engine [24][25]. This engine is able to harvest information from various components within the CloudSocket platform, such as the Cloud Provider Engine in order to retrieve (component-to-aaS service) allocation information, the Workflow Engine in order to collect workflow execution & monitoring information, the Monitoring Engine in order to retrieve monitoring information and the Repository Manager in order to collect service information. Apart from these internal components, the Harvesting Engine is also able to support the harvesting of information from external information sources. All the collected information is linked and semantically lifted based on the use of two main ontologies: (a) a BPaaS evaluation ontology which supports the modelling of whole BPaaS (dependency) hierarchies; (b) a KPI extension over the OWL-Q ontology, a non-functional service specification language. The final outcome is the storage of the semantically enriched information into a semantic repository (Semantic KnowledgeBase - SKB) which can then be queried in order to obtain the needed knowledge for supporting the various analysis capabilities offered by the BPaaS Evaluation Environment.

As KPI and other kind of analysis does not rely on raw information and by considering that the BPaaS landscape is not constantly changing, we have decided to execute the Harvesting Engine in a periodic manner. This can enable not overloading the CloudSocket platform as the collection and linking of all this information is not a very lightweight task. However, we plan to evaluate which should be the right periodicity of this information harvesting as a function of other parameters, such as the number of BPaaSes that need to be handled and the frequency via which high-level monitoring information should be delivered and evaluated.

5.1.2 Conceptual Analysis Engine

By reviewing the state-of-the-art in KPI evaluation and analysis, we observed the following shortcomings:

- (a) many approaches follow a static mapping approach of KPI metrics to underlying database queries. This leads to the effect that the expert needs to have a good knowledge of the database query, there is a lock in to a certain technology, and new KPIs are always associated to the need to create new queries that are mapped to them, spending considerable time in this mapping process;
- (b) many approaches only support a static list of KPI metrics. This does not allow them to be extensive while it can also lead to the issue that the incorporation of a new KPI might require a great re-engineering effort in the KPI evaluation system;
- (c) many approaches rely on a syntactic language for the specification of KPIs and their parts which is not expressive enough. Furthermore, such a language cannot be exploited in order to support the inferencing of new knowledge while it could lead to the production of inaccurate evaluation results;
- (d) most of the approaches do not enable the modeller to sufficiently explore the possible metric space in order to define the most suitable KPI (metric) for the current BPaaS at hand. However, as it has been proven in the literature, such capability is really essential in the mostly creativity-related task of designing the most suitable KPIs (metrics) for a certain BPaaS;
- (e) very few approaches support only one form of KPI drill-down via the use of machine learning techniques. However, such techniques require substantial and accurate historical knowledge in order to support the KPI drill-down which might not be always available, especially in the case of new BPaaSes. Furthermore, the KPI drill-down information produced might not be also accurate due to the non-consideration of semantic information (both in the definition of the KPIs and the specification of their measurements and metrics).

Based on the above drawbacks of the state-of-the-art, it has been decided to propose a novel semantic approach [24][25][35] for the evaluation and drill-down of KPIs, which relies on the following pillars:

- (a) the KPI extension of OWL-Q. This extension enables the complete semantic specification of KPIs and their parts (such as metrics and measurement units) while also catering for the validation of the KPI models and the production of added-value knowledge via the use of semantic rules. This language is also at a high-abstraction level independent of the underlying database technology. This enables the modeller to specify a KPI more close to the way he/she conceptualises it. This also assists in the better exploration of the KPI metric space as the modeller has additional freedom and time in order to specify a set of KPI metrics or evolutions of them before he/she finalises the KPI model of a certain BPaaS;
- (b) the on-the-fly mapping of OWL-Q KPI specifications to SPARQL queries which can be directly executed in the underlying Semantic Repository. This kind of mapping obviously does not require the modeller to have any kind of low-level technical language knowledge. As indicated in point/pillar (a), such a mapping also enables a better and more rapid exploration of the possible metric space. Furthermore, it also enables the production of more accurate evaluation results. Last but not least, this mapping enables independence from the underlying storage medium which provides for a high-level of flexibility in the metric definition and the capability to specify and support any kind of metric. While also the switching from one storage medium to another one is also possible;

- (c) the capability to support novel forms of KPI drill-down based on knowledge that is accurate and has been explicitly specified by an expert and not possibly inaccurate knowledge that has been derived from a machine learning approach. These novel forms of KPI drill-down follow naturally the relationships between KPIs and between KPI metrics in order to reach the actual level and respective component that is to be blamed for a certain KPI violation;
- (d) multi-tenancy: due to the way the information is stored by the Harvesting Engine, i.e., in separate graphs per BPaaS broker, our approach is able to support broker-based multi-tenancy. In other words, the various types of analysis can be restrained only on the actual information that pertains to a certain broker. This feature really provides an added value to our approach and enables it to discern from all other approaches in the literature.

Our semantic KPI analysis approach [24][25][35] has been realised in the form of a REST-based service. It not only provides the three types of analysis (KPI evaluation and two forms of KPI drill-down) but also additional functionality, such as the supply of the raw KPI metrics, which can be used for the specification of a novel composite KPI metric, and the set of tenants, for which measurements for a certain KPI have been produced (useful for visualisation purposes). Furthermore, two modes of KPI evaluation are offered. In the first mode, the KPI evaluation can be performed over a historical sub-graph of the current broker, which contains the measurements of high-level broker-specific KPIs. This mode enables the issuing of a simple query, which can be answered in an ultra fast manner. In the second mode, the KPI evaluation is performed on the fly over the measurements of low-level KPI metrics across the whole BPaaS history. This enables the actual well-advertised capability of the flexible exploration of the possible KPI metric space as the user can supply and evaluate any kind of KPI metric while it can vary both its computation formula as well as its evaluation period.

The KPI evaluation is performed via the transformation of the specification of the KPI metric (in OWL-Q) into a SPARQL query which is issued over the Semantic Knowledge Base (SKB). This transformation exploits mainly the KPI metrics computation hierarchy as well as the current measurement capabilities of the CloudSocket platform. In particular, it attempts to map the specification of component KPI metrics to their respective formulas until we reach the level of metrics for which measurements have been already collected and stored in the SKB. This transformation also takes into account the measurement window and schedule of the KPI metric as well as additional information given by the user/broker request, such as the exact evaluation period that restrains the actual measurement space to be exploited.

The KPI drill-down functionality builds over the KPI evaluation one and comes into two main forms:

- (a) KPI drill-down based on the parent-child relationships between KPIs. This enables us to go down into the KPI hierarchy in order to discover a low-level KPI that is to be blamed for the violation of the root KPI;
- (b) KPI drill-down based on parent-child relationships between KPI metrics. In this second form, we support a more fine-grained level of analysis as we do not stop at the level of low-level KPIs but we go even deeper by considering measurements of metrics for which KPIs have not been specified.

These two forms of KPI drill-down support different levels of analysis, which can play a complementary role to each other. The first form can be used for identifying problematic low-level KPIs and when such information is not enough, then the second form can be exploited in order to go deeper over more low-level metrics which can identify the root cause of a certain KPI violation. In our view, both this kind of KPI analysis and its two main forms represent another novelty of our work which can really provide knowledge even in cases where machine learning as an alternative technique for KPI drill-down cannot deliver appropriate or accurate results.

5.1.3 Hybrid Business Dashboard

The implementation of the Hybrid Business Dashboard is available as a feature of the BPaaS Design Environment. By visualising performance indicators, this component enables the integration between the requirements level in the BPaaS Design Environment and the analysis information/knowledge provided via the analytics engine. The Hybrid Business Dashboard is a web portal that gives the possibility to monitor the current status of all the KPIs defined in a BPaaS Design Package. A user-friendly interface has been provided in order to explore the KPIs and check if they are in line with the allowed value ranges or not. In the background, the *Conceptual Analytics Engine* in the BPaaS Evaluation Environment is exploited in order to derive the actual KPI metric values to be examined.

The Dashboard is model centred and this made it extremely dynamic and adaptable of any context thanks to the possibility to model in details what are the metrics, KPIs and Goals that would like to be monitored. This model is used by the dashboard as input and is based on the Cause & Effect model that follow the concepts defined in the Balanced Scorecard Framework and give the possibility to model Strategic and Operational Goals, the required KPIs and all the dependencies and relations among them.

The results are provided in a widget based cockpit that the user can combine and decide how must look like.

5.2 Lessons Learned

5.2.1 Lessons learned from Information Harvesting

The Harvesting Engine relied on the appropriate interfacing with other components from the CloudSocket platform. It also required some insights about the exact way the methods from the other CloudSocket components can be executed with the right order as well as the exact content of the information that is delivered. Based on the experience that has been faced, it should have been decided to follow a different realisation pathway in the overall CloudSocket platform where, instead of the current information polling mechanism, a pushing one should have been realised enabling:

- (a) to have one place where all information concerning the whole platform can be gathered and correctly correlated as well as queried and be exploited (e.g., for analysis purposes as in the case of the BPaaS evaluation);
- (b) to enable each component of the platform to deliver in the exact moment that is created the actual information that has to be delivered.

Based on this approach and by following a local-to-canonical model transformation method, we achieved the possibility to switch easily between components without any re-engineering effort with respect to the information collection functionality. Furthermore, we would be able to avoid the polling of information, which could become quite resource-intensive at particular time points, also stealing previous computation capabilities from other components of the platform, which might really need them.

Currently, CAMEL is exploited in order to support the modelling of various information aspects. The models of CAMEL also have the possibility to specify KPIs via the use of respective computation formulas. However, CAMEL needs also to be slightly extended in order to support all appropriate details that are needed for the specification of KPIs. For instance, it is not able to support the specification of the warning threshold for metric conditions as the latter could be considered as a representation of potential KPIs. In this direction, we foresee the use of two alternative directions:

- (a) either CAMEL is extended with the capability to fully specify KPIs;
- (b) there is a switch with respect to the non-functional aspects to OWL-Q across the whole platform.

The former direction requires a slight realisation effort while it is also consistent with the current mechanism of mapping CAMEL models to OWL-Q ones in the context of KPI evaluation. The second direction requires a substantial realisation effort but then leads to the use of a language, which is both semantic and naturally supports the specification of KPIs (based on its novel extension).

5.2.2 Lessons learned from KPI evaluation & drilldown

For both the KPI evaluation and drill-down functionality some form of validation has been performed according to particular use cases in the project. Future research shall consider other state-of-the-art approaches or, alternatively, realisations which equivalently represent them (e.g., in case their original code is not available) in order to better highlight the superiority of our KPI evaluation approach. A machine learning approach could unveil the superiority of each drill-down approach with respect to a machine learning approach as well as highlight any possible ways for synergy between these different kinds of approaches.

Implementation-wise, our KPI evaluation service does not support all possible mathematical operators that can be specified in OWL-Q. This is due to the current expressivity capabilities of SPARQL.

Apart from the KPI analysis functionality, we have also realised other forms of business intelligence knowledge derivation for BPaaSes. These forms, however, were not integrated in the final CloudSocket platform. By considering the added value of these forms as well as the service-based approach that has been used to realise them, the effort to integrate them will not be great and will be certainly worth it. This would certainly also raise the suitability, application and added value of the Hybrid Business Dashboard as it would enable it to visualise in suitable UI metaphors the results of all these forms of knowledge derivation. Furthermore, such an integration would enable passing that knowledge to the BPaaS designer, which would then take care of optimising the design of a BPaaS. In addition, this knowledge could pass over also to the BPaaS allocator which would have the ability to modify in a more optimal manner the current allocation solution for the BPaaS at hand. Besides, this allocator would also have the ability to inspect the adaptation rules that are semi-automatically produced by the BPaaS Evaluation Environment in order to adjust them, if needed, and include them in the specification of the BPaaS in order to better drive its adaptation behaviour.

5.2.3 Lessons learned from Hybrid Business Dashboard

In this section is provided the lesson learned during the development of the Hybrid Business Dashboard.

At the beginning the data source specification has been hard coded in the system. After some adaptation request we decided to externalize the definition to the modelling phase. So in the definition of the KPI model is possible to define also technical details about the data source position. Moreover has been required, in the case of the Evaluation Environment as data source, to give to possibility to specify custom http header of the REST request and in particular the one required for the basic http authentication.

With the same approach used for the data source, we decided to externalize also the algorithm used to evaluate Goals and KPIs. In such a way the modeller can define a goal and associate it to an existing algorithm or create a specific one.

The final users would like to personalize their view in the Dashboard. Some users may be interested in only some Goals while others only to some KPIs. The initial dashboard had a static interface and did not support such kind of scenarios, so we decide to introduce a widget based dashboard that the user can personalize in order to visualize only the KPIs or Goals of interest using the widget that they prefer (like a table chart instead of an Instagram chart for the same KPI).

We experienced that KPIs and Goals may have dependencies among each other's. This means that a Goal, in order to be evaluated, require the result of a sub-goals and the same for KPIs. At the beginning only the Sub-Goals definition was possible, so we changed the Dashboard architecture in order to support also the Sub-KPIs presence.

6 SUMMARY AND CONCLUSION

This document presents the evolution of the innovation items and tools over the life cycle of the CloudSocket project. First prototypes were available already after the first year of the project. In the next two years, these prototypes were further improved. This has the consequence that the solutions described in the earlier deliverables might not be up to date.

The document shows the richness of the research results and the sophisticated development that was achieved in the CloudSocket project. The innovation items made available to the public via the innovation shop on the project website.

A significant progress with respect to innovation has been made, which is measured with the indicators expressed in the innovation scorecard (see Table 1 in section 1.2), which shows the maturity of the innovation items. For nearly all innovation items there exist peer-reviewed publications and they are implemented as a prototype. A majority is already integrated in a tool by the technology partners and thus ready for exploitation.

The analysis of the innovation scorecard validates the efficiency of the agile approach of the project with the parallelisation of research (WP3) and implementation (WP4), because the research results (i.e. innovation items) could be integrated in the second implementation cycle, which itself was challenged by the demonstration (WP5).

In addition to the progress described, the evaluation of the individual innovation items also open issues are identified, for which further research is needed.

7 REFERENCES

- [1] A, Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, 2011., Web Services Agreement Specification (WS-Agreement), Errata Update, Open Grid Forum (OGF).
- [2] APQC, 2014. Process Classification Framework Version 6.1.1.
- [3] Baur, D., Domaschka, J., 2016. Experiences from Building a Cross-cloud Orchestration Tool. In: Proceedings of the 3rd Workshop on CrossCloud Infrastructures & Platforms. CrossCloud '16, New York, NY, USA, ACM 4:1–4:6
- [4] Cortés, A.R., Martín-Díaz, O., Toro, A.D., Toro, M.: Improving the Automatic Procurement of Web Services Using Constraint Programming. *Int. J. Cooperative Inf. Syst.* 14(4), 439(468 (2005).
- [5] Domaschka, J., Baur, D., Seybold, D., Griesinger, F., 2015. Cloudiator: A Cross-Cloud, MultiTenant Deployment and Runtime Engine. In: 9th SummerSoC.
- [6] Domaschka, J., Seybold, D., Griesinger, F., Baur, D., 2016. Axe: A Novel Approach for Generic, Flexible, and Comprehensive Monitoring and Adaptation of Cross-Cloud Applications. In: European Conference on Service-Oriented and Cloud Computing. Springer International Publishing, Cham, 184–196
- [7] Domaschka, J., Griesinger, F., Baur, D., Rossini, A., 2015. Beyond Mere Application Structure: Thoughts on the Future of Cloud Orchestration Tools. *Procedia Computer Science* 68, 151 – 162
- [8] EC Cloud Select Industry Group (C-SIG), 2014. Cloud Service Level Agreement Standardization Guidelines.
- [9] Falcioni, D. (Ed.), 2017. Explanatory Notes for the Final BPaaS Prototype, D4.6 – D4.7 – D4.8. https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D4.6_D4.7_D4.8-v1.0.pdf
- [10] FHNW, Forschungsbericht CLIMB-Broker.
- [11] Griesinger, F. (Ed.), 2017. CloudSocket Integration Report D4.9, https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D4.9_CloudSocket_Integration_Report-v1.0-2_FINAL.pdf
- [12] Griesinger, F. (Ed.), 2016. BPaaS Allocation and Execution Environment Prototypes. CloudSocket Project Deliverable D3.4. <https://site.cloudsocket.eu/documents/251273/350509/D3.4/>
- [13] Griesinger, F., Seybold, D., Domaschka, J., Kritikos, K., Woitsch, R., 2017. A DMN-based Approach for Dynamic Deployment Modelling of Cloud Applications. In: *Adv. in Service Oriented and Cloud Comp.—Workshops of ES OCC 2016*. CCIS, Springer.
- [14] Hinkelmann K. (Ed.) 2016: Explanatory Notes: Modelling Prototypes for BPaaS. CloudSocket Project Deliverable D3.2. https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D3.2_Modelling_Prototypes_for_BPaaS_Final.pdf
- [15] Hinkelmann, K. et al., 2016. A Modelling Environment for Business Process as a Service. In Springer, Cham, pp. 181–192. Available at: http://link.springer.com/10.1007/978-3-319-39564-7_18 [Accessed September 16, 2017].
- [16] Hinkelmann, K. et al., 2016. A Semantically-Enhanced Modelling Environment for Business Process as a Service. In 2016 4th International Conference on Enterprise Systems (ES). IEEE, pp. 143–152. Available at: <http://ieeexplore.ieee.org/document/7880484/> [Accessed September 16, 2017].
- [17] Hinkelmann, K., Laurenzi, E., Martin, A., Thönssen, B., 2018 (to appear). Ontology-Based Metamodeling. In Dornberger, R., ed.: *Business Information Systems and Technology 4.0 - New Trends in the Age of Digital Change*. Springer Berlin Heidelberg
- [18] Hinkelmann, K. (Ed.), 2015: Modelling Framework for BPaaS. CloudSocket Project Deliverable 3.1, https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D3.1_BPaaS_Design_Environment_Research_v1.0_20151231-FINAL.pdf

- [19] Hinkelmann, K., Gerber, A., Karagiannis, D., Thoenssen, B., Van Der Merwe, A., & Woitsch, R., 2016. A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. *Computers in Industry*, 79, pp. 77-86, <https://doi.org/10.1016/j.compind.2015.07.009>
- [20] Iranzo, J. (Ed.) 2016. Explanatory Notes: First BPaaS Prototype. Deliverable c. https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D4.2_D4.3_D4_4-v2.0.pdf
- [21] Laurenzi, E., Hinkelmann, K., Reimer, U., Van Der Merwe, A., Sibold, P., & Endl, R. (2017). DSML4PTM: A domain-specific modelling language for patient transferal management. In *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems (Vol. 3)*.
- [22] Karagiannis, D., Buchman, R.A., Burzynski, P., Reimer, U. and Walch, M., 2016. Fundamental Conceptual Modeling Languages in OMiLAB. In Karagiannis, D.; Mayr, H.C. and Mylopoulos, J. *Domain-Specific Conceptual Modeling*. Cham: Springer International Publishing, pp. 3–30. Available at: http://link.springer.com/10.1007/978-3-319-39417-6_1.
- [23] K. Kritikos, J. Domaschka and A. Rossini, 2014: "SRL: A Scalability Rule Language for Multi-cloud Environments," 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, Singapore, 2014, pp. 1-9. doi: 10.1109/CloudCom.2014.170
- [24] Kritikos, K. (Ed.) 2016. BPaaS Monitoring and Evaluation Blueprint, Deliverable D3.5. <https://site.cloudsocket.eu/documents/251273/350509/D3.5>
- [25] Kritikos, K. (Ed.) 2017. BPaaS Monitoring and Evaluation Prototypes, Deliverable D3.6. June 2017.
- [26] Kritikos, K., Laurenzi, E. & Hinkelmann, K., 2017. Towards Business-to-IT Alignment in the Cloud. In *BPM@Cloud 2017: 1st International Workshop on Business Process Management in the Cloud*, in conjunction with the 6th European Conference on Service-Oriented and Cloud Computing (ESOCC). Oslo, Norway.
- [27] Kyriakos Kritikos, Kostas Magoutis, Dimitris Plexousakis: Towards Knowledge-Based Assisted IaaS Selection. *CloudCom 2016*.
- [28] K. Kritikos and D. Plexousakis, 'Semantic QoS Metric Matching', in *ECOWS*, 2006, pp. 265–274.
- [29] K. Kritikos and D. Plexousakis, 'Mixed-Integer Programming for QoS-Based Web Service Matchmaking', *IEEE Trans Serv Comput*, vol. 2, no. 2, pp. 122–139, 2009.
- [30] Kyriakos Kritikos, Dimitris Plexousakis: Multi-cloud Application Design through Cloud Service Composition. *CLOUD 2015*: 686-69.
- [31] Kyriakos Kritikos, Dimitris Plexousakis: Semantic SLAs for Services with Q-SLA. *CF 2016*.
- [32] Kyriakos Kritikos, Dimitris Plexousakis: Towards Combined Functional and Non-functional Semantic Service Discovery. *ESOCC 2016*: 102-117.
- [33] Kritikos, K., & Plexousakis, D. (2016). Subsumption Reasoning for QoS-Based Service Matchmaking. *Service-Oriented and Cloud Computing - 5th IFIP WG 2.14 European Conference, ESOCC 2016*, Vienna, Austria, September 5-7, 2016, (pp. 87-101).
- [34] Kritikos, K. & Plexousakis, D., 2015. TOWARDS SEMANTIC-BASED CLOUD APPLICATION MANAGEMENT. *Services Transactions on Cloud Computing*, 3(3). Available at: https://site.cloudsocket.eu/documents/251273/401914/STCC_Towards+Semantic+Based+Application+Management_FORTH.pdf/39862d13-c0ca-4b36-9c26-dcae0dff7317 [Accessed December 21, 2017].
- [35] Kritikos, K., Plexousakis, D., & Woitsch, R. (2017). Towards Semantic KPI Measurement. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, Porto, Portugal, 24 - 26 April (pp. 63-74).
- [36] Kritikos, Kyriakos; Zeginis, Chrysostomos; Griesinger, Frank; Seybold, Daniel and Domaschka, Jörg, 2017: A Cross-Layer BPaaS Adaptation Framework, *International Conference on Future Internet of Things and Cloud (FiCloud)*.
- [37] Object Management Group, 2011. Business Process Model and Notation (BPMN), Version 2.0. Available at: <http://www.omg.org/spec/BPMN/2.0/PDF>

- [38] Object Management Group: Decision Model and Notation (DMN) V1.1. Technical report, OMG, <http://www.omg.org/spec/DMN/1.1/PDF> (2015)
- [39] Palma, D., Spatzier, T., 2013. Topology and Orchestration Specification for Cloud Applications Version 1.0. OASIS Standard.
- [40] Popovici, A (Ed.), 2017. BPaaS Implementation for Use Case Sites, D5.5, CloudSocket Project https://site.cloudsocket.eu/documents/251273/350514/CloudSocket_D5.5_BPaaS+Implementation+for+use+case+sites+v1.0-FINAL.pdf
- [41] Rossini, A., 2016. Cloud Application Modelling and Execution Language (CAMEL) and the PaaSage Workflow. In: Adv. in Service-Oriented and Cloud Comp.—Workshops of ESOC 2015. Volume 567 of CCIS., Springer, 437–439
- [42] Rossini, A., Kritikos, K., Nikolov, N., Domaschka, J., Griesinger, F., Seybold, D., Romero, D., 2015. D2.1.3 CAMEL documentation. PaaSage project deliverable.
- [43] Seybold, D (Ed.), 2016. BPaaS Allocation and Execution Environment Blueprints. CloudSocket Project Deliverable D3.3, https://site.cloudsocket.eu/documents/251273/350509/CloudSocket_D3.3_BPaaS-Allocation-Execution.pdf
- [44] Silver, Bruce R., 2011. BPMN method and style: with BPMN implementer’s guide, Cody-Cassidy Press.
- [45] The Open Group, 2017, ArchiMate® 3.0.1 Specification. <http://pubs.opengroup.org/architecture/archimate3-doc/> [Accessed December 22, 2017].
- [46] Utz, W., 2015. Cloud Transformation Framework, D2.3. https://dev.cloudsocket.eu/web/guest/documents/251273/350509/CloudSocket_D2.3_Transformation_Framework_v1.0.pdf
- [47] Woitsch, R. (Ed.) Final CloudSocket Architecture. CloudSocket Project Deliverable D4.5
- [48] Woitsch, R. et al., 2016. Business Process as a Service (BPaaS): The BPaaS Design Environment. Available at: <http://ceur-ws.org/Vol-1600/session1p1.pdf> [Accessed December 21, 2017].
- [49] Woitsch, R. & Utz, W., 2015. Business Processes as a Service (BPaaS): A Model-Based Approach to align Business with Cloud offerings. Available at: <https://zenodo.org/record/35583#.WbDscNhLfmE> [Accessed September 7, 2017].
- [50] Xiong, P., Pu, C., Zhu, X., and Griffith, R. “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.
- [51] Zeginis, Ch., et al. (2013). Event Pattern Discovery in Multi-Cloud Service-Based Applications. *International Journal of Systems and Service-Oriented Engineering, Special Issue on Service-Oriented Architecture in the Era of Big Data*, 5(4), 78-103.
- [52] Zeginis, Ch., et al. (2013). Towards cross-layer monitoring of multi-cloud service-based applications, in *Service-Oriented and Cloud Computing*, Springer, 2013, pp. 188-195.